ConSol*CM Process Designer Handbuch (bis CM-Version 6.9.3)

Inhalt

1 Einleitung zum ConSol*CM Process Designer 6
1.1 ConSol*CM für das Business Process Management 7
1.2 Über dieses Handbuch 8
1.2.1 Bevor Sie dieses Handbuch lesen ... 8
1.2.2 Struktur dieses Handbuchs 8
1.2.3 Erklärung der Hinweissymbole in diesem Handbuch 8
1.3 Business-Prozesse 10
1.4 Einleitung zu Workflows in ConSol*CM 11
1.5 Der ConSol*CM Process Designer auf einen Blick 12
1.5.1 Workflow-Modellierung 12
1.5.2 Tickets und Aktivitäten 13
1.5.3 Drag-and-Drop-Modellierung der Workflow-Komponenten 14
1.5.4 Scopes und Verschachteln von Scopes 15
1.5.5 Modellierung von Eskalationsmechanismen (Trigger und Wartezustände) 15
1.5.6 Modellierungen von Interrupts und Exceptions 16
1.5.7 Skripte und ihre Fähigkeiten 16
1.5.8 Versionierung von Workflows 17
2 Grundkomponenten von ConSol*CM-Prozessen 18
2.1 Allgemeine Objekte 19
2.2 Datenfelder 21
2.2.1 Datenfelder in ConSol*CM-Versionen 6.8 und niedriger 21
2.2.2 Datenfelder in ConSol*CM-Versionen 6.9 und höher 21
2.3 Standard-Ticketdatenfelder 23
3 ConSol CM Process Designer Handbuch - Arbeiten mit der Process Designer Applikation 24
3.1 Arbeiten mit der Process Designer Applikation 25
3.1.1 Benötigte Schritte für einen neuen Prozess 25
3.1.2 Starten des Process Designers 25
3.2 Process Designer GUI 27
3.2.1 Einleitung zu den Process Designer GUI-Elementen 27
3.2.2 Der Skript-Editor 45
4 ConSol CM Process Designer Handbuch - Komponenten von ConSol CM Workflows 46
4.1 Komponenten von ConSol*CM-Workflows 47
4.1.1 Einleitung 47
4.2 Workflow-Komponenten: START-Knoten 48
4.2.1 Einleitung 48
4.2.2 Eigenschaften eines Startknotens 49
4.3 Workflow-Komponenten: ENDE-Knoten 50
4.3.1 Einleitung 50
4.3.2 Eigenschaften eines Endknotens 52
4.4 Scopes (Bereiche) 53
4.4.1 Einleitung zu Scopes (Bereichen) 53
4.4.2 Definieren eines neuen Scopes 55

4.4.3 Eigenschaften eines Scopes 58
4.4.4 Scopes und Sichten 59
4.5 Workflow-Komponenten: Aktivitäten 60
4.5.1 Einleitung zu Aktivitäten 60
4.5.2 Eigenschaften einer Aktivität 63
4.5.3 Prozesslogik von Aktivitäten 65
4.5.4 Beispiele für Aktivitäten 65
4.6 Workflow-Komponenten: Entscheidungsknoten 70
4.6.1 Einleitung zu Entscheidungsknoten 70
4.6.2 Eigenschaften eines Entscheidungsknotens 70
4.6.3 Beispiel für einen Entscheidungsknoten 71
4.7 ConSol CM Process Designer Handbuch - Adornments (Trigger und ACFs) 75
4.7.1 Adornments (Trigger und ACFs) 75
4.7.2 Zeit-Trigger 76
4.7.3 Mail-Trigger 88
4.7.4 Business-Event-Trigger 97
4.7.5 Aktivitätsformulare (ACFs) 109
4.8 Aussprung- und Einsprungknoten 117
4.8.1 Einleitung 117
4.8.2 Aussprungknoten 118
4.8.3 Einsprungknoten 120
5 Prozesslogik 122
5.1 Aktivitäten 123
5.2 Interrupts und Exceptions 124
5.2.1 Einleitung zu Interrupts und Exceptions 124
5.2.2 Interrupts 124
5.2.3 Exceptions 125
5.3 Schleifen (Fehler in Workflows) 126
5.4 Prozesslogik von Zeit-Triggern 127
5.5 Prozesslogik von Business-Event-Triggern 128
6 ConSol CM Process Designer Handbuch - Workflow-Programmierung 129
6.1 Workflow-Programmierung 130
6.1.1 Einleitung 130
6.1.2 Zusätzliche Werkzeuge für die Workflow-Programmierung 131
6.1.3 Anmerkungen über die Syntax von Methoden 131
6.2 Wichtige Klassen und Objekte 133
6.2.1 Einleitung 133
6.2.2 Wichtige Objekte 133
6.2.3 Convenience-Klassen und -Methoden 134
6.3 Arbeiten mit Datenfeldern 137
6.3.1 Einleitung zu Datenfeldern 137
6.3.2 Datentypen für Datenfelder 139
6.3.3 Benutzerdefinierte Felder für Ticketdaten 140
6.3.4 Datenfelder für Kundendaten 149
6.3.5 Datenfelder für (unsichtbare) Variablen nutzen 157
6.4 Senden von E-Mails 158

6.4.1 Einleitung zum Senden von E-Mails 158
6.4.2 Wichtige Methoden 159
6.4.3 Beispiele 159
6.5 Arbeiten mit Pfadinformationen 165
6.5.1 Einleitung 165
6.5.2 Abrufen von Pfadinformationen eines Workflow-Elements 165
6.5.3 Beispiele für die Verwendung von Pfadinformationen 166
6.6 Arbeiten mit Kalendern und Zeiten 167
6.6.1 Einleitung 167
6.6.2 Rechnen mit Daten und Zeiten ohne CM-Arbeitszeitkalender 168
6.6.3 Rechnen mit Daten und Zeiten mit CM-Arbeitszeitkalender 169
6.7 ConSol*CM Process Designer Handbuch - Arbeiten mit Objektrelationen 170
6.7.1 Arbeiten mit Objektrelationen 170
6.7.2 Arbeiten mit Ticketrelationen 171
6.7.3 Arbeiten mit Kundenrelationen (Datenobjektrelationen) 179
6.8 Suche nach Tickets und Kunden mittels der ConSol*CM Workflow API 188
6.8.1 Einleitung 188
6.8.2 Suche nach Tickets 188
6.8.3 Suche nach Units (Kontakten und Firmen) 192
6.9 Debug-Informationen 194
6.9.1 Einleitung 194
6.9.2 Befehle für den Debug-Output 194
7 Best Practices 196
7.1 Die grundlegende Organisation eines Workflows: Verwendung von Bereichen (Scopes) 197
7.1.1 Variante A: Verwendung eines globalen Scopes 197
7.1.2 Variant B: Verwendung von drei oder mehr Haupt-Scopes 198
7.2 Die Position des START-Knotens 200
7.3 Speichern Sie einige Workflow-Skripte im Admin-Tool 201
7.3.1 Wann Admin-Tool-Workflow-Skripte verwendet werden 201
7.3.2 Wie Admin-Tool-Workflow-Skripte verwendet werden 201
7.4 Bedenken Sie die Verwendung von Trigger-Kombinationen genau 203
7.5 Stoßen Sie keine unnötigen Ticket-Update-Events an 206
7.6 Wie Sie den Parameter "Automatische Aktualisierung deaktivieren" verwenden 207
7.7 Vermeiden Sie selbstauslösende Business-Event-Trigger 209
8 Installieren von Workflows 210
8.1 Einleitung und Workflow-Lifecycle 211
8.2 Für die Workflow-Installation benötigte Bearbeiter-Berechtigungen 212
8.3 Aktionen während der Workflow-Installation 213
9 Appendix A - Liste der Annotationen 215
9.1 Alphabetische Liste der Feld-Annotationen (bis Version 6.9.3) 216
9.2 Alphabetische Liste der Annotationen für Benutzerdefinierte Feldgruppen (Version 6.8 und niedriger) 232
9.3 Alphabetische Liste der Annotationen für Benutzerdefinierte Feldgruppen (Version 6.9 und höher) 242
10 Appendix B - Glossar 245
11 Appendix C - System-Properties 252

11.1 System-Properties sortiert nach Modul 253
11.2 System-Properties sortiert nach Name der System-Property 303
12 Appendix D - Hinweise zu Marken 353
13 Index 354

1 Einleitung zum ConSol*CM Process Designer

- ConSol*CM für das Business Process Management
- Über dieses Handbuch
  - Bevor Sie dieses Handbuch lesen ...
  - Struktur dieses Handbuchs
  - Erklärung der Hinweissymbole in diesem Handbuch
- Business-Prozesse
- Einleitung zu Workflows in ConSol*CM
- Der ConSol*CM Process Designer auf einen Blick
  - Workflow-Modellierung
  - Tickets und Aktivitäten
  - Drag-and-Drop-Modellierung der Workflow-Komponenten
  - Scopes und Verschachteln von Scopes
  - Modellierung von Eskalationsmechanismen (Trigger und Wartezustände)
  - Modellierungen von Interrupts und Exceptions
  - Skripte und ihre Fähigkeiten
  - Versionierung von Workflows

1.1 ConSol*CM für das Business Process Management

ConSol*CM ist ein **Business Process Management System mit Kundenfokus.** Mit Hilfe von ConSol*CM können Sie Geschäftsprozesse kontrollieren und steuern. Der Fokus liegt dabei auf der menschlichen Kommunikation und Interaktion, beispielsweise Prozesse im Bereich Helpdesk, Customer Service, Marketing, Vertrieb oder Einkauf. Grundsätzlich lässt sich jeder in einem Unternehmen eingesetzte Prozess mit ConSol*CM abbilden und zum Leben erwecken.

Mit ConSol*CM können Sie alle für Ihre Business-Prozesse relevanten Komponenten miteinbeziehen, um die Prozesse Ihres Unternehmens optimal darzustellen und zu steuern. ConSol*CM wird in verschiedenen Industriezweigen und Branchen eingesetzt, von Versicherungen und Banken über die Fashion-Industrie bis hin zu Autowaschanlagen oder Produzenten von Ticketautomaten. Die flexible Gestaltung von Prozessen und die Workflow-Engine stellen eine perfekte Basis für die Erstellung und Steuerung von Business-Prozessen aller Art dar.

1.2 Über dieses Handbuch

1.2.1 Bevor Sie dieses Handbuch lesen ...

Wenn Sie dieses Handbuch lesen, setzt Ihr Unternehmen wahrscheinlich ConSol*CM als Business Process Management Tool ein und Ihre Aufgabe ist es, das System zu administrieren und die Prozesse Ihres Unternehmens in dem System zu implementieren. Dieses Handbuch wird Ihnen dabei helfen, das Prinzip von ConSol*CM-Workflows zu verstehen und die Arbeit mit dem Process Designer zu erlernen. Zahlreiche *Tipps und Tricks* von unseren erfahrenen Consultants werden Ihnen dabei helfen, die beste Vorgehensweise zu finden, wie Sie Ihre Prozesse verbessern können.

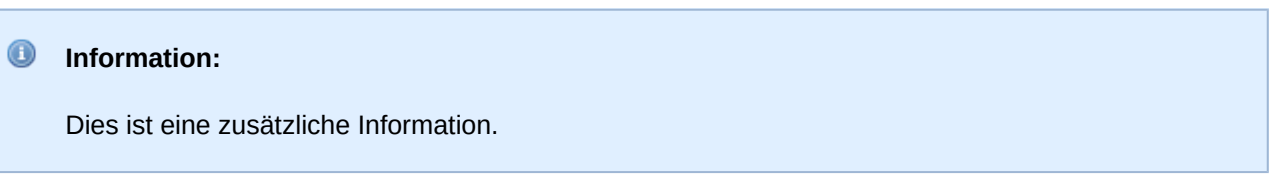
Bevor Sie anfangen, mit dem Process Designer zu arbeiten, sollten Sie fundierte Kenntnisse bezüglich der ConSol*CM-Administration besitzen, da das Programmieren von CM-Workflows die Nutzung verschiedener CM-Komponenten voraussetzt, welche konfiguriert werden, bevor (oder während) die Workflow-Entwicklung stattfindet. Lesen Sie daher bitte zuerst das *ConSol*CM Administratorhandbuch*.

1.2.2 Struktur dieses Handbuchs

1. Zuerst werden einige grundlegenden Komponenten von Business-Prozessen allgemein erklärt (siehe dazu diesen Abschnitt).
2. Danach folgt ein Überblick über die Implementierung von diesen Prozessen in ConSol*CM (siehe Abschnitt Grundkomponenten von ConSol*CM-Prozessen)
3. Darauf folgt eine detaillierte Erklärung des Process Designers (siehe Abschnitte Arbeiten mit der Process Designer Applikation und Komponenten von ConSol*CM Workflows)
4. Die Abschnitte Prozesslogik, Workflow-Programmierung und Best Practices liefern Expertenwissen über die Entwicklung von Workflows.
5. Da jeder Workflow installiert werden muss, um aktiviert zu werden, wird dieses Thema im Abschnitt Installieren von Workflows behandelt.
6. In den Appendizes finden Sie Listen mit allen wichtigen Begriffen, die in diesem Handbuch verwendet werden (Glossar), mit allen Annotationen (wichtig für das GUI-Design) und den System-Properties (wichtig für das CM-Systemmanagement). Bitte beachten Sie außerdem die Seite mit Informationen zum Markenrecht.

1.2.3 Erklärung der Hinweissymbole in diesem Handbuch

Folgende Icons dienen zur Markierung und/oder Hervorhebung von Informationen:



Attention:

Dies ist ein wichtiger Hinweis. Sie finden ihn an Stellen, an denen besondere Vorsicht geboten ist.

Warning:

Dies ist eine Warnung. Warnungen sind von größter Wichtigkeit und betreffen meist Stellen, die für das reibungslose Funktionieren des Systems relevant sind.

Tip:

Dies ist ein Tipp. Tipps sind meist Empfehlungen, die aus den Erfahrungen des Consultings stammen.

1.3 Business-Prozesse

Innerhalb eines Business-Prozesses muss eine bestimmte Anzahl von Aufgaben in einer festgelegten Reihenfolge ausgeführt werden, um ein spezifisches Ziel zu erreichen.

Die folgenden Komponenten sind (üblicherweise) in einem Business-Prozess relevant. Bitte lesen Sie den Abschnitt Grundkomponenten von ConSol*CM-Prozessen, um einen Überblick über die ConSol*CM-Objekte zu erhalten, die diesen Komponenten entsprechen.

- **Prozess**
  DIes ist eine Sammlung von Aufgaben, die in einer bestimmten Reihenfolge ausgeführt werden müssen. Diese Aufgaben können in serieller Reihenfolge stehen oder parallel ausgeführt werden. In ConSol*CM wird ein Prozess mit einem oder mehreren Workflows abgebildet. ConSol*CM kann sowohl einzelne Prozesse als auch komplexe Prozessketten abbilden.
  Jeder Prozess muss einen definierten Input und einen definierten Output besitzen. Das Objekt, das einen Fall repräsentiert und sich durch den Prozess bewegt, ist ein *Ticket*. Für den Endbenutzer kann das Ticket als *Ticket*, *Vorgang* oder mit jedem anderen gewünschten Namen bezeichnet werden.
- **Rollen und Verantwortlichkeiten**
  Üblicherweise besitzen die Personen, die innerhalb eines Prozesses arbeiten, unterschiedliche Rollen, d.h. unterschiedliche Verantwortlichkeiten. In ConSol*CM kann jeder Bearbeiter, d.h. jede Person, die mit dem System arbeitet, eine oder mehrere Rollen besitzen.
- **Zugriffsberechtigungen**
  Ein Business Process Management System kann unterschiedliche Prozesse eines Unternehmens steuern. Daher ist die Vergabe und Steuerung von Zugriffsberechtigungen eine Kernfunktionalität. In ConSol*CM werden die Zugriffsberechtigungen über Rollen zugewiesen.
- **Kunde**
  Dies ist die Person, die Interesse am Resultat des Prozesses besitzt. In ConSol*CM gibt es immer einen Hauptkunden für ein Ticket. Dies kann eine Person sein, d.h. ein Kontakt, oder eine Firma. Es können einem Ticket auch noch mehr Kunden (zusätzliche Kunden) hinzugefügt werden.
- **Aufgaben**
  Innerhalb eines Business-Prozesses kann es verschiedene Arten von Aufgaben geben:
  - Manuelle Aufgaben
  - Systemgestützte Aufgaben
  - Vollautomatische Aufgaben

  ConSol*CM kann alle Typen von Aufgaben managen. Für manuelle Aufgaben existieren To-Do-Listen für die Bearbeiter und verschiedene Mechanismen, die sicherstellen, dass keine Aufgabe vergessen oder ignoriert wird.

1.4 Einleitung zu Workflows in ConSol*CM

Eine der Kernkomponenten von ConSol*CM ist eine leistungsstarke Workflow-Engine. Daher wird ein Prozess in ConSol*CM durch einen **Workflow** repräsentiert. Dieser ist eine technische Repräsentation der aufeinanderfolgenden Schritte, die zur Erfüllung aller Aufgaben, die innerhalb des Business-Prozesses ausgeführt werden sollen, benötigt werden.

Beispiele

In einer IT-Helpdesk-Umgebung könnte ein Workflow aus den folgenden Schritten bestehen: *Neues Ticket - Ticket akzeptieren - An einer Lösung arbeiten - Den Kunden informieren - Ticket schließen.*

In einem Sales-Prozess könnten die Schritte sein: *Erstkontakt: Lead - Zweitkontakt: Opportunity - Vertragskandidat - Vertrag.*

Der Workflow, der alle benötigten Schritte enthält, läuft in einer Workflow-Engine. Mit diesem Handbuch werden Sie die Details über die Komponenten eines Workflows kennenlernen und lernen, wie man diese benutzt, um einen Workflow, der Ihren Business-Prozess repräsentiert, zu entwickeln.

Ein Workflow ...

- repräsentiert einen bestimmten Prozess, z.B. die Schritte, die vollzogen werden müssen, um eine Kundenanfrage zu bearbeiten.
- stellt Aktivitäten und Entscheidungen in eine bestimmte Reihenfolge.
- legt die möglichen Wege fest, die ein Ticket einschlagen kann.

Der Vorgang oder die Anfrage, die bearbeitet werden soll, wird durch ein **Ticket** repräsentiert, d.h. das Objekt, das sich durch den Workflow bewegt.

Das folgende Bild zeigt eine grafische Repräsentation eines einfachen Helpdesk-Prozesses:

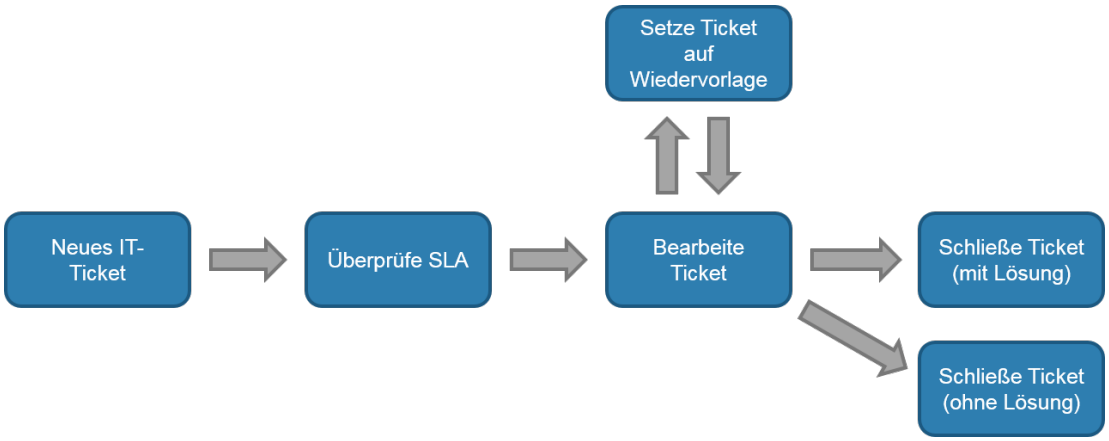


Fig. 1: ConSol*CM Process Designer - Prozess: Vereinfachte Darstellung

1.5 Der ConSol*CM Process Designer auf einen Blick

1.5.1 Workflow-Modellierung

Ein Business-Prozess wird in ConSol*CM mit dem *Process Designer* modelliert, einer Applikation, die integraler Bestandteil einer Standardinstallation von ConSol*CM ist. Ein Prozess kann durch einen oder mehrere *Workflows* repräsentiert werden, d.h. Sie benutzen den *Process Designer,* um *Workflows* zu entwickeln.

> ⚠ In der ConSol*CM-Terminologie bezeichnet *Workflow* immer die technische Entität, während *Prozess* den Business-Prozess aus der logischen bzw. der Management-Sicht bezeichnet. Ein Prozess kann durch einen oder mehrere *Workflows* repräsentiert werden.

Einer der Vorteile des Process Designers ist, dass es im Prozedere keine Lücke zwischen Workflow-Design und Workflow-Implementierung gibt. Sie können einen Workflow für einen Prozess mittels der grafischen Benutzeroberfläche des Process Designers entwickeln und, sobald Sie den Workflow einer Queue zugewiesen sowie die Rollen und Bearbeiter definiert haben, wird der Prozess lebendig und die Bearbeiter können damit arbeiten. Dies bedeutet, dass Sie den Process Designer für beide Schritte, die für die Erstellung eines IT-gestützten Business-Prozesses wichtig sind, benutzen können:

- Modellierung und Entwicklung des Prozesses aus der logischen Perspektive
- Implementierung des Prozesses in einer technischen Instanz

Aufgrund dieser Flexibilität können Sie mit einer einfachen Version eines Workflows beginnen, üblicherweise in einer Test-Umgebung, und iterativ die gewünschten Funktionalitäten des Prozesses entwickeln. Das Team der Bearbeiter kann in jedem Schritt des Entwicklungs- und Optimierungsprozesses testen, ob die Anwendungsfälle wie gewünscht abgebildet werden.

Die grafische Repräsentation eines Workflows im Process Designer ist der *Business Process Model and Notation* (BPMN) sehr ähnlich, so dass Sie intuitiv arbeiten können.

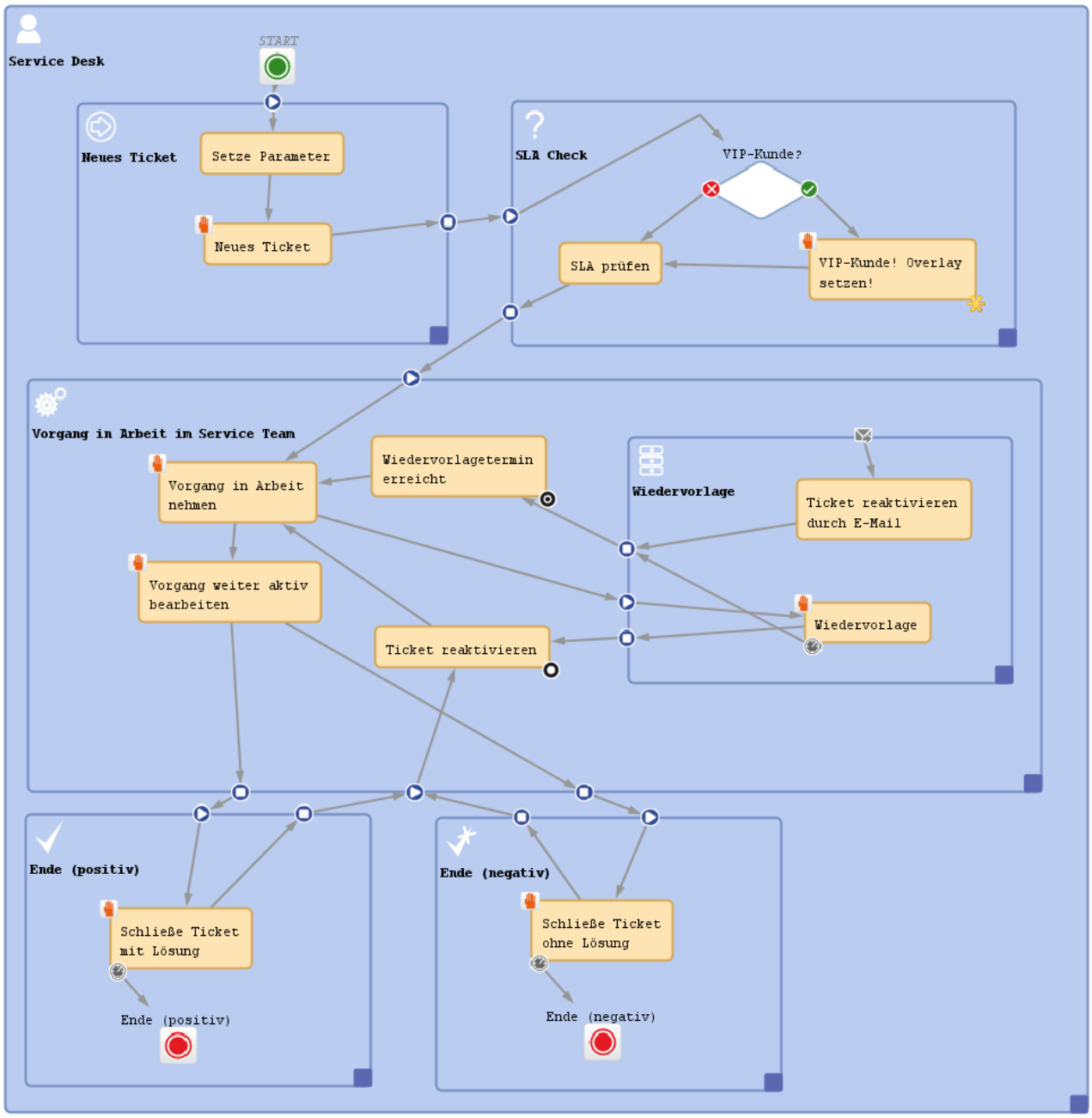


Fig. 2: ConSol*CM Process Designer - Workflow-Modellierung des Prozesses aus dem vorherigen Bild

Bitte lesen Sie die folgenden Abschnitte, um einen ersten Eindruck von den Features und Funktionalitäten des Process Designers zu erhalten. Alle Themen werden in den entsprechenden Kapiteln dieses Handbuchs im Detail erklärt.

1.5.2 Tickets und Aktivitäten

Jeder Vorgang, der bearbeitet werden soll, wird durch ein *Ticket* repräsentiert. Dies bedeutet, dass ein Ticket einen konkreten Durchlauf durch den Workflow darstellt. Dies kann eine Anfrage, eine Bestellung oder jegliche andere Aufgabe sein, die in einem Business-Prozess bearbeitet werden soll.

Wenn ein neues Ticket in ConSol*CM erstellt wird, wird es einem Workflow zugewiesen (mittels der Queue, zu der es gehört). Zuerst befindet sich das neue Ticket im START-Knoten. Im Laufe seines weiteren Lebenszyklus bewegt sich das Ticket durch die verschiedenen Aktivitäten des Workflows. Sein Lebenszyklus endet, wenn es einen ENDE-Knoten erreicht.

Sie modellieren einen Prozess in einem Workflow, indem Sie Aktivitäten in einer bestimmten Reihenfolge verbinden. Das Ergebnis ist ein gerichteter Flussgraph. Er zeigt, welche Aktivitäten ausgeführt werden müssen, damit sich ein Ticket erfolgreich durch den Workflow (und damit den Business-Prozess) bewegt. Workflows können Zweige besitzen, so dass verschiedene Flusswege möglich sind. Auf diese Weise können Sie sicherstellen, dass ein Ticket z.B. zuerst akzeptiert werden muss, daraufhin muss das Problem gelöst werden und danach muss die Lösung dokumentiert werden. Nur dann kann das Ticket geschlossen werden.

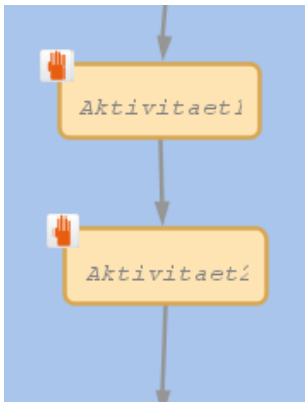


Fig. 3: ConSol*CM Process Designer - Zwei sequenzielle manuelle Aktivitäten

Es gibt manuelle und automatische Aktivitäten. Manuelle Aktivitäten erfordern eine Interaktion von einem Bearbeiter und werden als Workflow-Aktivitäten im CM Web Client angeboten. Automatische Aktivitäten werden im Gegensatz dazu ohne menschliche Eingabe ausgeführt und sind vor den Bearbeitern versteckt. Dies ermöglicht es ConSol*CM, dem Bearbeiter Zeit zu sparen und Daten aus verschiedenen Quellen hinter den Kulissen zu verarbeiten. Nur wenn eine Interaktion mit einem Bearbeiter nötig ist, stoppt der Prozess und wartet auf die Eingabe eines Bearbeiters.

Fig. 4: ConSol*CM/Web Client - Workflow-Aktivitäten

1.5.3 Drag-and-Drop-Modellierung der Workflow-Komponenten

Sie können Ihren Workflow einfach und intuitiv mittels Drag-and-Drop modellieren. Ziehen Sie die benötigten Workflow-Elemente, z.B. eine Aktivität oder einen Entscheidungsknoten, aus der Palette in den Arbeitsbereich und verlinken Sie sie. Danach stellen Sie die Eigenschaften der Elemente im Eigenschaften-Editor ein.

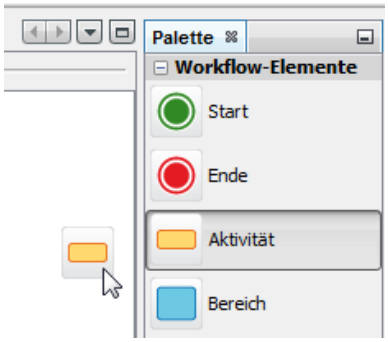


Fig. 5: ConSol*CM Process Designer - Drag-and-Drop einer Aktivität

Mittels der Grundelemente erstellen Sie Schritt für Schritt komplexe Workflows. Auf diese Weise können Sie selbst die kompliziertesten Business-Prozesse modellieren.

1.5.4 Scopes und Verschachteln von Scopes

Bei einem Prozess läuft ein Ticket durch verschiedene Status, z.B. *Neues Ticket*, *Vorqualifikation*, *Aktive Arbeit* und *Dokumentation*. Es kann auch für eine bestimmte Zeit auf Wiedervorlage gesetzt werden müssen. All diese Status werden durch Scopes repräsentiert. In jedem Scope können sich eine oder mehr Aktivitäten befinden. Dadurch ist es einfach, Workflows mit einer klaren Struktur zu entwickeln. Scopes können auch hierarchisch organisiert werden, z.B. wenn ein Ticket während der Dokumentation auf Wiedervorlage gesetzt werden muss. Auf diese Weise können Sie mit hierarchischen Scopes auch komplizierte Prozesse nachbilden. Sie wählen die Detailtiefe dabei jederzeit selbst aus.

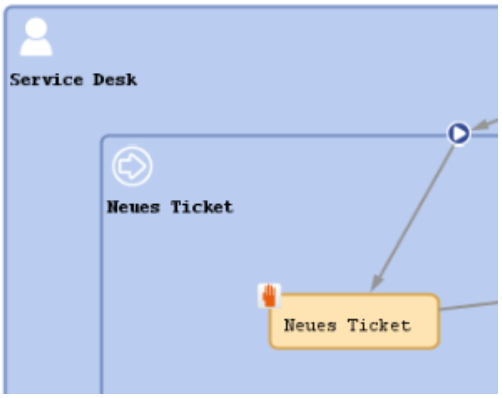


Fig. 6: ConSol*CM Process Designer - Verschachtelung von Scopes

1.5.5 Modellierung von Eskalationsmechanismen (Trigger und Wartezustände)

Bei den meisten Business-Prozessen ist das Einhalten von Terminen und Deadlines unentbehrlich. ConSol*CM unterstützt durch automatische Trigger die Einhaltung von Deadlines und bewahrt vor Verzögerungen. Diese Trigger messen z.B. die Reaktionszeit oder initialisieren Erinnerungsnachrichten.

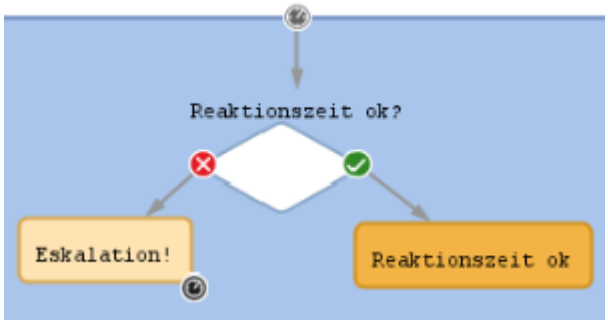


Fig. 7: ConSol*CM Process Designer - Triggern von Aktionen

1.5.6 Modellierungen von Interrupts und Exceptions

Im realen Arbeitsalltag werden die Aufgaben eines Prozesses nicht immer Schritt für Schritt vollzogen, sondern können durch außergewöhnliche Ereignisse unterbrochen werden. Dies können verschiedene äußere Ereignisse sein. Solche Interrupts sequenziell zu modellieren, ist häufig sehr komplex oder sogar unmöglich. Der Process Designer stellt umfangreiche Tools zu diesem Zweck bereit.

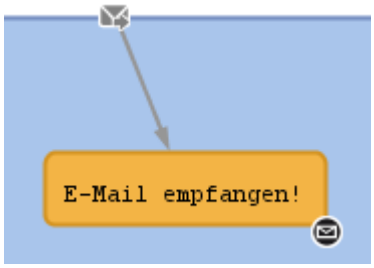


Fig. 8: ConSol*CM Process Designer - Modellierung von Interrupts

1.5.7 Skripte und ihre Fähigkeiten

Der Prozess, der als ConSol*CM-Workflow modelliert wurde, kann nicht nur aus Grundelementen wie Aktivitäten oder Entscheidungsknoten bestehen. Zu jedem Knoten eines Workflows kann ein Skript hinzugefügt werden, um Intelligenz eines Prozesses durch Programmierung zu liefern. Es können beispielsweise E-Mails an Kunden oder Bearbeiter verschickt, Interaktionen mit anderen Systemen implementiert oder Tickets übergeben werden. Grundsätzlich können alle Operationen, die durch Groovy-Skripte implementiert werden können, ausgeführt werden.

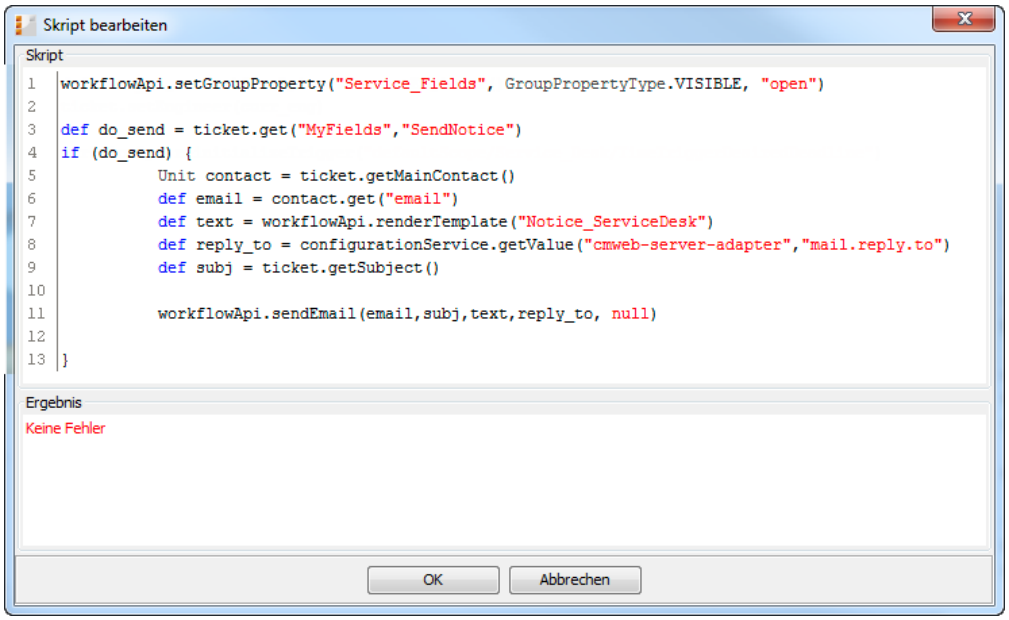


Fig. 9: ConSol*CM Process Designer - Skript einer Aktivität

1.5.8 Versionierung von Workflows

Business-Prozesse verändern sich ständig, z.B. aufgrund sich verändernder ökonomischer und technischer Anforderungen. Der Process Designer stellt eine kontinuierliche Versionierung der installierten Workflows bereit. Auf diese Weise können Sie einen neuen Workflow auf einfache Weise verwerfen (z.B. wenn Sie während der Systementwicklung eine neue Installation getestet haben) und zu einer der vorherigen Versionen zurückkehren.

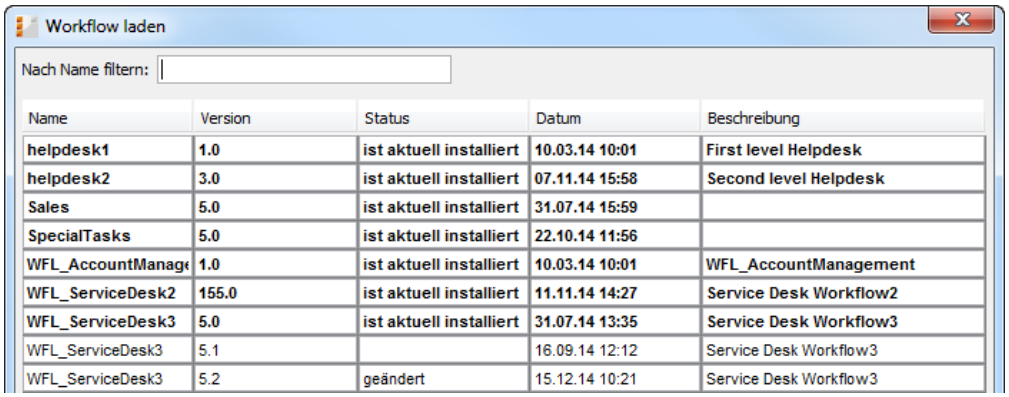


Fig. 10: ConSol*CM Process Designer - Workflow-Versionen

2 Grundkomponenten von ConSol*CM-Prozessen

- Allgemeine Objekte
- Datenfelder
  - Datenfelder in ConSol*CM-Versionen 6.8 und niedriger
  - Datenfelder in ConSol*CM-Versionen 6.9 und höher
- Standard-Ticketdatenfelder

2.1 Allgemeine Objekte

Während des Prozess-Designs und der Workflow-Entwicklung arbeiten Sie hauptsächlich mit den folgenden Objekten:

Notwendige Objekte:

- **Ticket**
  Dies repräsentiert den Vorgang. Abhängig vom Anwendungsfall kann dies z.B. ein Helpdesk-Fall, eine Sales-Opportunity, eine Bestellung oder eine Service-Anfrage sein.
- **Hauptkunde**
  Dies ist die Person, d.h. der Kontakt, welcher der Klient, der Initiator des Tickets ist. In ConSol*CM Version 6.9 und höher kann dies auch eine Firma sein. Der Kunde repräsentiert die externe Seite eines Tickets.
- **Queue**
  Dies ist die organisatorische Einheit innerhalb des ConSol*CM-Systems, die Tickets eines (Fach-) Bereichs gruppiert und die der zentrale Punkt für die Zuweisung von Zugriffsberechtigungen und des Workflows ist. Eine Queue besitzt genau einen Workflow, welcher nicht geändert werden kann. Innerhalb einer Firma kann es beispielsweise eine Queue für die Sales-Abteilung, eine für die Abteilung Customer Service und eine für die interne IT geben.
- **Bearbeiter**
  Dies ist die Person, die für die Erfüllung der Aufgaben in einem Ticket verantwortlich ist. Ein ConSol*CM-Bearbeiter besitzt ein Login und ein Passwort für den Web Client. Der Hauptbearbeiter wird teilweise auch als Ticket-Owner bezeichnet. Dieser kann sich während des Prozesses ändern.
- **Workflow**
  Dies ist das logische und technische Modell eines Prozesses. Ein Workflow wird einer Queue zugewiesen (und kann mehr als einer Queue zugewiesen werden). Dies führt dazu, dass alle Tickets, die sich in dieser Queue befinden, den Prozess durchlaufen, der durch diesen Workflow abgebildet wird. Die Workflow-Elemente, z.B. Aktivitäten, Bedingungen oder Entscheidungen, repräsentieren die wichtigsten Mittel in ConSol*CM, um den Prozessfluss zu konfigurieren und zu steuern. Ein Workflow kann einer oder mehreren Queues zugewiesen werden, z.B. können das IT-Servicedesk-Team und das Customer-Service-Team beide mit dem Workflow *ServiceWorkflow* arbeiten.
- **Benutzerdefinierte Felder** (CM-Versionen 6.8 und 6.9) und **Datenobjektgruppenfelder** (ab CM-Version 6.9)
  Dies sind die Datenfelder, die zur Definition des Datenmodells für das Ticket und der Kundendaten benutzt werden. Sie legen auch das GUI-Design des Web Clients fest. Benutzerdefinierte Felder werden nie auf der Basis eines einzelnen Felds definiert, sondern immer in *Benutzerdefinierten Feldgruppen*.
  In ConSol*CM-Version 6.9 und höher werden die Datenfelder für Ticketdaten *Benutzerdefinierte Felder* und die Datenfelder innerhalb des Kundendatenmodells *Datenobjektgruppenfelder* genannt.

Optionale Objekte:

- **Ein oder mehrere zusätzliche Kunde(n)**
  Neben dem Hauptkunden, d.h. dem Hauptkontakt oder (Version 6.9 oder höher) der Hauptfirma, können zusätzliche Kontakte (oder Firmen) zum Ticket hinzugefügt werden. Jedem zusätzlichen Kunden kann eine Kundenrolle zugewiesen werden. Es könnte z.B. einen Stellvertreter für denjenigen, der das Ticket eröffnet hat, geben oder der Team-Manager soll ebenfalls der Kontakt für einen Support-Fall sein. Ein zusätzlicher Kunde kann während des Prozesses der Hauptkunde werden und umgekehrt.
- **Ein oder mehrere zusätzliche Bearbeiter**
  Einem Ticket können zusätzliche Bearbeiter mit speziellen Rollen, die nach Bedarf definiert werden, zugewiesen werden. Zum Beispiel könnte ein Supervisor als zusätzlicher Bearbeiter gesetzt werden, um eine Genehmigung zu erteilen (Rolle *Genehmiger*), oder ein QA-Teammitglied kann mit der Rolle *QA* zum Ticket hinzugefügt werden, um das Ergebnis zu überprüfen, bevor das Ticket geschlossen wird.

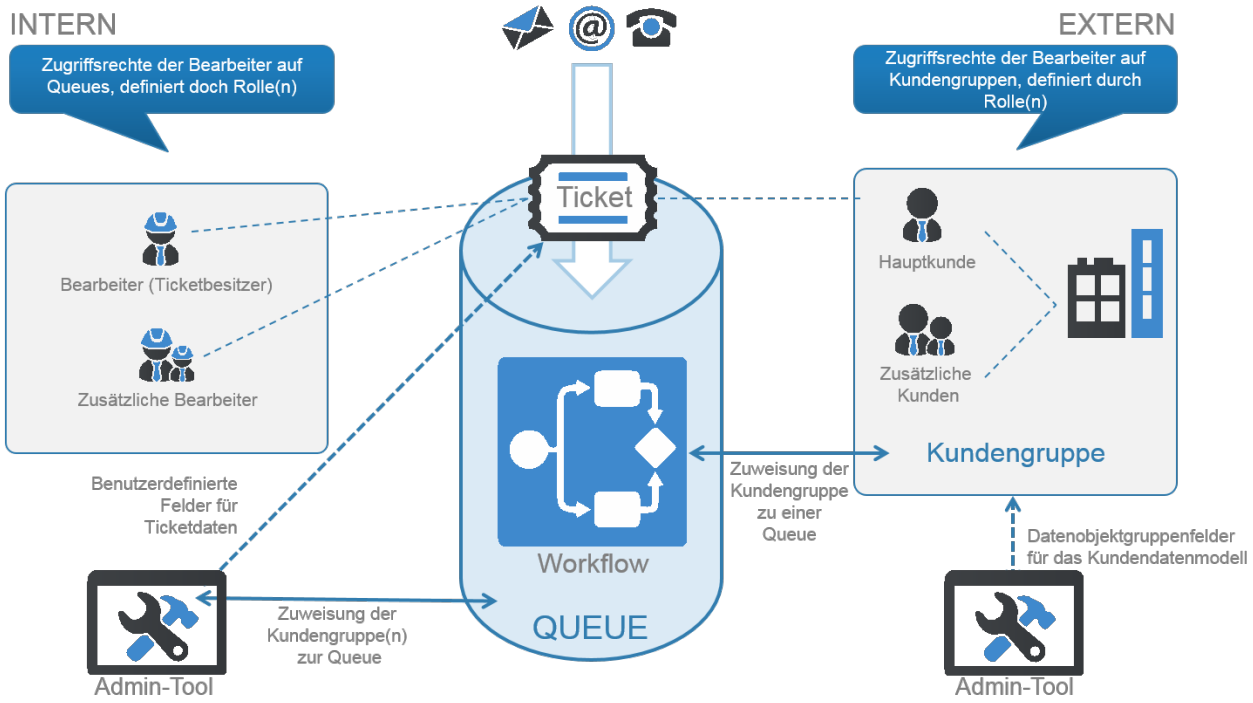


Fig. 1: ConSol*CM - Grundprinzip

2.2 Datenfelder

2.2.1 Datenfelder in ConSol*CM-Versionen 6.8 und niedriger

Benutzerdefinierte Felder sind Datenfelder eines bestimmten Datentyps, die Ticketdaten oder Kundendaten enthalten können. Die Benutzerdefinierten Felder definieren in ihrer Gesamtheit das Datenmodell des ConSol*CM-Systems. Alle Benutzerdefinierten Felder können nach Bedarf konfiguriert werden, d.h. Sie als Systemadministrator können so viele Benutzerdefinierte Felder wie benötigt erstellen und diese in der GUI (also dem CM Web Client) dort platzieren, wo Sie es wollen oder wo sie aufgrund der Usability am besten angeordnet werden sollten.

Benutzerdefinierte Felder werden immer in Benutzerdefinierten Feldgruppen verwaltet, niemals auf Basis eines einzelnen Feldes. Natürlich können Sie den Wert eines einzelnen Feldes auslesen oder setzen, wenn Sie ein Workflow-Skript schreiben, aber im Admin-Tool sowie im Process Designer kann eine große Anzahl von Operationen nur für Benutzerdefinierte Feldgruppen durchgeführt werden, z.B. Einblenden der Gruppe, Platzierung der Gruppendaten in einem Tab oder Zuweisung der Benutzerdefinierten Feldgruppe zu einer Queue. In Skripten greifen Sie auf Felder im Allgemeinen über die folgende Schreibweise zu:

Zugriff auf den Inhalt von Benutzerdefinierten Feldern, CM-Versionen 6.8 und niedriger

```
ticket:
ticket.get("<group name>.<field name>")
unit:
unit.get("<field name>")
```

Die initiale Erstellung von Benutzerdefinierten Feldgruppen und Benutzerdefinierten Feldern erfolgt im Admin-Tool. Ticketdaten werden in der *Verwaltung der Benutzerdefinierten Felder* definiert, jeweils in den entsprechenden Tabs für *Ticketdaten* und *Kundendaten*.

2.2.2 Datenfelder in ConSol*CM-Versionen 6.9 und höher

Beginnend mit der CM-Version 6.9.0 existieren zwei Typen von Datenfeldern:

- **Benutzerdefinierte Felder**
  Zur Definition von Ticketdaten, verwaltet in Benutzerdefinierten Feldgruppen, wie bekannt aus vorherigen CM-Versionen.
- **Datenobjektgruppenfelder**
  Zur Definition von Kundendaten als Teil von FlexCDM, dem neuen Kundendatenmodell. Verwaltet in Datenobjektgruppen.

Sie können auf den Inhalt eines Benutzerdefinierten Felds oder eines Datenobjektgruppenfelds mittels der folgenden Schreibweise zugreifen:

Zugriff auf den Inhalt von Datenobjektgruppenfeldern, CM-Versionen 6.9 und höher

```
ticket:
ticket.get("<group name>.<field name>")
unit, for one field:
unit.get("<group name>:<field name>")
```

2.3 Standard-Ticketdatenfelder

Manche Felder müssen nicht als Benutzerdefinierte Felder im Admin-Tool definiert werden, da sie immer präsent sind. Dies sind die folgenden Felder eines Tickets:

- **Ticket-ID**
  Unsichtbar für den Bearbeiter, nur interner Gebrauch in der Datenbank.
- **Ticket-Name**
  Sichtbar im Web Client, normalerweise Ticketnummer genannt.
- **Ticket-Thema**
  Muss bei den meisten Systemen gesetzt sein. Durch die System-Property *cmweb-server-adapter*, *commentRequiredForTicketCreation* kann dies umgestellt werden.
- **Eröffnungsdatum**
  Wird automatisch vom System gesetzt.
- **Bearbeiter**
  Kann *Null* oder einer der Bearbeiter sein.
- **Queue**
  Die aktuelle Queue, in der sich das Ticket befindet.

3 ConSol CM Process Designer Handbuch - Arbeiten mit der Process Designer Applikation

3.1 Arbeiten mit der Process Designer Applikation

- Benötigte Schritte für einen neuen Prozess
- Starten des Process Designers

3.1.1 Benötigte Schritte für einen neuen Prozess

Die Arbeit mit dem Process Designer ist einer der ersten Schritte in der Abfolge von Schritten, die Sie durchführen müssen, wenn Sie einen neuen Prozess mit Bearbeitern, Rollen usw. erstellen möchten. Bevor wir Ihnen die Arbeit mit dem Process Designer erklären, erhalten Sie an dieser Stelle daher eine kurze Liste der Aufgaben, die zu erledigen sind:

1. Designen und installieren Sie den Workflow mit dem Process Designer.
2. Erstellen Sie eine neue Queue mit diesem Workflow. Hier benötigen Sie außerdem die Definiton aller erforderlichen Benutzerdefinierten Felder und Kundengruppen.
3. Erstellen Sie die Sichten für die neuen Bearbeiter mittels der Scopes des neuen Workflows.
4. Erstellen Sie eine oder mehrere Rolle(n), die Zugriff auf die neue Queue besitzen. Beachten Sie dabei, dass der Zugriff auf die Kundengruppe(n) dem Zugriff der Queue entsprechen muss.
5. Erstellen Sie einen oder mehrere Bearbeiter und weisen Sie diesen die neue(n) Rolle(n) zu.
6. Überprüfen Sie das Login in den Web Client. Können Sie in der neuen Rolle ein Ticket erstellen?

3.1.2 Starten des Process Designers

Sie können den Process Designer auf jedem PC oder Laptop starten, auf dem ein Standard-Webbrowser installiert ist (lesen Sie dazu bitte die *System Requirements*) und der Netzwerk-Zugriff zum ConSol*CM-Server und zur ConSol*CM-Datenbank besitzt.

Um den Process Designer zu starten, öffnen Sie die ConSol*CM-Startseite und klicken auf den Link zum Process Designer. Für den Start der Process Designer Applikation ist Java Web Start (JWS) erforderlich, das auf der lokalen Maschine läuft. Da JWS heute ein integraler Bestandteil aller Java-Distributionen ist, sollte dies kein Problem darstellen.

Information:

Falls der Process Designer nicht gestartet werden kann, könnte dies an der Netzwerkverbindung liegen. Überprüfen Sie bei Windows-Systemen die Java-Parameter für Netzwerkeinstellungen: *Direkte Verbindung* könnte notwendig sein oder es kann erforderlich sein, einen spezifischen Proxy zu setzen. Bei Linux-Systemen überprüfen Sie bitte die Proxy-Einstellungen.

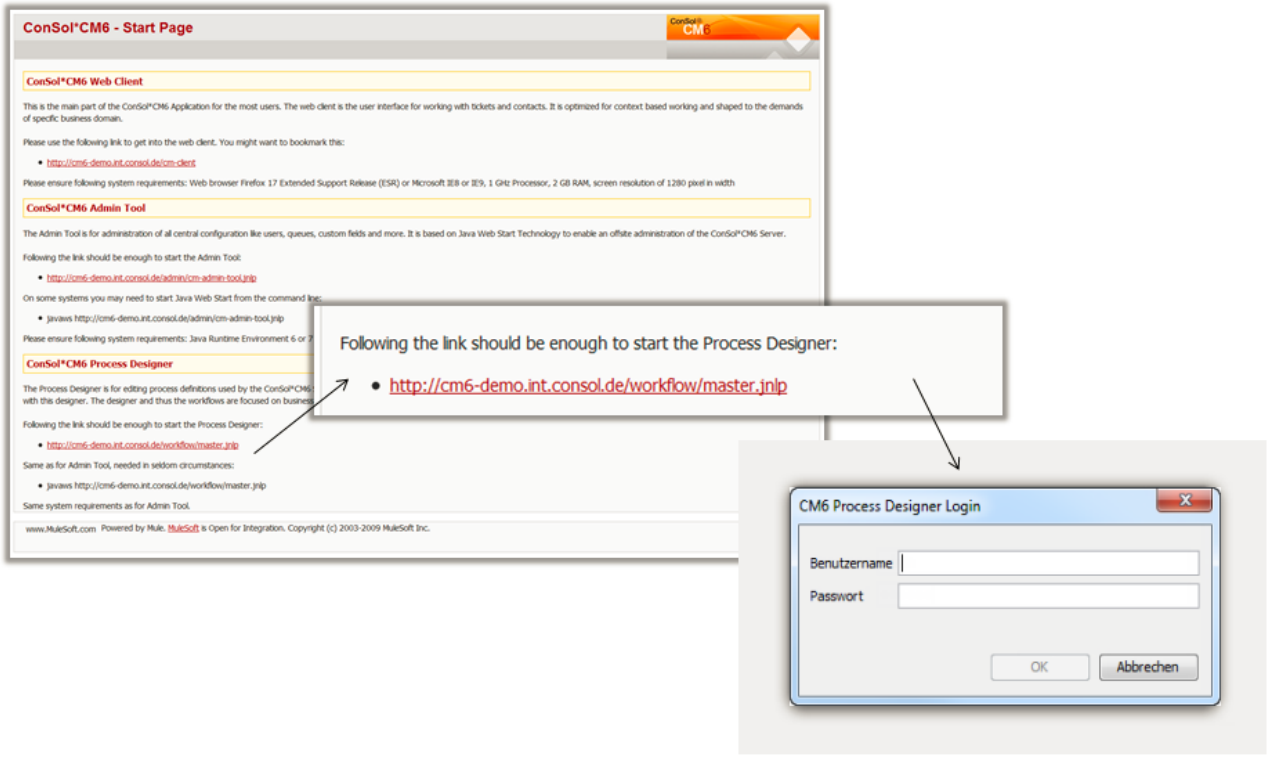


Fig. 1: ConSol*CM - Starten des Process Designers

Loggen Sie sich mit dem Administrator-Account oder mit einem Account, der Berechtigungen für die Workflow-Verwaltung besitzt, ein. Bitte lesen Sie für Details dazu das *ConSol*CM Administratorhandbuch*, Abschnitt *Rollenverwaltung*.

3.2 Process Designer GUI

- Einleitung zu den Process Designer GUI-Elementen
  - Überblick: GUI-Bereiche
  - Hauptmenü
  - Workflow-Bearbeitungsbereich
    - Laden und Löschen von Workflows
      - Laden eines Workflows
      - Löschen eines Workflows
  - Palette für Elemente und Adornments
    - Elemente
    - Adornments
  - Der Eigenschaften-Editor (Beispiel: Aktivität)
- Der Skript-Editor

3.2.1 Einleitung zu den Process Designer GUI-Elementen

Überblick: GUI-Bereiche

Die Process Designer GUI enthält die folgenden Elemente, die Sie auf dem nächsten Bild und der Liste unten sehen:

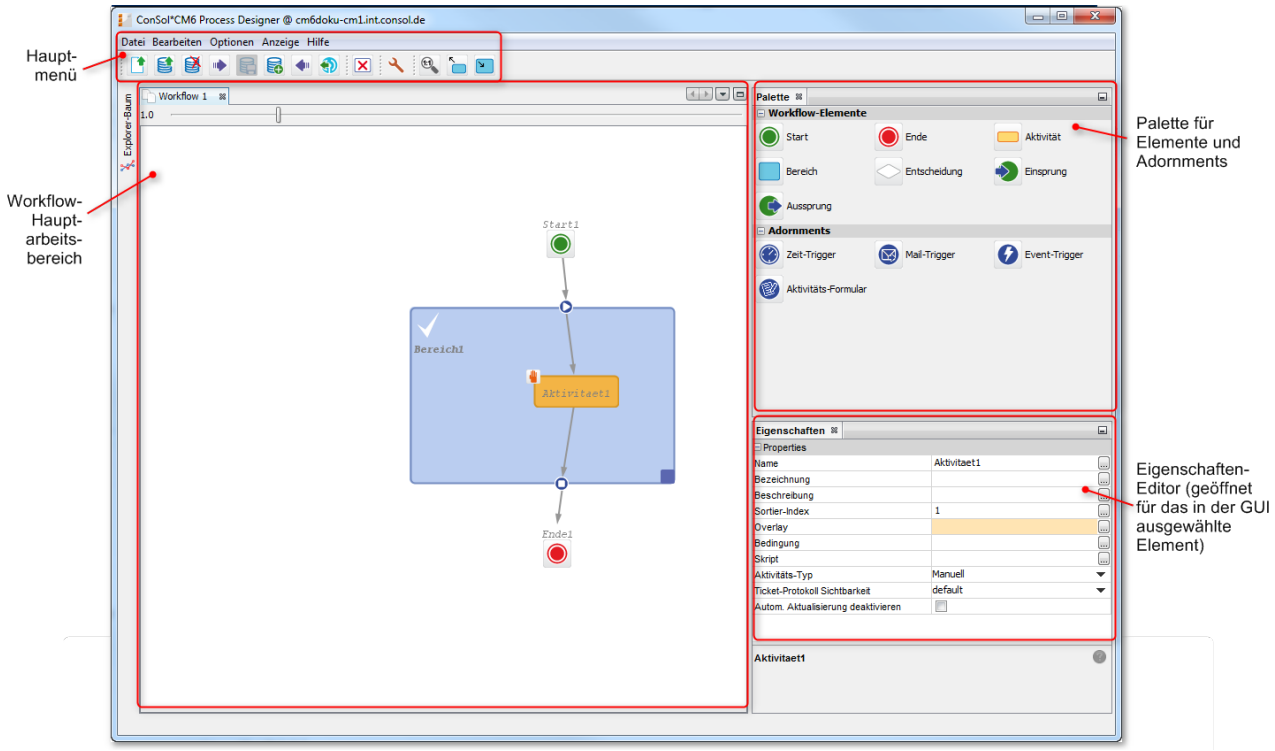


Fig. 1: ConSol*CM Process Designer - GUI-Elemente

Hauptmenü

Das Hauptmenü enthält die Menüpunkte als Texteinträge und eine Liste von Menü-Icons.

Hauptmenü-Eintrag	Menü-Untereintrag	Icon	Bemerkung
Datei			
	Neu ...		Einen neuen Workflow beginnen/erstellen
	Laden ...		Laden eines Workflows. Öffnet eine Tabelle mit existierenden Workflows, siehe Abschnitt Laden eines Workflows.
	Löschen ...		Löschen eines Workflows. Öffnet eine Tabelle mit existierenden Workflows, siehe Abschnitt Löschen eines Workflows.
	Importieren ...		Importieren eines Workflows aus einer Datei (in einem proprietären Workflow-Format).
	Speichern ...		Speichern des Workflows (existierende Version).
	Speichern als neue Version		Speichern des Workflows als neue Version.
	Exportieren ...		Export des Workflows in eine Datei. Öffnet den Dateibrowser des Betriebssystems. Der Workflow wird in einem proprietären Workflow-Formar (*.par*) gespeichert.

Hauptmenü-Eintrag	Menü-Untereintrag	Icon	Bemerkung
	Installieren ...		(Speichern als neue Version und) Installieren des Workflows, d.h. Installieren des Workflows im System. Das System wird Sie dafür vielleicht nach einer Entscheidung fragen (siehe Abschnitt Aktionen während der Workflow-Installation): • Position im Prozess beibehalten. • Prozess neu starten (Ticket am Startknoten neu anfangen lassen).
	Login		Login in den Process Designer. Normalerweise wird das Login-Fenster direkt nach dem Start des Process Designers angezeigt. Für das Login wird ein Account mit Administrator-Berechtigungen oder mit den Berechtigungen zur Workflow-Verwaltung (siehe *ConSol*CM Administratorhandbuch*, Abschnitt *Rollenverwaltung*) benötigt.
	Logout		Logout. Beendet nicht den Process Designer.

Hauptmenü-Eintrag	Menü-Untereintrag	Icon	Bemerkung
	Exit		Beenden/Stoppen der Process Designer Applikation.
Bearbeiten			
	Alles aus aktuellem Reiter entfernen		Löschen des gesamten Workflows, aller Elemente im Hauptarbeitsbereich.
Optionen			
	Lokale Konfiguration		Öffnet ein Pop-Up-Fenster, in dem Sie die Anzeigesprache des Process Designers auswählen können. Alle Sprachen, die für das System konfiguriert wurden (siehe Abschnitt *Allgemeine Konfiguration* im *ConSol *CM Administratorhandbuch*), sind verfügbar. Die Beschriftungen eines Workflows im Hauptbearbeitungsfenster werden in der ausgewählten Sprache angezeigt.
Anzeige			
	Normale Größe		Zeigt den Workflow in der Standardgröße an (so wie beim Start des Process Designers).
	Alle Bereiche aufklappen		Zeigt alle Bereiche (Scopes) aufgeklappt an.
	Alle Bereiche zuklappen		Zeigt alle Bereiche (Scopes) zugeklappt an.

Hauptmenü-Eintrag	Menü-Untereintrag	Icon	Bemerkung
	Palette ausblenden /einblenden		Blendet die Palette in der Oberfläche aus/ein.
	Eigenschaften ausblenden/einblenden		Blendet den Eigenschaften-Editor in der Oberfläche aus/ein.
	Explorer ausblenden /einblenden		Blendet den Explorer(-Baum) in der Oberfläche aus/ein.
	Zeige die Übertragungshistorie der Tickets an		Öffnet ein Pop-up-Fenster, in dem die Parameter für die Ticketübertragung während der Entwicklung des Workflows angezeigt werden: • **Workflow-Name** Name des Workflows. • **Version** Version des Workflows. • **Startzeit** Start der Übertragung, dies ist die Startzeit des Installieren-Befehls. • **Endezeit** Ende der Übertragung, nach dieser Zeit wird der Workflow voll funktionsfähig sein.

Hauptmenü-Eintrag	Menü-Untereintrag	Icon	Bemerkung
			• **Übertragene Tickets** Anzahl der Tickets, die übertragen wurden, d.h. welche während der Workflow-Entwicklung vom System angefasst wurden. Sollte identisch sein mit der Summe der offenen Tickets aus allen Queues, die diesen Workflow benutzen**.** • **Details** Zusätzliche Informationen bezüglich der Installation mit der Ticketübertragung.
Hilfe			
	About		Zeigt Versionsinformationen über den Process Designer und über die Java Virtual Machine, die in der aktuellen Konfiguration benutzt wird (dies ist die JVM des Browser-Plugins).

Workflow-Bearbeitungsbereich

Um einen Workflow zu designen, definieren Sie die Workflow-Elemente im grafischen Layout-Modus des Process Designers und fügen, wenn benötigt, Groovy-Skripte zu den Elementen hinzu.

Ein neues Element kann einem Workflow mittels Drag-and-Drop aus der Palette hinzugefügt werden.

Als Nachfolger eines existierenden Elements kann ein neues Element auch über das Kontextmenü (Rechtsklick) eines existierenden Elements erstellt werden, z.B. für eine Aktivität (siehe folgendes Bild). Das neue Element und eine Verbindung zu diesem Element werden erstellt.

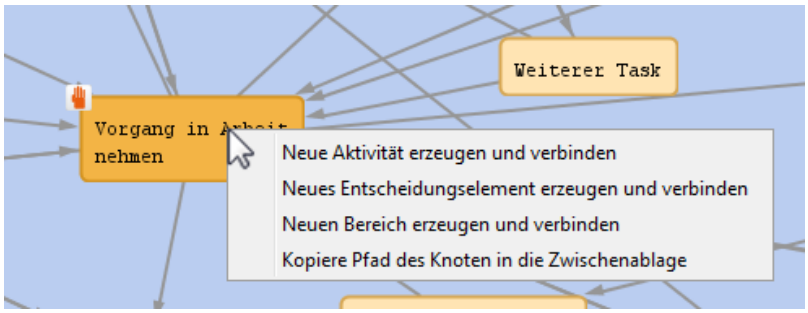


Fig. 2: ConSol*CM Process Designer - Kontextmenü für eine Workflow-Aktivität

Eine neue Verbindung zwischen Elementen wird erstellt, indem man mit gedrückter linker Maustaste und gleichzeitig gedrückter *STRG*-Taste eine Linie zwischen den Elementen zieht. Wenn die Verbindung von einem Scope zu einem anderen führt, werden ein Eintritts- und Austrittspunkt automatisch hinzugefügt.

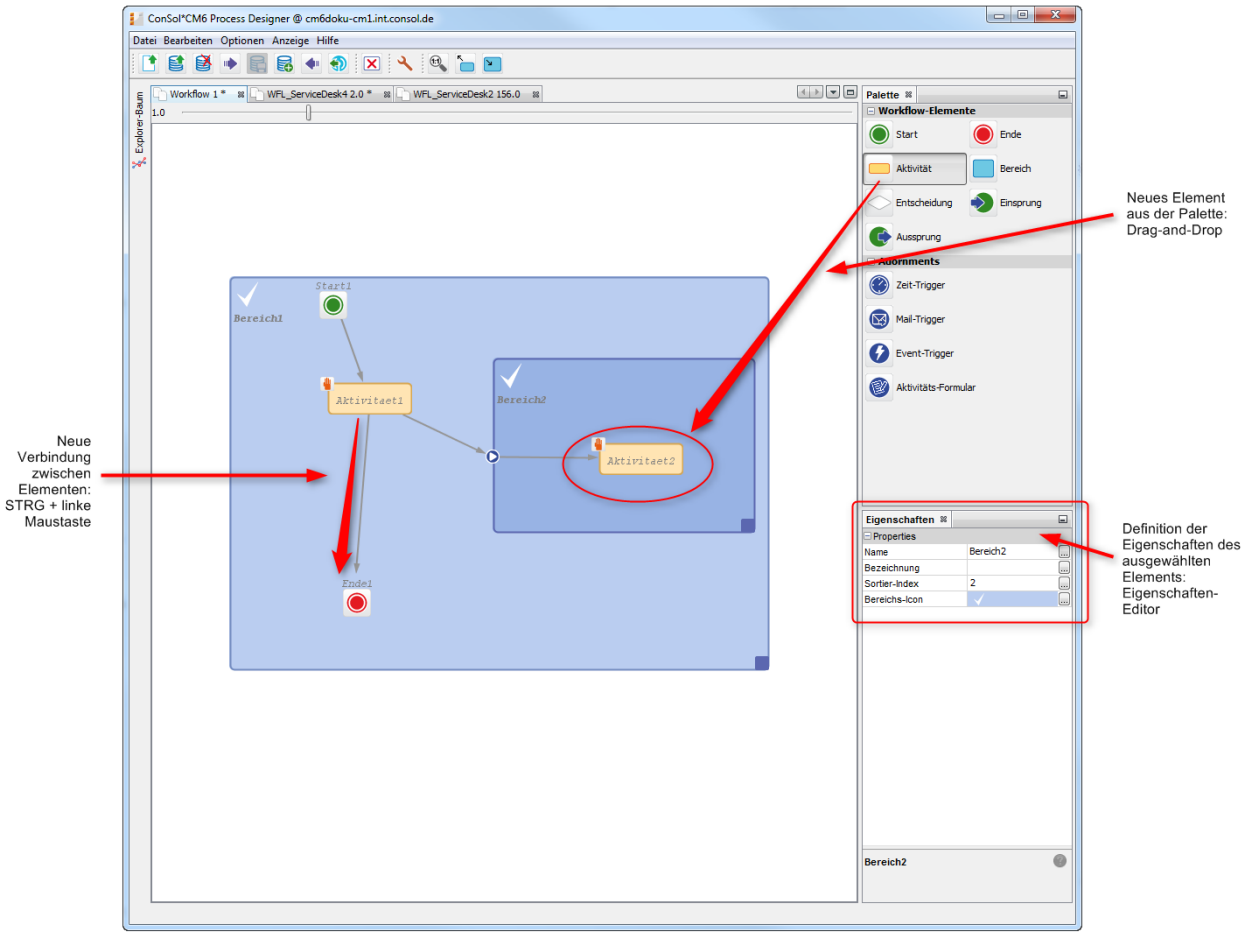


Fig. 3: ConSol*CM Process Designer - Hinzufügen neuer Elemente und Verbindungen

Es empfiehlt sich, für jeden Workflow einen globalen Scope zu erstellen. Bitte lesen Sie für mehr Informationen darüber, wie man gute Workflows erstellt, den Abschnitt Best Practices.

Laden und Löschen von Workflows

Laden eines Workflows

Nachdem Sie das Icon oder den Menü-Eintrag *Laden* ausgewählt haben, wird Ihnen eine Tabelle mit allen verfügbaren Workflows angezeigt.

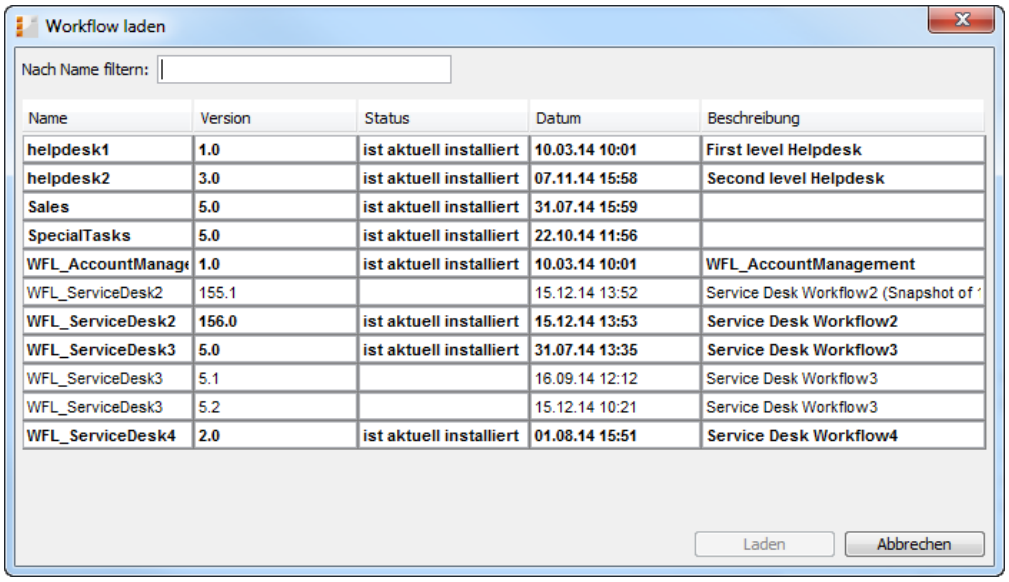


Fig. 4: ConSol*CM Process Designer - Laden eines Workflows

Die Tabelle kann nach einer ihrer Spalten sortiert werden, indem Sie auf das kleine Dreieck-Icon neben der Spaltenüberschrift klicken.

Die Tabelle enthält die folgenden Spalten:

- **Name**
  Der Name des Workflows, der unter *Name* in den Eigenschaften des Workflows angegeben wurde (klicken Sie auf den weißen Bereich, der sich um den globalen Scope herum befindet, um sich die Eigenschaften des Workflows anzeigen zu lassen).
- **Version**
  Die Version des Workflows. Diese wird vom ConSol*CM-System automatisch vergeben. Wenn ein Szenario exportiert und danach wieder importiert wurde, beginnt die Numerierung wieder mit 1.0.
- **Status**
  Bei älteren Workflows ist dieses Feld leer. Workflows, die aktuell installiert sind, werden mit *ist aktuell installiert* beschrieben.
- **Datum**
  Das Datum der letzten Modifikation (Datum, an dem der Workflow gespeichert wurde) wird angezeigt.
- **Beschreibung**
  Die Beschreibung, die im Feld *Workflow-Beschreibung* (**nicht** *Beschreibung*!) eingegeben wurde.

Um einen Workflow zu laden, wählen Sie ihn aus der Liste aus und klicken Sie auf *Laden*. Hierbei ist keine Mehrfachauswahl möglich.

Löschen eines Workflows

Nachdem Sie das Icon oder den Menü-Eintrag *Löschen* ausgewählt haben, erscheint eine Tabelle mit allen verfügbaren Workflows.

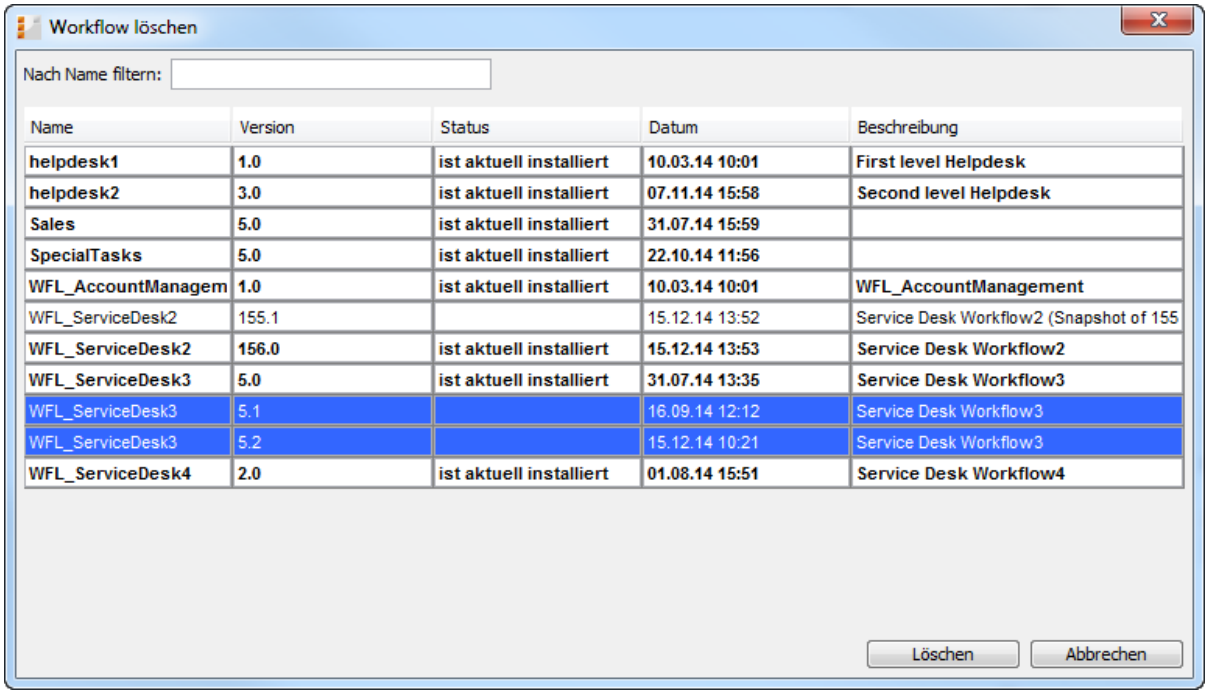


Fig. 5: ConSol*CM Process Designer - Löschen von Workflows

Die Tabelle kann nach einer ihrer Spalten sortiert werden, indem Sie auf das kleine Dreieck-Icon neben der Spaltenüberschrift klicken.

Die Tabelle enthält die folgenden Spalten:

- **Name**
  Der Name des Workflows, der unter *Name* in den Eigenschaften des Workflows angegeben wurde (klicken Sie auf den weißen Bereich, der sich um den globalen Scope herum befindet, um sich die Eigenschaften des Workflows anzeigen zu lassen).
- **Version**
  Die Version des Workflows. Diese wird vom ConSol*CM-System automatisch vergeben. Wenn ein Szenario exportiert und danach wieder importiert wurde, beginnt die Numerierung wieder mit 1.0.
- **Status**
  Bei älteren Workflows ist dieses Feld leer. Workflows, die aktuell installiert sind, werden mit *ist aktuell/ installiert* beschrieben.
- **Datum**
  Das Datum der letzten Modifikation (Datum, an dem der Workflow gespeichert wurde) wird angezeigt.
- **Beschreibung**
  Die Beschreibung, die im Feld *Workflow-Beschreibung* (**nicht** *Beschreibung*!) eingegeben wurde.

Um einen oder mehrere Workflow(s) zu löschen, wählen sie den oder die Workflows aus der Liste aus und klicken Sie auf *Löschen*. Sie erhalten für jeden Workflow eine Aufforderung, das Löschen dieses Workflows zu bestätigen. Wenn Sie also eine große Anzahl von Workflows zum Löschen ausgewählt haben und dann bemerken, dass Sie einen der Workflows doch nicht löschen möchten, können sie dies tun, ohne die gesamte Ausführung des Löschens abzubrechen.

Information:

Es empfiehlt sich, alle oder fast alle alten Workflows zu löschen, bevor Sie ein Szenario exportieren, da eine große Anzahl an Workflows die Größe eines Szenarios beträchtlich erhöht. Für den Export und Import von Szenarios lesen Sie bitte den Abschnitt *Deployment* im *ConSol*CM Administratorhandbuch.*

Palette für Elemente und Adornments

Die Standardeinstellung ist, dass die Palette in der oberen rechten Ecke angezeigt wird. Sie können die Palette über den Hauptmenü-Eintrag *Anzeige* mit *Palette ausblenden/einblenden* aus- und wieder einblenden.

Die Palette enthält zwei Typen von Workflow-Komponenten:

- Elemente
- Adornments

Elemente

Elemente sind die Basiskomponenten, welche den Workflow bilden und die Prozesslogik repräsentieren:

Icon	Element	Bemerkung	Abschnitt
	Startknoten	Wird automatisch gesetzt, es kann kein anderer Startknoten als der Standard-Startknoten hinzugefügt werden.	START-Knoten
	Endknoten	Ein Workflow kann einen oder mehrere Endknoten besitzen.	ENDE-Knoten
	Aktivität	Die Aktionen eines Workflows, manuell oder automatisch.	Aktivitäten
	Bereich (Scope)	Das höchste Hierarchie-Level eines Workflows.	Scopes (Bereiche)
	Entscheidung	Entscheidungsknoten mit einem *true*- und einem *false*-Ausgangspunkt.	Entscheidungsknoten
	Einsprung	Einsprungpunkt für Tickets aus anderen Workflows/Queues.	Aussprung- und Einsprungknoten
	Aussprung	Aussprungpunkt für Tickets. Es muss eine Ziel-Queue festgelegt werden. Ein Zielknoten kann festgelegt werden, ist aber optional.	Aussprung- und Einsprungknoten

Adornments

Adornments sind Objekte, die einer Workflow-Aktivität oder einem Scope zugewiesen werden. Bitte lesen Sie für eine detaillierte Erklärung die angegebenen Abschnitte.

Icon	Adornment	Bemerkung	Abschnitt
	Zeit-Trigger	Kann Zeitintervalle messen. Feuert, wenn das Ende des Intervalls erreicht wurde. Kann optional auch einen Arbeitszeitkalender verwenden.	Zeit-Trigger
	Mail-Trigger	Feuert, wenn ein E-Mail für das Ticket eingeht.	Mail-Trigger
	Business-Event-Trigger	Feuert, wenn ein Ereignis geschieht. Der Typ des Ereignisses kann angegeben werden (z.B. Wechsel des Bearbeiters, Wechsel der Priorität).	Business-Event-Trigger
	Aktivitätsformular (Activity Control Form, ACF)	Legt das ACF fest, das angezeigt werden soll, wenn die Aktivität ausgeführt wird. ACFs werden im Admin-Tool definiert.	Aktivitätsformulare (ACFs)

Der Eigenschaften-Editor (Beispiel: Aktivität)

Der Eigenschaften-Editor ist für das Element geöffnet, das im Hauptarbeitsbereich ausgewählt wurde, und beinhaltet die für diese Komponente spezifischen Parameter. Einige allgemeine Parameter sind für alle Komponenten sichtbar, manche sind nur für einen bestimmten Komponententyp vorhanden.

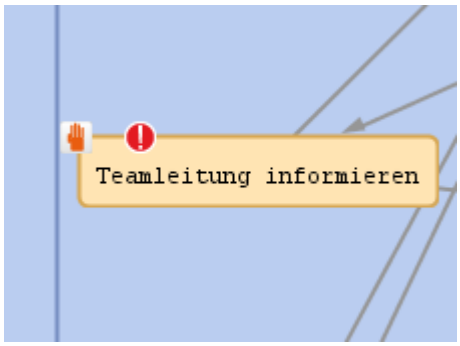


Fig. 6: ConSol*CM Process Designer - Ausgewählte Aktivität im Workflow

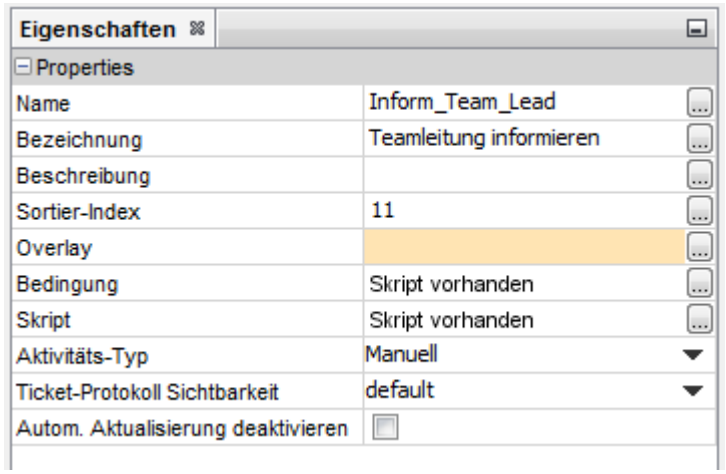


Fig. 7: ConSol*CM Process Designer - Eigenschaften-Editor

Eigenschaften:

- **Name**
  Notwendiger, technischer Name des Objekts. Wenn ein Objekt neu erstellt wird, können Sie seine Bezeichnung editieren und der Objekt-Name wird automatisch aus der Bezeichnung erstellt (Umlaute werden ausgelassen). Danach wird der Name des Objekts nicht mehr automatisch geändert, kann aber noch manuell editiert werden. Erlaubte Zeichen für Namen sind:
  - Buchstaben (Klein- und Großbuchstaben), aber keine Umlaute
  - Unterstriche
  - Zahlen
- **Bezeichnung**
  Lokalisierter Name, der im Web Client angezeigt wird. Alle Sprachen, die für das System konfiguriert worden sind, sind verfügbar und können ausgefüllt werden. Im Web Client wird die Bezeichnung in der Sprache angezeigt, die im Webbrowser des Bearbeiters eingestellt ist. Wenn keine Bezeichnung in dieser Sprache verfügbar ist, wird die Bezeichnung in der Standardsprache angezeigt.

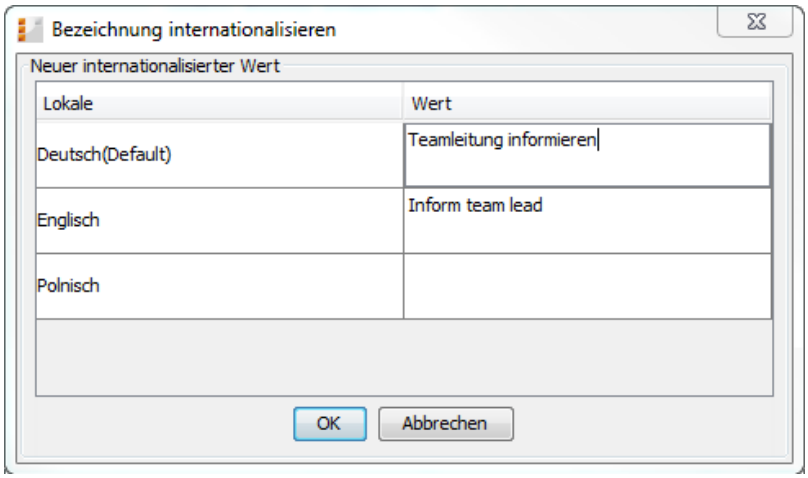


Fig. 8: ConSol*CM Process Designer - Lokalisierung für Bezeichnungen

- **Beschreibung**
  Optional. Es kann ein lokalisierter Text eingegeben werden, der als Mouseover im Web Client angezeigt wird. Dies kann dem Bearbeiter Informationen dazu liefern, was passieren wird, wenn die betreffende Workflow-Aktivität ausgeführt wird.

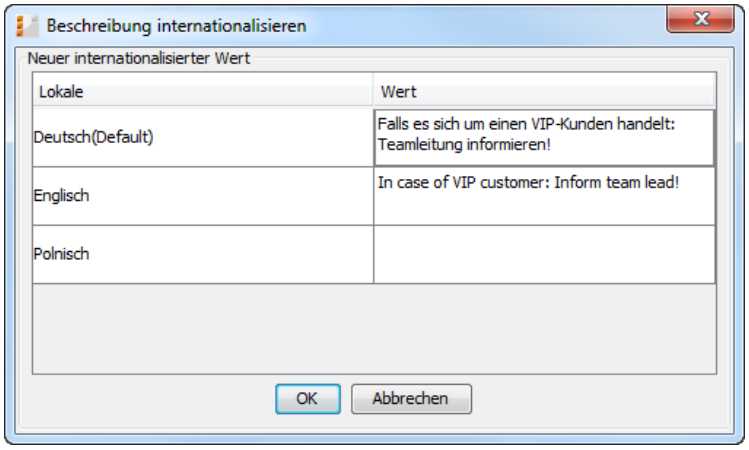


Fig. 9: ConSol*CM Process Designer - Lokalisierte Beschreibung einer Aktivität

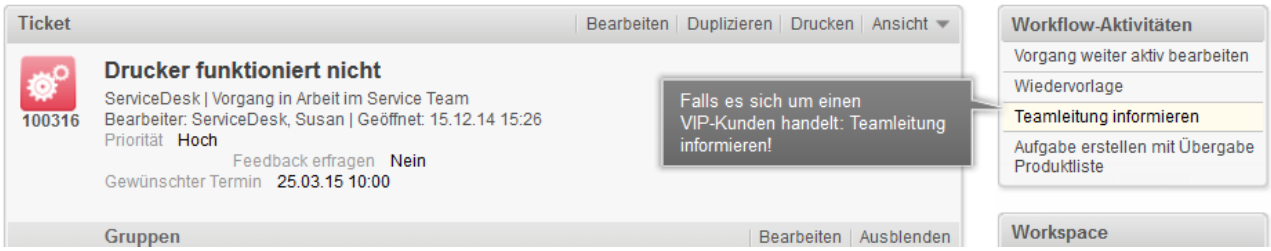


Fig. 10: ConSol*CM/Web Client - Lokalisierte Beschreibung einer Aktivität als Mouseover

- **Sortier-Index**
  Definiert:
  - **Für Aktivitäten:**
    Die Reihenfolge der Aktivitäten in der Liste der Workflow-Aktivitäten im Web Client. Je höher die Nummer ist, desto weiter unten steht die Aktivität in der Liste im Web Client.
  - **Für Scopes:**
    Die Reihenfolge der Tickets in der Ticketliste (Web Client) in Sichten. Je höher der Scope-Sortier-Index ist, desto weiter oben in der Ticketliste werden die Tickets angezeigt. Dies ist z.B. wichtig für die Sortierung der Ticketliste nach Scopes.
- **Overlay**
  Optional, für Aktivitäten. Klicken Sie in den orangen Bereich, um eines der Standard-ConSol*CM-Overlays oder ein bereits hochgeladenes Icon zu verwenden. Klicken Sie auf den Dateibrowser-Button (...), um ein neues Icon aus Ihrem Dateisystem hochzuladen. Wenn das Ticket durch die Aktivität läuft, wird im Web Client das gewählte Overlay zum Ticket-Icon hinzugefügt. Es können maximal drei Overlays gleichzeitig mit einem Ticket-Icon verbunden sein. Dieser Mechanismus kann für verschiedene Zwecke genutzt werden, einige Beispiele dafür sind:
  - **Eine Eskalation:**
    Das Ticket wurde geöffnet, ohne dass sich ein Bearbeiter um das Ticket kümmert.
  - **Eine E-Mail:**
    Das Ticket hat eine E-Mail erhalten.
  - **Eine Nachricht für den Bearbeiter:**
    Ein anderer Bearbeiter hat z.B. einen Kommentar zu *meinem* Ticket hinzugefügt.

Fig. 11: ConSol*CM Process Designer - Eigenschaften-Editor: Standard-Overlays und ein kundenspezifisches Overlay

Fig. 12: ConSol*CM/Web Client - Icons mit Overlays

- **Overlay-Gültigkeit**
  Wird nur angezeigt, wenn ein Overlay ausgewählt wurde.
  - **Aktivität**
    Das Overlay bleibt so lange am Ticket, wie das Ticket hinter der Aktivität steht (die Aktivität also ausgeführt wurde). Sobald die nächste Aktivität ausgeführt wird, wird das Overlay vom Ticket entfernt.
  - **Bereich**
    Das Overlay wird entfernt, wenn das Ticket den Scope verlässt.
  - **Prozess**
    Das einmal zum Ticket-Icon hinzugefügte Overlay bleibt für den Rest des Prozesses am Ticket.
  - **Nächstes Overlay**
    Das Overlay bleibt so lange am Ticket, wie kein neues Overlay hinzugefügt wird. Sobald ein neues Overlay hinzugefügt wird, wird das alte entfernt.
- **Bedingung**
  Optional, für Aktivitäten. Ein Skript, welches *true* oder *false* zurückgeben muss, kann mit dem Skript-Editor (siehe Abschnitt Der Skript-Editor) eingegeben werden. Das Skript wird ausgeführt, wenn die vorhergehende Aktivität ausgeführt wurde, d.h. wenn es möglich wird, die Aktivität mit der Bedingung anzuzeigen. Wenn *true* zurückgegeben wird, wird die Aktivität angezeigt, wenn *false* zurückgegeben wird, wird die Aktivität nicht angezeigt. Wenn eine Bedingung für eine Aktivität definiert wurde, wird die Aktivität mit dem *Ausrufezeichen/Bedingungs*-Icon markiert.
- **Skript**
  Optional, für Aktivitäten. Es kann ein Skript mit dem Skript-Editor (siehe Abschnitt Der Skript-Editor) definiert werden, das ausgeführt wird, wenn das Ticket die Aktivität durchläuft.
- **Aktivitäts-Typ**
  Notwendig, für Aktivitäten. Es muss entweder *automatisch* oder *manuell* ausgewählt werden. Eine manuelle Aktivität wird im Web Client angezeigt und muss explizit von einem Bearbeiter ausgewählt /ausgeführt werden. Im Process Designer ist die Aktivität mit dem *Hand/Manuell*-Icon gekennzeichnet. Eine automatische Aktivität wird ohne die Interaktion mit einem Bearbeiter ausgeführt. Für eine detaillierte Erklärung der Prozesslogik von ConSol*CM lesen Sie bitte den Abschnitt Prozesslogik.

- **Ticket-Protokoll Sichtbarkeit**
  Notwendig, der Standardwert ist aber immer bereits gesetzt (*default*). Der Wert definiert, auf welchem Sichtbarkeitslevel des Web Clients die Aktion (dass die Aktivität ausgeführt wurde) angezeigt werden soll:
  - *2nd level and 3rd level*
  - *only 3rd level*
  - *on every level*
  - *default*
    Dies bezieht sich auf den Wert, der im Admin-Tool unter *Ticketprotokoll* für die Aktivitäten konfiguriert ist. Abhängig von dem Typ der Aktivität wird einer der folgenden Parameter verwendet:
    - Manuelle Aktivität/Aktivität mit Overlay ausgeführt
    - Aktivität nach Eskalation ausgeführt
    - Automatische Aktivität ausgeführt

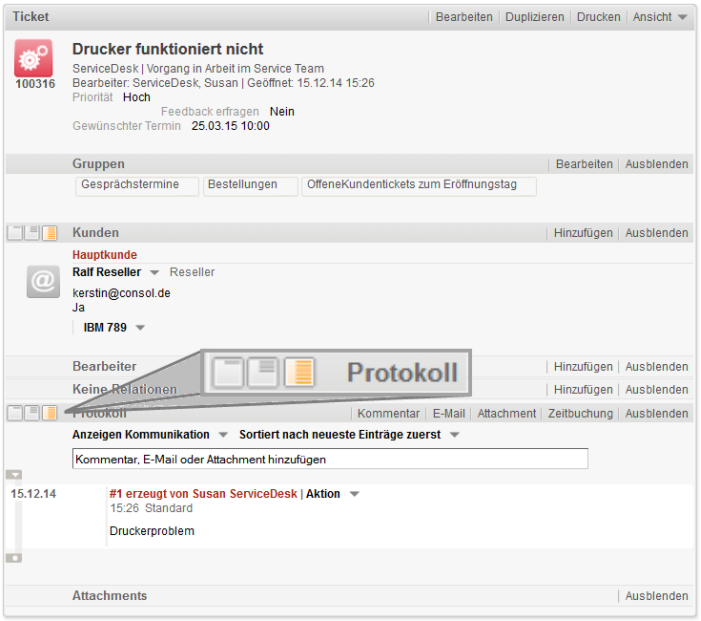


Fig. 13: ConSol*CM/Web Client - Sichtbarkeitslevel im Ticketprotokoll

- **Autom. Aktualisierung deaktivieren**
  Legt das Verhalten des Tickets fest, wenn ein Event gefeuert oder ausgeführt wurde. Normalerweise wird nach einem Ereignis eine Ticket-Update-Operation automatisch ausgeführt. Falls eine Kette von Ereignissen verwendet wird, sollten Sie vermeiden, nach jedem einzelnen Ereignis eine Ticket-Update-Operation anzustoßen. Um dies zu vermeiden, setzen Sie für alle Ereignisse außer dem letzten Ereignis der Kette das Feld *Autom. Aktualisierung deaktivieren* auf *true*. Dann wird das Ticket nur noch nach dem letzten Ereignis aktualisiert. Siehe dazu den Abschnitt Best Practices, Ticket-Update-Events.

3.2.2 Der Skript-Editor

Sie verwenden den Skript-Editor im Process Designer, um Groovy-Skripte zu schreiben (d.h. reiner Groovy- und Java-Code wird akzeptiert). Für Erklärungen, Empfehlungen und Beispiele bezüglich der Workflow-Programmierung mittels Skripte lesen Sie bitte den Abschnitt Workflow-Programmierung.

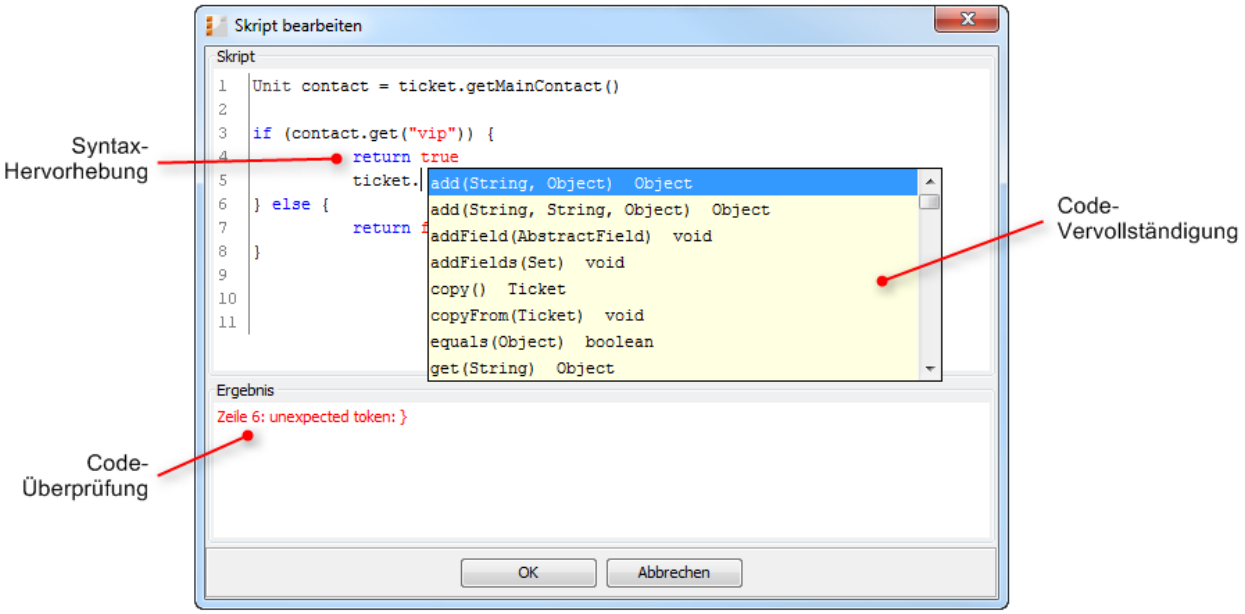


Fig. 14: ConSol*CM Process Designer - Skript-Editor

Der Skript-Editor stellt die folgenden Features bereit:

- **Syntax-Hervorhebung**
  Groovy-Code wird Keywords entsprechend hervorgehoben.
- **Code-Vervollständigung**
  Wenn Sie den Namen eines Objekts und den Punkt eingegeben haben, werden Ihnen mögliche Methoden vorgeschlagen. Drücken Sie *STRG + SPACE*, um die Code-Vervollständigung zu aktivieren.
- **Code-Überprüfung**
  Der eingegebene Code wird auf den korrekten Gebrauch von allgemeiner Syntax und Methoden hin kontrolliert. Der fehlerhafte Code wird im Bereich *Ergebnis* angezeigt.

4 ConSol CM Process Designer Handbuch - Komponenten von ConSol CM Workflows

4.1 Komponenten von ConSol*CM-Workflows

4.1.1 Einleitung

Sie können mit verschiedenen Typen von Workflow-Komponenten arbeiten, um Workflows für Ihr ConSol*CM-System zu erstellen. Die Palette innerhalb des Process Designers enthält alle Elemente und Adornments, für einen Überblick darüber siehe Abschnitt Palette für Elemente und Adornments.

In den folgenden Kapiteln werden alle Workflow-Elemente und Adornments im Detail beschrieben:

Workflow-Komponente	Erklärung
START-Knoten	Der erste Knoten in einem Workflow, siehe Abschnitt START-Knoten.
ENDE-Knoten	Ein oder mehrere Endknoten eines Prozesses. Das Ticket wird geschlossen. Siehe Abschnitt END E-Knoten.
Scopes (Bereiche)	Bereiche eines Prozesses, siehe Abschnitt Scopes (Bereiche).
Aktivitäten	Die Schritte eines Prozesses. Können automatisch oder manuell sein, siehe Abschnitt Aktivitäten.
Entscheidungsknoten	Workflow-Element, welches eine *true/false*-Entscheidung repräsentiert, siehe Abschnitt Entscheidungsknoten.
Adornments	Elemente, um den Prozessfluss zu steuern: Trigger und Aktivitätsformulare (Activity Control Forms = ACF). Siehe Abschnitt Adornments (Trigger und ACFs).
Einsprung- und Aussprungknoten	Elemente, die Workflows untereinander verbinden, siehe Abschnitt Aussprung- und Einsprungknoten.

4.2 Workflow-Komponenten: START-Knoten

- Einleitung
- Eigenschaften eines Startknotens

4.2.1 Einleitung

Jeder Workflow enthält genau einen START-Knoten. Wenn Sie einen neuen Workflow erstellen, wird der Startknoten automatisch hinzugefügt. Sie müssen ihn also nicht selbst hinzufügen.

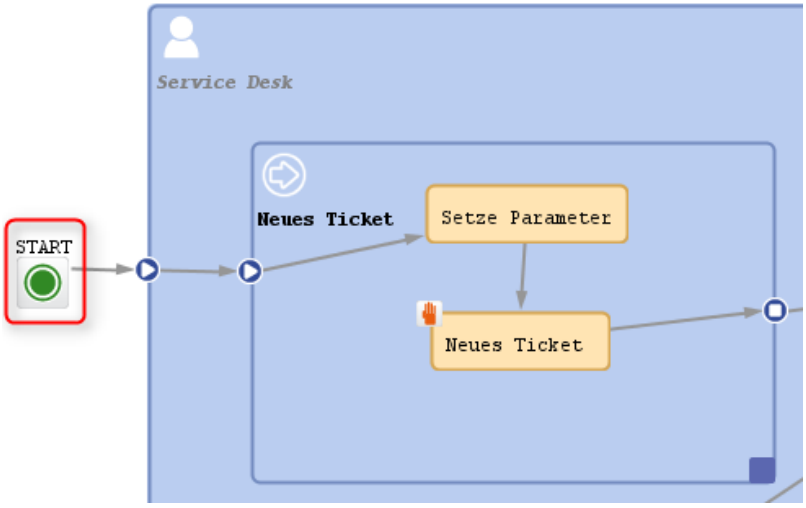


Fig. 1: ConSol*CM Process Designer - Startknoten

Der Startknoten besitzt kein Skript und kann nicht konfiguriert werden.

Wenn ein Ticket in den Workflow eintritt und kein spezieller Einsprungspunkt definiert wurde, durchläuft das Ticket den Startknoten.

Best Practices:

Der Startknoten sollte sich nicht innerhalb des globalen Scopes befinden. Siehe dazu auch Best Practices.

4.2.2 Eigenschaften eines Startknotens

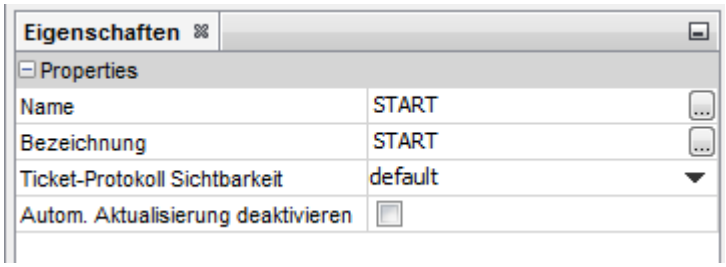


Fig. 2: ConSol*CM Process Designer - Eigenschaften eines Startknotens

Eigenschaften:

- **Name**
  Technischer Name des Objekts.
- **Bezeichnung**
  Lokalisierter Name, der im Web Client angezeigt wird.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll-Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Automatische Aktualisierung deaktivieren.

4.3 Workflow-Komponenten: ENDE-Knoten

- Einleitung
- Eigenschaften eines Endknotens

4.3.1 Einleitung

Ein Workflow in ConSol*CM kann einen oder mehrere ENDE-Knoten besitzen.

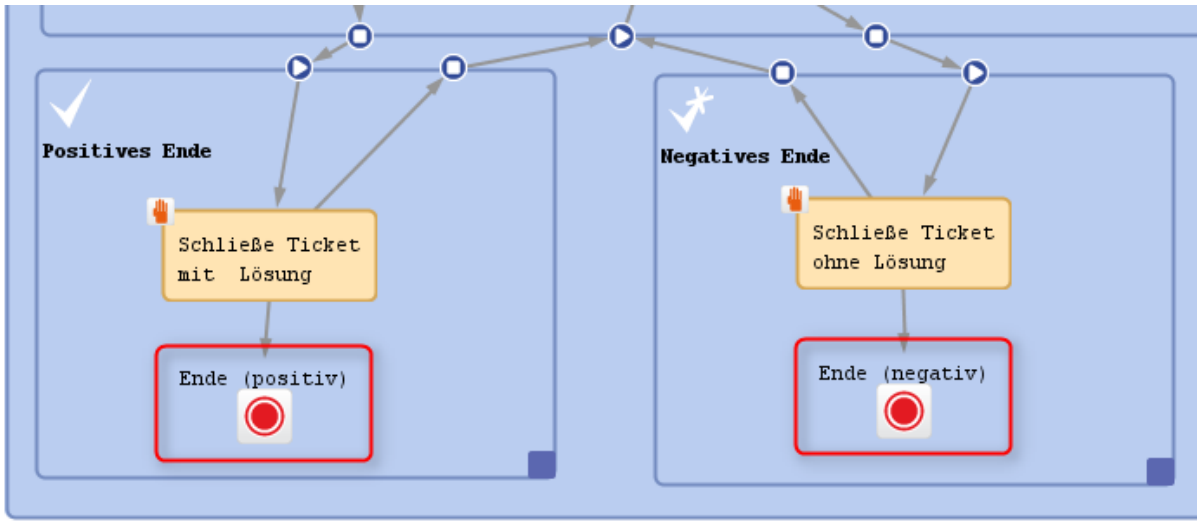


Fig. 1: ConSol*CM Process Designer - Endknoten

Ein Endknoten repräsentiert das Schließen eines Tickets, d.h. wenn ein Ticket in einen Endknoten wandert, ist es im technischen Sinne geschlossen. Kein Bearbeiter kann das Ticket dann mehr bearbeiten. Das Ticket kann durch den Administrator mittels der *Ticket-Verwaltung* im Admin-Tool wiedereröffnet werden, bitte lesen Sie für detaillierte Informationen den entsprechenden Abschnitt im *ConSol*CM Administratorhandbuch.*

Trotzdem können Bearbeiter, wenn sie die benötigten Berechtigungen besitzen, das Ticket noch lesen. Dies ist eine wichtige Grundlage für die Nutzung von allen Tickets eines ConSol*CM-Systems als Wissensdatenbank.

Ein Ticket kann durch eine manuelle oder eine automatische Aktion in einen Endkonten wandern. Im oberen Bild sind die Endknoten automatische Endknoten, d.h. das Ticket wandert in diesen Knoten, wenn die vorhergehende Aktivität vollzogen wurde.

Ein Workflow muss mindestens einen Endknoten besitzen, da es einen Weg geben muss, um ein Ticket zu schließen.

Sie können auch mehr als einen Endknoten erstellen. Dies kann nützlich sein, wenn Sie Reports erstellen, z. B. um zwischen einem positiven und negativen Ende unterscheiden zu können.

Ein Endknoten kann ein Skript besitzen, d.h. bevor das Ticket geschlossen wird, kann ein Skript ausgeführt werden.

Best Practices:

Manchmal kann es notwendig sein, ein Ticket aus der Sicht eines Bearbeiters auf *geschlossen*, *erledigt* oder *fertig* zu setzen, d.h. das Ticket auf ein vorläufiges ENDE zu setzen. Nach einer Weile, wenn es keine Fragen oder Anmerkungen mehr vom Kunden gibt, soll das Ticket automatisch geschlossen werden. Dies können Sie erreichen, indem Sie einen Zeit-Trigger an eine Beenden-Aktivität setzen und das Ticket automatisch nach der festgelegten Zeit in den Endknoten wandern lassen (siehe folgendes Bild).

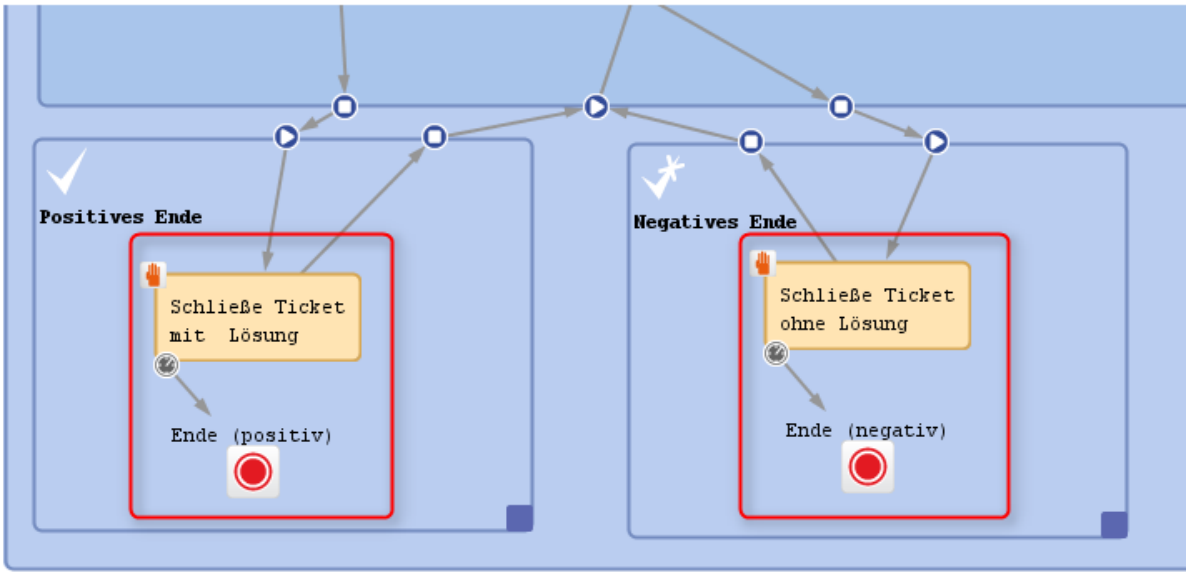


Fig. 2: ConSol*CM Process Designer - Mittels Zeit-Triggern erreichte Endknoten

4.3.2 Eigenschaften eines Endknotens

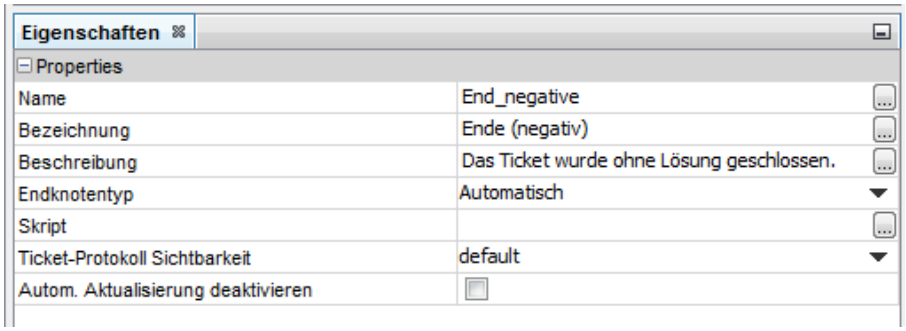


Fig. 3: ConSol*CM Process Designer - Eigenschaften von Endknoten

Eigenschaften:

- **Name**
  Technischer Name des Objekts.
- **Bezeichnung**
  Lokalisierter Name, der im Web Client angezeigt wird.
- **Beschreibung**
  Beschreibung, die als Mouseover-Text angezeigt wird.
- **Endknotentyp**
  Automatisch/Manuell.
- **Skript**
  Hier kann ein Skript eingegeben werden, das ausgeführt werden soll, wenn das Ticket in den Endknoten eintritt, d.h. bevor das Ticket geschlossen wird.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll-Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Automatische Aktualisierung deaktivieren.

4.4 Scopes (Bereiche)

- Einleitung zu Scopes (Bereichen)
- Definieren eines neuen Scopes
- Eigenschaften eines Scopes
- Scopes und Sichten

4.4.1 Einleitung zu Scopes (Bereichen)

Wenn ein Ticket durch einen Prozess läuft, gibt es verschiedene Positionen, die es durchlaufen muss, alle in einer vorher festgelegten Reihenfolge. Zum Beispiel kann das Ticket in einer Servicedesk-Umgebung als *neues Ticket* hereinkommen, dann muss es vorqualifiziert werden (in diesem Beispiel: Gibt es SLAs, die berücksichtigt werden müsen, handelt es sich um einen VIP-Kunden?). Anschließend kann der Bearbeiter an dem Ticket arbeiten und es möglicherweise für eine Zeit auf Wiedervorlage legen. Danach soll das Ticket geschlossen werden, entweder als *positiv, mit Lösung* oder *negativ, ohne Lösung*. Diese Hauptschritte des Prozesses werden in ConSol*CM-Workflows durch Scopes repräsentiert. Das folgende Bild stellt dies in einem Beispiel-Workflow dar. Die Scopes (blaue Bereiche) sind hier z.B. *Service Desk*, *Neues Ticket* und *Vorgang in Arbeit im Service Team*.

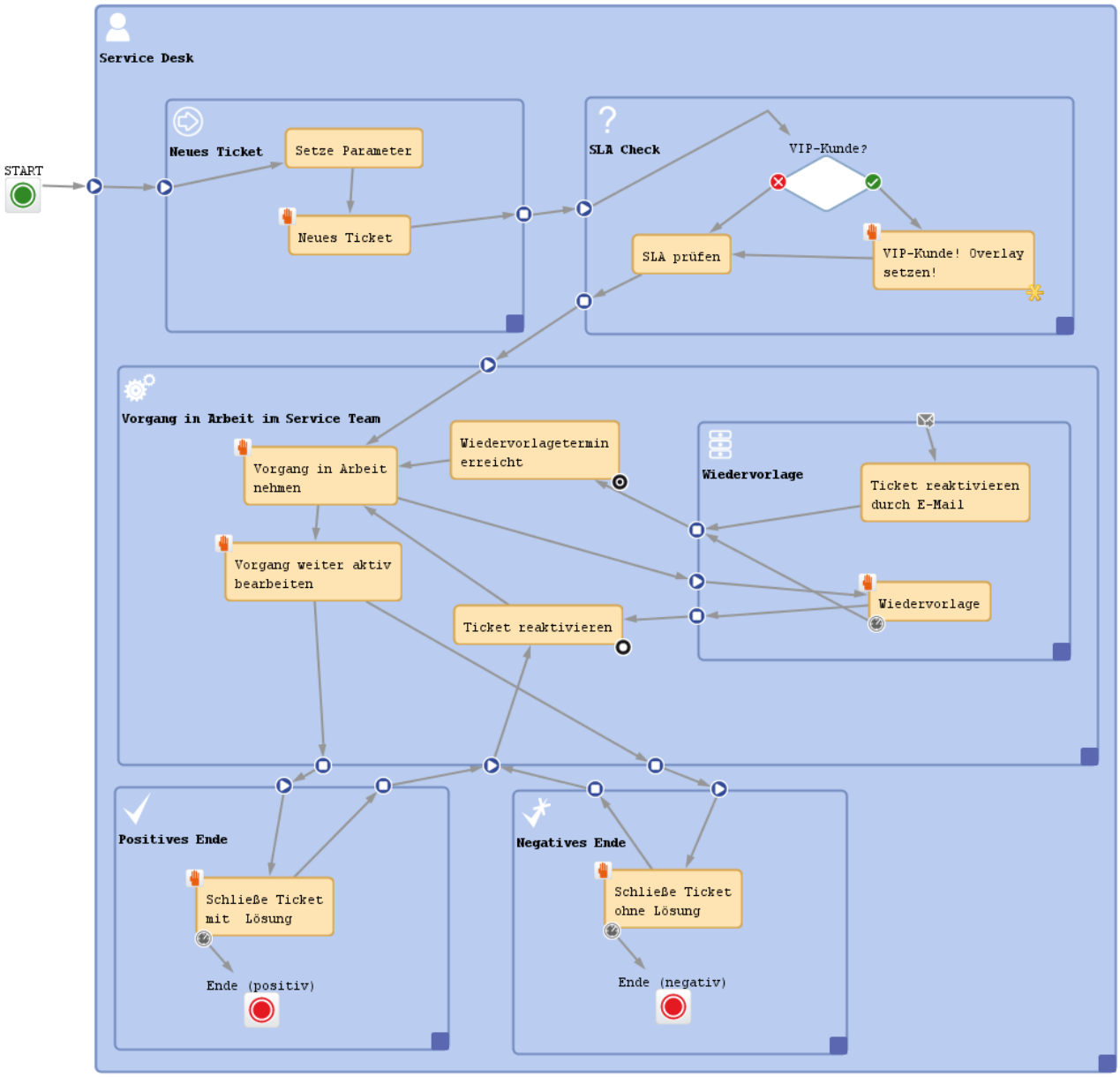


Fig. 1: ConSol*CM Process Designer - Workflow mit Scopes

Innerhalb jedes Prozessschrittes kann es eine oder mehrere Aktivitäten geben, z.B. wird während der Vorqualifikation zuerst der VIP-Status des Kunden überprüft, danach wird das SLA kontrolliert. Diese Aktivitäten werden im Abschnitt Aktivitäten detailliert beschrieben. An dieser Stelle werden nur die Scopes erklärt.

Ein Scope kann Teil eines anderen Scopes sein oder - von der anderen Seite betrachtet - ein Scope kann Sub-Scopes enthalten.

Ein Scope kann verschiedene Typen von Triggern besitzen, z.B. feuert ein Mail-Trigger immer dann, wenn eine E-Mail an ein Ticket, das sich gerade in diesem Scope befindet, empfangen wurde. Bitte lesen Sie für Details die Abschnitte Mail-Trigger, Zeit-Trigger und Business-Event-Trigger.

4.4.2 Definieren eines neuen Scopes

Um einen neuen Scope zu definieren, d.h. einen neuen Scope zu einem Workflow hinzuzufügen, klicken Sie auf das Scope-Icon in der Palette und ziehen Sie es per Drag-and-Drop in den Workflow an die gewünschte Stelle. Aktivieren Sie ihn mit einem Doppelklick. Danach können Sie neue Aktivitäten oder Elemente hinzufügen oder bereits existierende Aktivitäten/Elemente in den Scope ziehen. Wenn Sie die Elemente durch das Ziehen von Pfeilen miteinander verbinden, werden die Austritts- und Eintrittspunkte eines Scopes automatisch definiert.

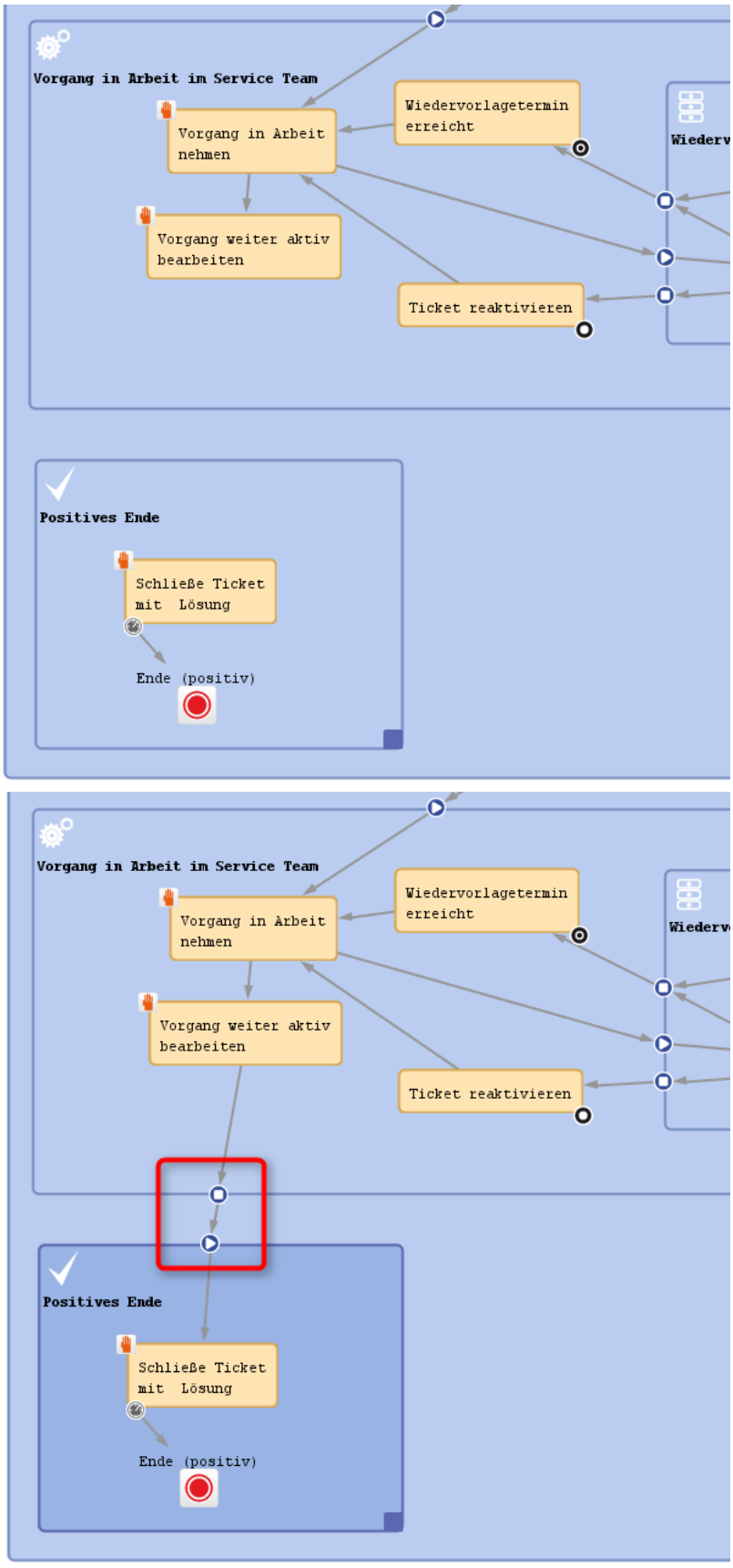


Fig. 2: ConSol*CM Process Designer - Automatisch generierte Austritts- und Eintrittspunkte an Scopes

Wenn Sie den neuen Scope definiert/hinzugefügt haben, können Sie die Eigenschaften des Scopes festlegen, siehe dazu den nächsten Abschnitt.

4.4.3 Eigenschaften eines Scopes

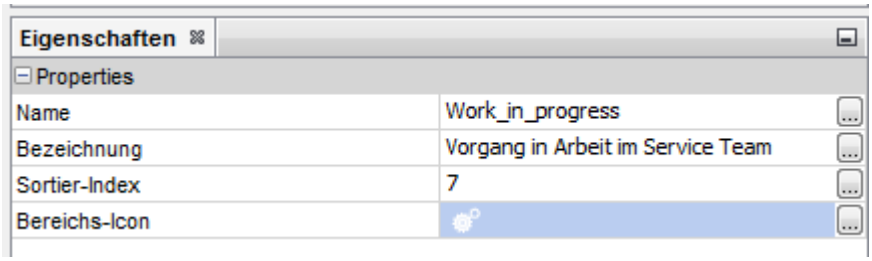


Fig. 3: ConSol*CM Process Designer - Eigenschaften eines Scopes

Die folgenden Eigenschaften können für einen Scope definiert werden:

- **Name**
  Der technische Name des Objekts.
- **Bezeichnung**
  Lokalisierter Name, der im Web Client angezeigt wird.
- **Sortier-Index**
  Definiert die Position der Tickets dieses Scopes innerhalb einer Sicht in der Ticketliste (für den Fall, dass eine Sicht mehr als einen Scope beinhaltet)**.**
- **Bereichs-Icon**
  Das Icon, welches als Scope-Icon im Web Client angezeigt wird (siehe folgendes Bild). Klicken Sie in den blauen Bereich, um eines der ConSol*CM-Standard-Icons auszuwählen, oder benutzen Sie den Dateibrowser (...), um ein Icon aus Ihrem Dateisystem zu laden.

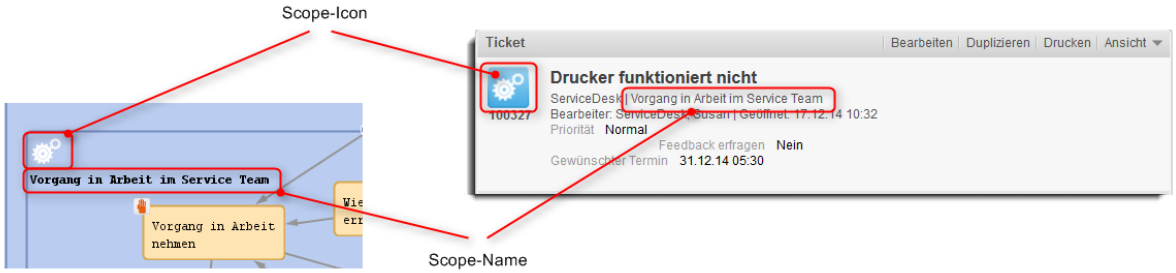


Fig. 4: ConSol*CM/Web Client - Scope-Icon

> **Vorsicht:**
>
> Bitte beachten Sie, dass das Icon mit der Farbe des Ticket-Icons verbunden wird. Falls Sie also Ihre eigenen Ticket-Icons hochladen möchten, sollten Sie transparente Bilder für die Ticket-Icons benutzen. Andernfalls kann die Hintergrundfarbe verloren gehen oder nur als kleiner Rand um das Icon herum sichtbar sein.

4.4.4 Scopes und Sichten

Sichten, d.h. die Auswahlkriterien für die Ticketliste(n), werden auf der Basis von Scopes definiert. Für eine detaillierte Erklärung von Sichten und der Definition von Sichten lesen Sie bitte den entsprechenden Abschnitt im *ConSol*CM Administratorhandbuch.*

Im aktuellen Kontext, d.h. wenn Sie die Scopes innerhalb eines Workflows definieren, ist wichtig zu beachten, welche Sichten später benötigt werden. Zum Beispiel basiert der Mechanismus von *neuen*, *aktiven* und *inaktiven* Tickets vollständig auf der Definition von Scopes und Sichten:

- **Sicht: Neu**
  Alle Tickets, die sich im Scope *Neu* befinden.
- **Sicht: Aktiv**
  Alle aktiven Tickets, d.h. Tickets, **die sich nicht** in einem Scope wie *Auf Wiedervorlage*, *Wartend* o. ä. befinden.
- **Sicht: Inaktiv**
  Alle Tickets, **die sich** in einem Scope wie *Auf Wiedervorlage*, *Wartend* o.ä. befinden.

Dies bedeutet, dass, immer wenn eine Sicht benötigt wird, um nur eine bestimmte Art von Tickets anzuzeigen, dafür ein Scope definiert werden muss.

Vorsicht:

Wir empfehlen dringend, **keine** Sichten zu definieren, die geschlossene Tickets enthalten.

Die Anzahl der geschlossenen Tickets wird im Laufe der Arbeit mit der Applikation erheblich anwachsen. Aus diesem Grund würde die Sicht mit den geschlossenen Tickets schnell immer die maximale Anzahl erreichen, die für eine Sicht erlaubt ist (die Anzahl kann mittels einer System-Property festgelegt werden). Dies kann negativen Einfluss auf die Performance des Web Clients haben und in den meisten Fällen befinden sich die gewünschten Tickets nicht zwischen den ersten 50 oder 100 Tickets.

Fazit: Eine Sicht für geschlossene Tickets ist nicht hilfreich und kann die Geschwindigkeit des Systems für den Bearbeiter heruntersetzen. Eine Sicht für geschlossene Tickets kann nur innerhalb von Testumgebungen möglicherweise eine sinnvolle Option sein.

4.5 Workflow-Komponenten: Aktivitäten

- Einleitung zu Aktivitäten
- Eigenschaften einer Aktivität
- Prozesslogik von Aktivitäten
- Beispiele für Aktivitäten
  - Beispiel 1: Bedingung für die Anzeige der Aktivität "Teamleitung informieren"
  - Beispiel 2: Sende eine E-Mail an den Hauptkunden, wenn das Ticket geöffnet wurde
  - Beispiel 3: Weise das Ticket dem aktuellen Bearbeiter zu

4.5.1 Einleitung zu Aktivitäten

Eine Aktivität repräsentiert eine Aktion in einem Workflow. Eine Aktivität befindet sich innerhalb eines Scopes und ist von einem der folgenden Typen:

- manuell
- automatisch

Eine **manuelle** Aktivität muss durch eine manuelle Aktion eines Bearbeiters im Web Client erfolgen. Die Aktivität wird als Workflow-Aktivität im Web Client angezeigt (vorausgesetzt, mindestens eine der Rollen des Bearbeiters besitzt die *Ausführen*-Berechtigung, bitte lesen Sie für Details dazu da*s ConSol*CM Administratorhandbuch*, Abschnitt *Rollenverwaltung*). Im Process Designer ist die Aktivität mit dem *Hand /Manuell*-Icon  markiert.

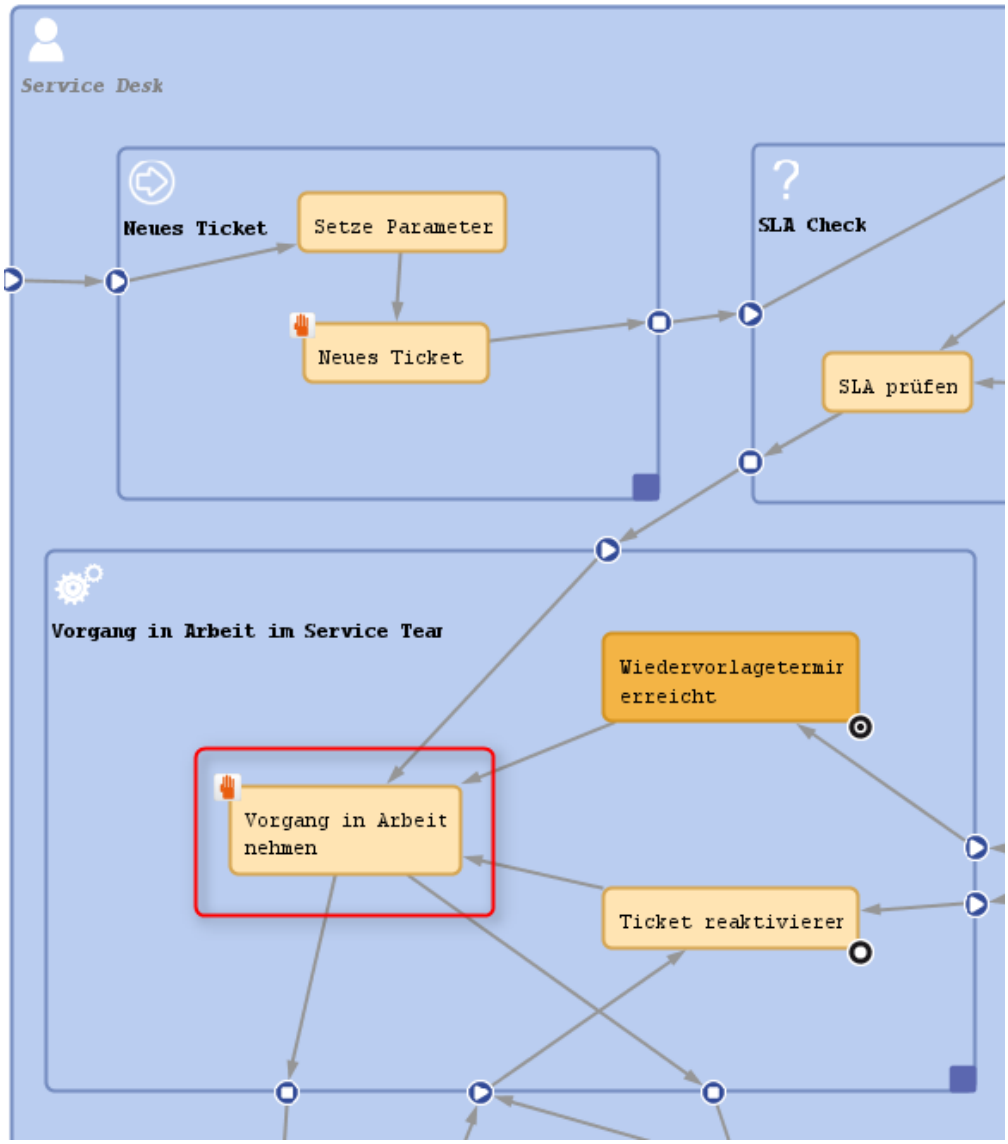


Fig. 1: ConSol*CM Process Designer - Manuelle Aktivität im Workflow

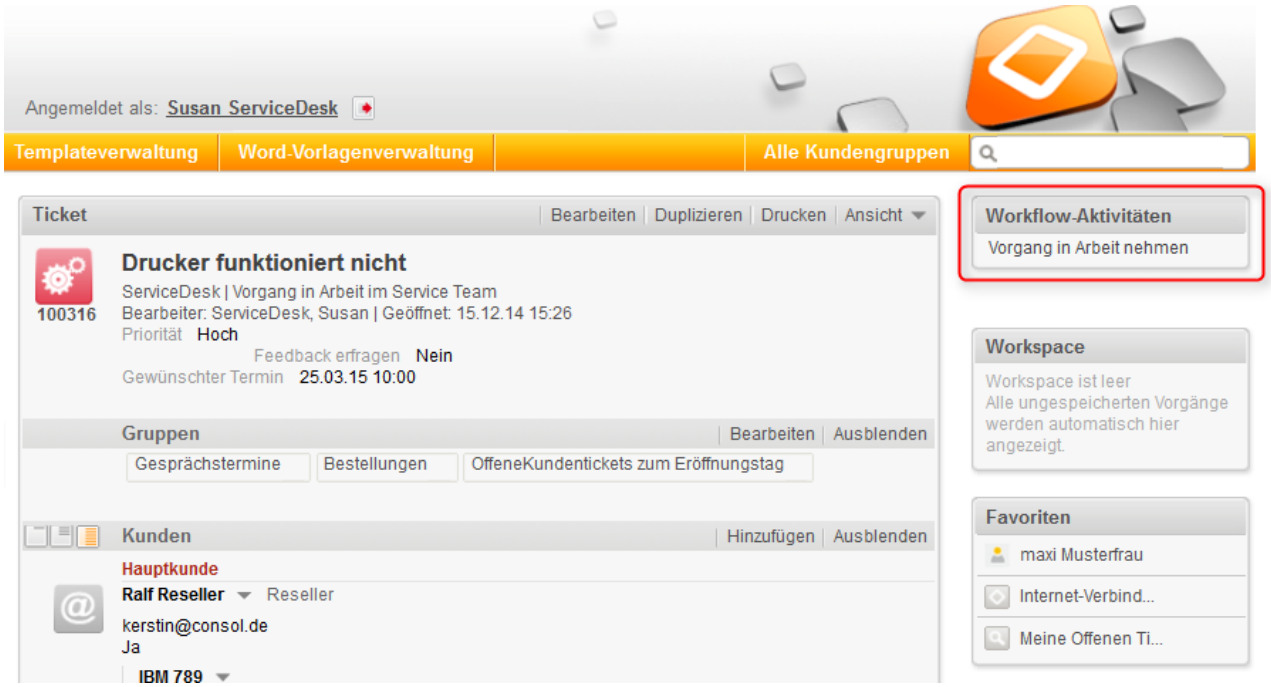


Fig. 2: ConSol*CM/Web Client - Manuelle Aktivität

Eine **automatische** Aktivität wird automatisch durch das System ausgeführt und nicht im Web Client angezeigt. Im Process Designer wird eine automatische Aktivität nicht durch ein spezielles Icon markiert.

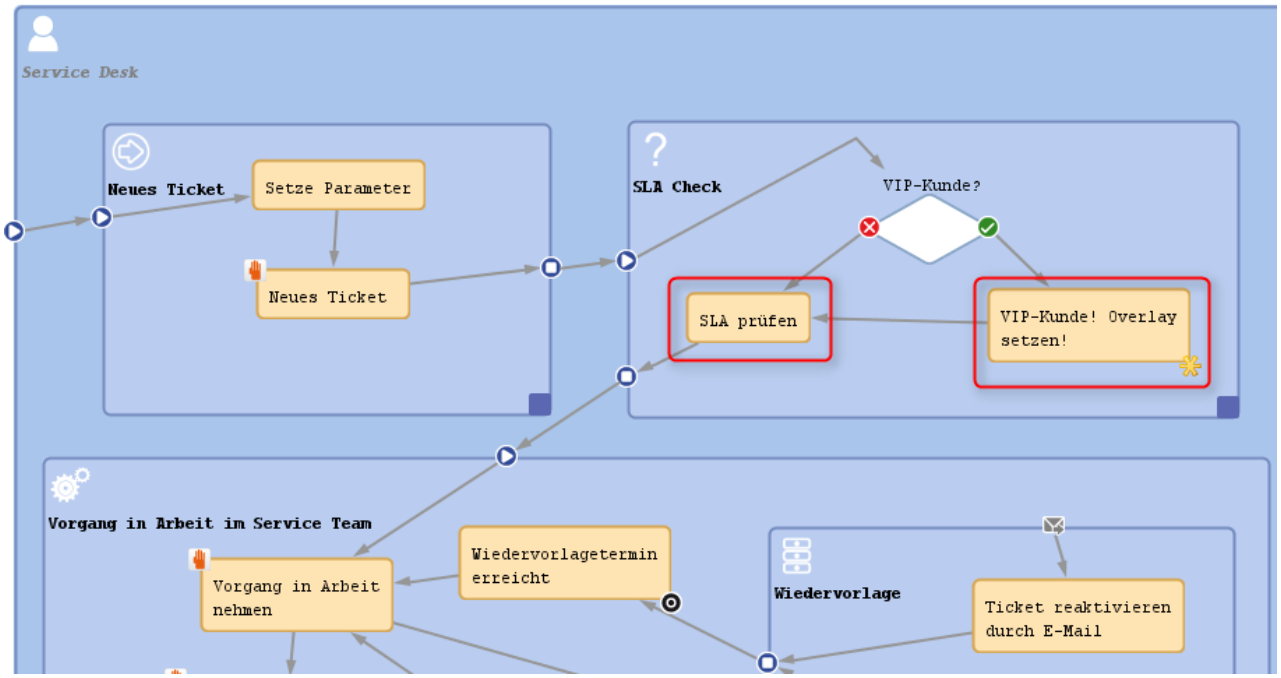


Fig. 3: ConSol*CM Process Designer - Automatische Aktivitäten

4.5.2 Eigenschaften einer Aktivität

Um die Eigenschaften einer Aktivität anzuzeigen und zu editieren, markieren Sie die Aktivität im Process Designer.

Fig. 4: ConSol*CM Process Designer - Aktivität

Der Eigenschaften-Editor für diese Aktivität wird geöffnet:

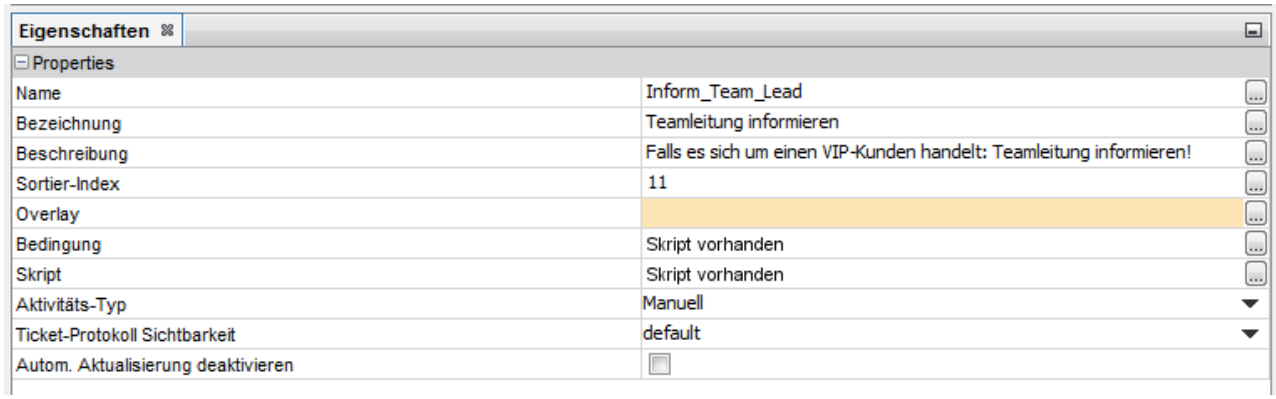


Fig. 5: ConSol*CM Process Designer - Eigenschaften einer Aktivität

Eine Aktivität kann die folgenden Eigenschaften besitzen:

- **Name**
  Notwendiger, technischer Name des Objekts.
- **Bezeichnung**
  Optional (wenn nichts gesetzt wird, wird der technische Name verwendet). Lokalisierter Name, der im Web Client angezeigt wird. Dabei wird die Sprache verwendet, die im Webbrowser eingestellt ist.
- **Beschreibung**
  Optional. Wird als Mouseover im Web Client angezeigt.
- **Sortier-Index**
  Definiert die Reihenfolge der angebotenen Aktivitäten im Web Client.
- **Overlay**
  Optional. Klicken Sie in den orangen Bereich, um eines der Standard-ConSol*CM-Overlays zu laden, oder verwenden Sie den Dateibrowser (...), um ein anderes Icon aus Ihrem Dateisystem hochzuladen.

- **Overlay-Gültigkeit**
  Wird nur angezeigt, wenn ein Overlay ausgewählt wurde.
  - **Aktivität**
    Das Overlay bleibt so lange am Ticket, wie das Ticket hinter der Aktivität steht. Sobald die nächste Aktivität ausgeführt wird, wird das Overlay vom Ticket entfernt.
  - **Bereich**
    Das Overlay wird entfernt, wenn das Ticket den Scope verlässt.
  - **Prozess**
    Das einmal zum Ticket-Icon hinzugefügte Overlay bleibt für den Rest des Prozesses am Ticket.
  - **Nächstes Overlay**
    Das Overlay bleibt so lange am Ticket, wie kein neues Overlay hinzugefügt wird. Sobald ein neues Overlay hinzugefügt wird, wird das alte entfernt.
- **Bedingung**
  Optional. Ein Skript kann eingegeben werden, welches ausgeführt wird, wenn die Aktivität im Web Client angezeigt werden soll. Das Skript muss *true* oder *false* zurückgeben. Wenn eine Bedingung für eine Aktivität definiert wurde, wird die Aktivität mit dem *Ausrufezeichen/Bedingungs*-Icon markiert (siehe Bild oben).
  - Zurückgegebener Wert ist **true**.
    Die Aktivität wird angezeigt. Wenn es eine manuelle Aktivität ist, kann sie durch einen Bearbeiter im Web Client ausgewählt/ausgeführt werden.
  - Zurückgegebener Wert ist **false**.
    Die Aktivität wird nicht im Web Client angezeigt.

> **Vorsicht:**
>
> CM Version 6.9 und höher:
> Wenn Sie mit Datenobjektgruppenfeldern arbeiten, d.h. Datenfeldern, die Kundendaten enthalten, beachten Sie bitte, dass es möglicherweise notwendig ist, die verschiedenen Datenmodelle der unterschiedlichen Kundengruppen zu berücksichtigen, wenn ein Workflow für Queues benutzt wird, die mehr als einer Kundengruppe zugewiesen wurden.

- **Skript**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Ticket die Aktivität durchläuft.
- **Aktivitäts-Typ**
  Notwendig. Es muss entweder *automatisch* oder *manuell* ausgewählt werden. Wenn es eine manuelle Aktivität ist, wird die Aktivität mit dem *Hand/Manuell*-Icon in der Process Designer GUI markiert.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll-Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Automatische Aktualisierung deaktivieren.

4.5.3 Prozesslogik von Aktivitäten

Folgendes ist die Prozesslogik von Aktivitäten:

1. Wenn ein Ticket eine Aktivität durchlaufen hat, wartet es immer **hinter** dieser Aktivität (und nicht vor der nächsten!).
2. Wenn ein Ticket eine Aktivität durchlaufen hat, überprüft es, ob es eine automatische Folgeaktivität gibt. Wenn ja, durchläuft das Ticket diese automatische Aktivität ebenfalls.
3. Das Ticket läuft automatisch durch (automatische) Aktivitäten, solange es neue automatische Aktivitäten gibt. Es hält an, sobald es eine oder mehrere manuelle Aktivitäten, die Interaktion mit einem Bearbeiter erfordern, gibt.
4. Wenn eine oder mehrere der folgenden manuellen Aktivitäten ein Bedingungsskript besitzen, wird dieses Skript ausgeführt, um zu entscheiden, ob die Aktvität im Web Client angezeigt werden muss oder nicht.
5. Wenn ein Bearbeiter die Aktivität im Web Client auswählt, wird das Skript der Aktivität ausgeführt.
6. Wenn es ein *postActivityScript* gibt, wird dieses Skript sofort nach der Ausführung des Aktivitätenskripts ausgeführt.
7. Das Ticket wartet hinter der manuellen Aktivität. Wenn die darauffolgende Aktivität sich in einem neuen Scope befindet, tritt das Ticket nicht in den neuen Scope ein. Es wartet hinter der alten Aktivität und nicht vor der neuen!

> **Vorsicht:**
>
> Ein Ticket wartet immer **hinter** der letzten Aktivität, die ausgeführt wurde, und **nicht** vor der neuen!!

4.5.4 Beispiele für Aktivitäten

Beispiel 1: Bedingung für die Anzeige der Aktivität "Teamleitung informieren"

Wenn das Ticket für einen *VIP*-Kontakt, d.h. einen Kontakt, bei dem das boolesche Feld *vip* auf *true* gesetzt ist, eröffnet wurde, soll der Teamleiter informiert werden. Wenn der Kunde kein *VIP* ist, soll diese Aktivität nicht zur Verfügung stehen. Zu diesem Zweck wird das Benutzerdefinierte Feld (bzw. Datenobjektgruppenfeld ab CM-Version 6.9) *vip*, das Teil des Kundendatenmodells ist, überprüft.

Fig. 6: ConSol*CM Process Designer - Workflow-Aktivitäten (eine davon mit Bedingungsskript)

Bedingungsskript: Workflow nur in Queues mit einer (derselben) Kundengruppe genutzt

```
// Hole den Hauptkontakt des Tickets. Das Unit-Objekt (kann ein Kunde oder eine Firma sein) ist
gegeben;
// hier muss es ein Objekt auf Kontakt-Ebene sein:
Unit contact = ticket.getMainContact()
// Ueberpruefe das Benutzerdefinierte Feld "vip" des Hauptkontakts. (siehe naechstes Bild)
// Wenn es auf true gesetzt ist, gebe true zurueck, d.h. die Bedingung ist TRUE.
// Andernfalls gebe false zurueck, d.h. die Bedingung ist FALSE:
if (contact.get("vip")) {
    return true
} else {
    return false
}
```

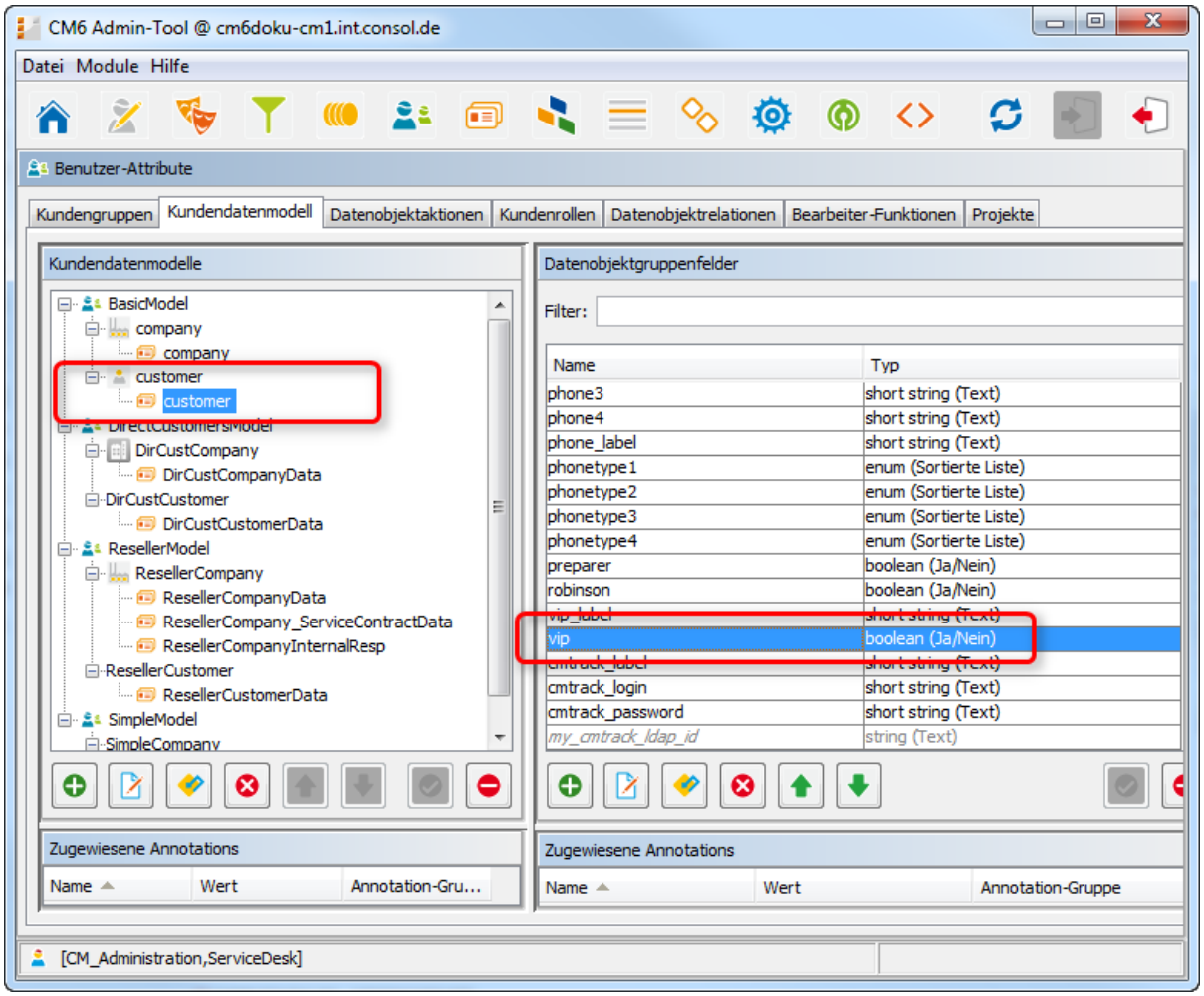


Fig. 7: ConSol*CM Admin-Tool - Datenobjektgruppenfeld "vip" (CM-Version 6.9)

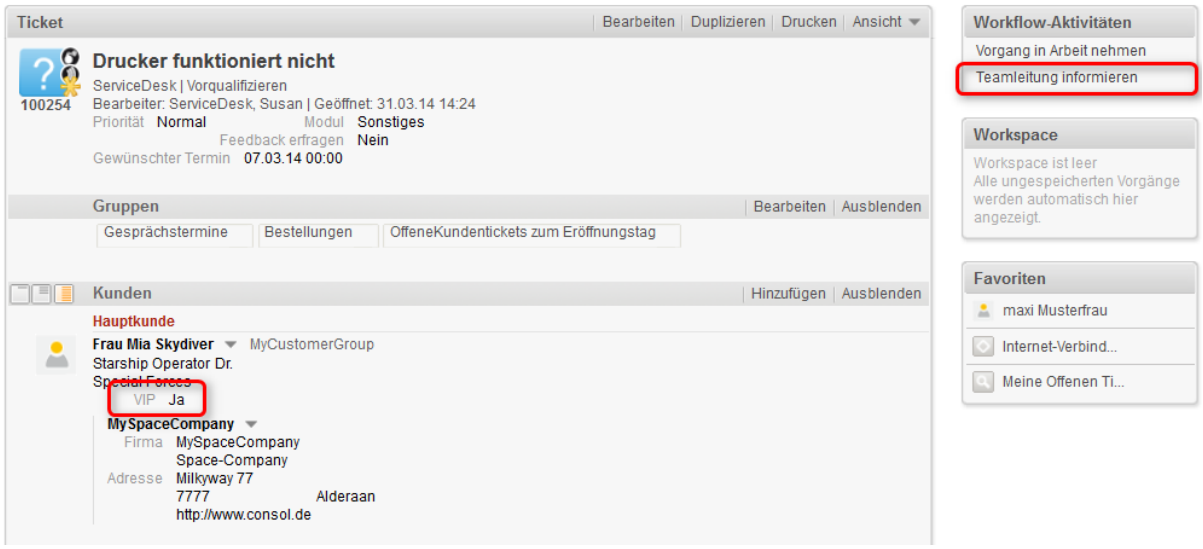


Fig. 8: ConSol*CM/Web Client - Bedingung: Rückgabewert TRUE

Fig. 9: ConSol*CM/Web Client - Bedingung: Rückgabewert FALSE

Beispiel 2: Sende eine E-Mail an den Hauptkunden, wenn das Ticket geöffnet wurde

Wenn ein Ticket geöffnet wird, soll eine E-Mail an den Hauptkunden des Tickets gesendet werden.

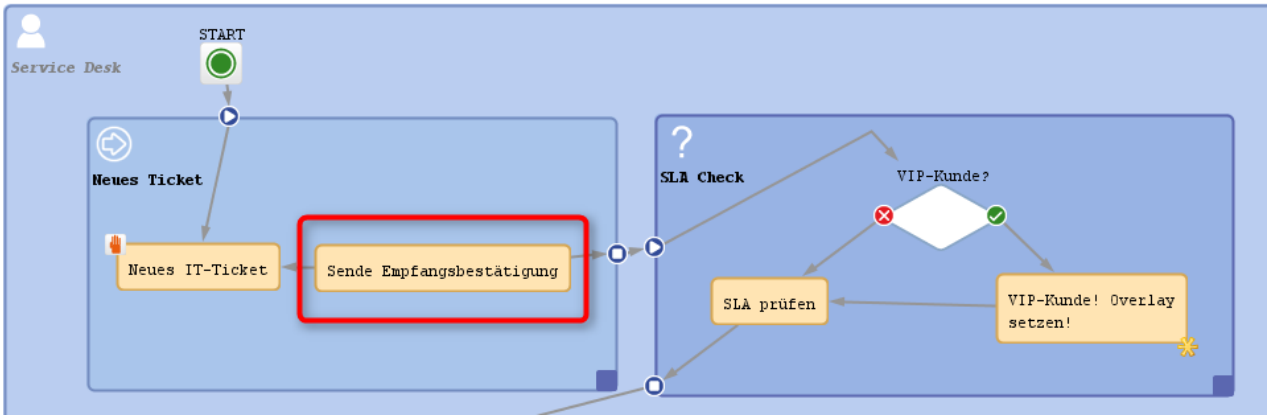


Fig. 10: ConSol*CM Process Designer - Automatische Aktivität, bei der eine Empfangsbestätigung versendet wird

Skript für automatische Aktivität, bei der eine Empfangsbestätigung versendet wird, Variante 1

```
// Hole den Hauptkontakt des Tickets (hier immer ein Objekt auf Kontakt-Ebene, nicht auf Firmen-Ebene):
def contact = ticket.getMainContact()
// Hole den Wert des Benutzerdefinierten Felds "email" des Hauptkontakts:
def contact_e = contact.get("email")
// Benutze als Text das E-Mail-Template mit dem Namen "receipt_notice_ServiceDesk".
// Dies kann sich im Template Designer oder im Admin-Tool befinden.
// Normalerweise werden E-Mail-Templates im Template Designer gespeichert:
def text = workflowApi.renderTemplate("receipt_notice_ServiceDesk")
// Hole die reply-to-Adresse für die E-Mails.
// Diese ist in diesem Beispiel in der System-Property "cmweb-server-adapter","mail.reply.to" gespeichert:
def replyto = configurationService.getValue("cmweb-server-adapter","mail.reply.to")
// Baue den String für den E-Mail-Betreff.
// Beachten Sie, dass der regulaere Ausdruck, der den Ticket-Indentifikator definiert, sich in diesem Betreff befinden muss.
// Andernfalls kann eine E-Mail nicht dem korrekten Ticket zugewiesen werden.
def subj = "Ihre Anfrage wurde von uns erhalten: ticket (" + ticket.getId() + ")"
//Versende die E-Mail
workflowApi.sendEmail(contact_e,subj,text,replyto,null)
```

Skript für automatische Aktivität, bei der eine Empfangsbestätigung versendet wird, Variante 2

```
// Alle Zeilen des Codes sind identisch zu Variante 1, mit Ausnahme der letzten Zeile:
new Mail().setSubject( subj ).setTo( contact_e ).setReplyTo( replyto ).setText( text ).
setTicketAttachments( null ).send()
```

Beispiel 3: Weise das Ticket dem aktuellen Bearbeiter zu

Das Ticket soll dem Bearbeiter zugewiesen werden, der die Aktivität *Neues IT-Ticket* ausführt.

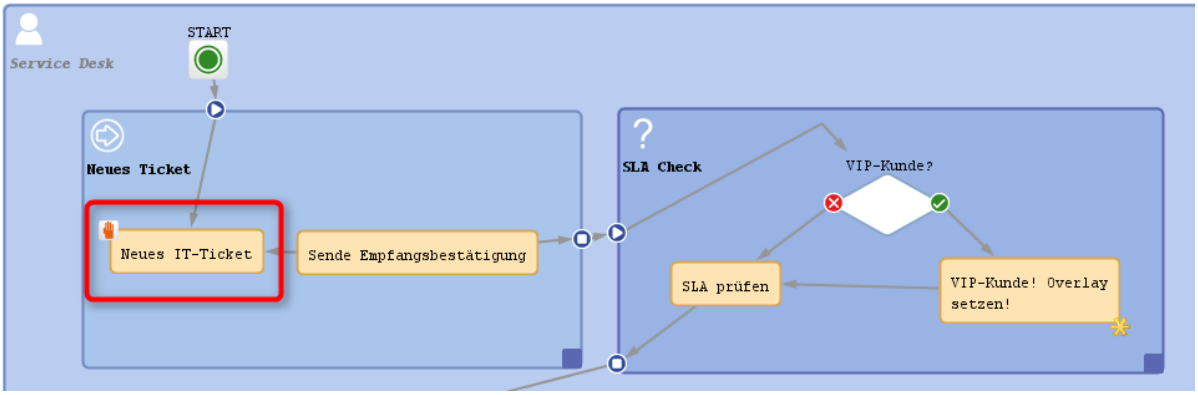


Fig. 11: ConSol*CM Process Designer - Workflow-Aktivität, bei der ein Bearbeiter zugewiesen werden soll

Fig. 12: ConSol*CM/Web Client - Ticket hat die Aktivität durchlaufen, bei der ein Bearbeiter zugewiesen wurde

Skript für das Zuweisen des Tickets an den aktuellen Bearbeiter (current engineer)

```
// Hole den Bearbeiter, der die Aktivitaet ausfuehrt:
def curr_eng = workflowApi.getCurrentEngineer()
// Weise das Ticket dem aktuellen Bearbeiter zu
ticket.setEngineer(curr_eng)
```

Vorsicht:

Stellen Sie sicher, dass Sie immer das korrekte Bearbeiter-Objekt verwenden!

Der aktuelle Bearbeiter ist der Bearbeiter, der gerade eingeloggt ist und die aktuelle Aktivität ausführt. Sie können dieses Objekt mittels der folgenden Methode holen:

```
def curr_eng = workflowApi.getCurrentEngineer()
// liefert immer true, wenn die Aktivitaet manuell ausgefuehrt wird, da dan
```

Der Bearbeiter des Tickets ist die Person, die (zu diesem Zeitpunkt) als Ticket-Bearbeiter in dem enstprechenden Datenfeld gesetzt ist und die damit für das Ticket verantwortlich ist. Sie können dieses Objekt mittels der folgenden Methode holen:

```
def tic_eng = ticket.getEngineer()
// liefert nur true, wenn das Ticket einem Bearbeiter zugewiesen ist, sonst
```

4.6 Workflow-Komponenten: Entscheidungsknoten

- Einleitung zu Entscheidungsknoten
- Eigenschaften eines Entscheidungsknotens
- Beispiel für einen Entscheidungsknoten

4.6.1 Einleitung zu Entscheidungsknoten

Ein Entscheidungsknoten ist ein Knoten, der einen oder mehrere Eingangspunkte und genau zwei Ausgangspunkte besitzt: *true* und *false*. Ein Entscheidungsknoten muss immer ein Skript besitzen, welches entweder *true* oder *false* zurückgibt.

Das Ticket tritt in den Entscheidungsknoten ein, dann wird das Skript ausgeführt und - abhängig vom Ergebnis (*true* oder *false*) - verlässt das Ticket den Knoten über den entsprechenden Ausgangspunkt.

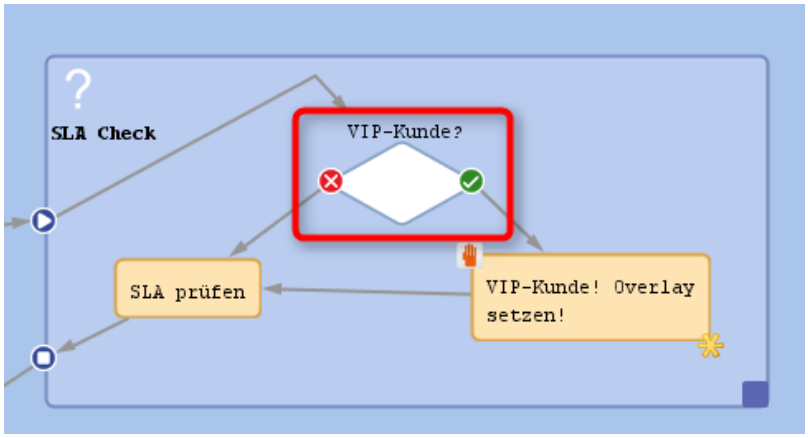


Fig. 1: ConSol*CM Process Designer - Entscheidungsknoten

4.6.2 Eigenschaften eines Entscheidungsknotens

Ein Entscheidungsknoten besitzt die folgenden Eigenschaften:

- **Name**
  Notwendiger, technischer Name des Objekts.
- **Bezeichnung**
  Optional. Lokalisierter Name, der im Web Client angezeigt wird.
- **Bedingung**
  Notwendig. Ein Skript, das *true* oder *false* zurückgibt, muss hinterlegt werden.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll-Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Automatische Aktualisierung deaktivieren.

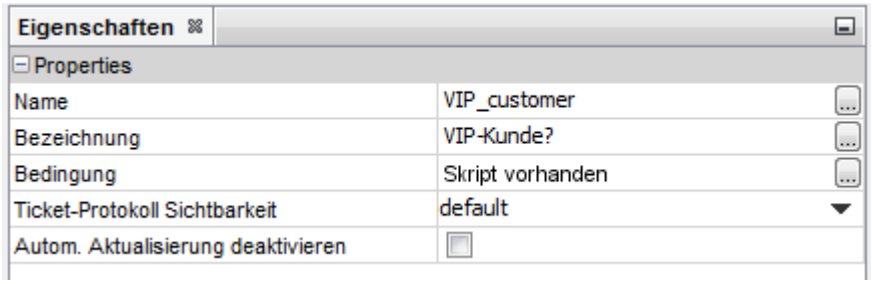


Fig. 2: ConSol*CM Process Designer - Eigenschaften eines Entscheidungsknotens

4.6.3 Beispiel für einen Entscheidungsknoten

Im folgenden Beispiel (CM-Version 6.8) soll das System automatisch überpüfen, ob der Kunde (Hauptkontakt des Tickets) ein *VIP*-Kunde ist. Falls ja, soll das Ticket mit dem *VIP*-Overlay markiert werden (in diesem Beispiel eine gelbe Sonne).

1. Ein Benutzerdefiniertes Feld vom Typ *boolean* muss im Kundendatenmodell definiert werden, um einen Kunden als *VIP* zu markieren (ja/nein). Bitte lesen Sie dazu das *ConSol*CM Administratorhandbuch 6.8*, Abschnitt *Verwaltung der Benutzerdefinierten Felder.*

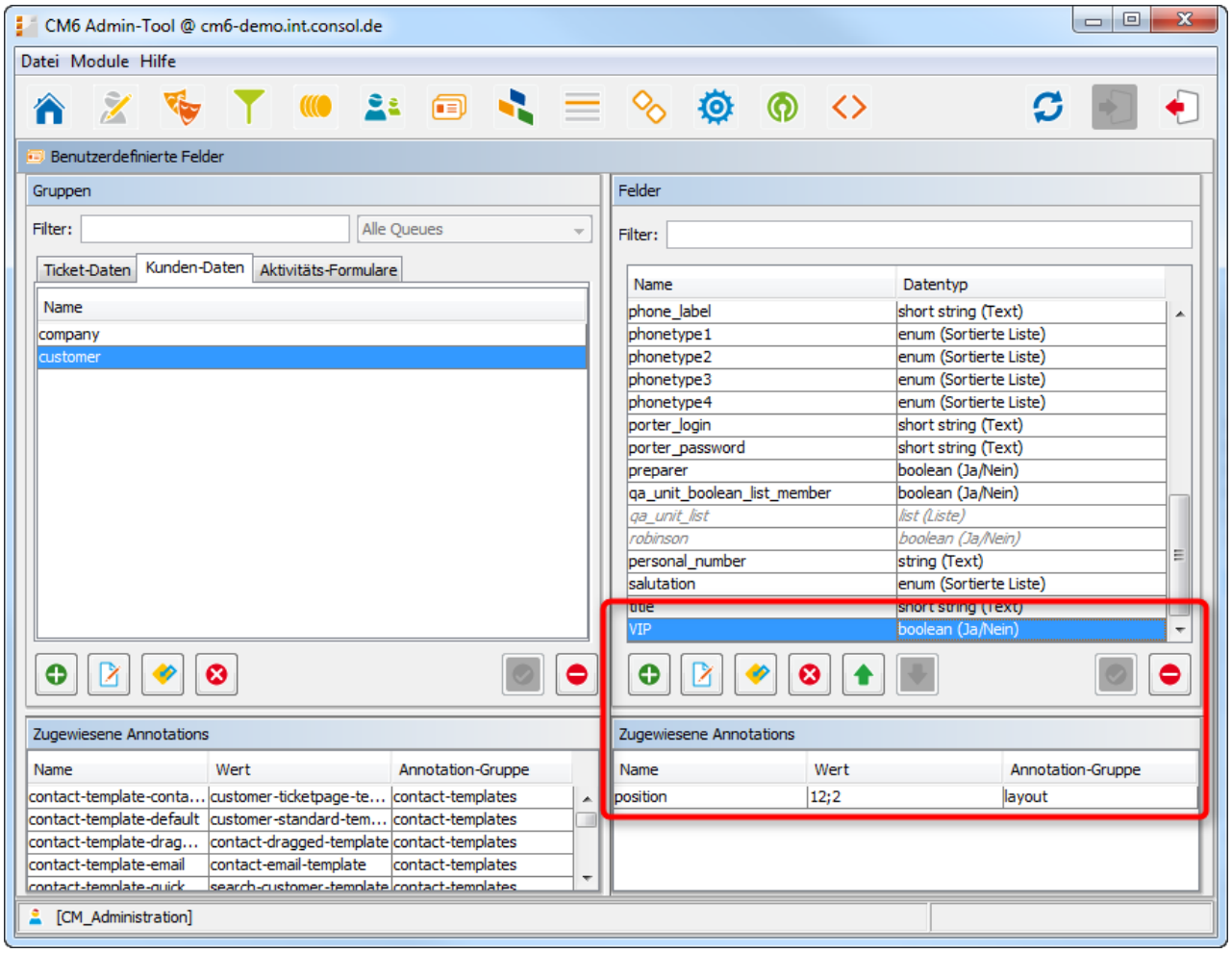


Fig. 3: ConSol*CM Admin-Tool - Benutzerdefiniertes Feld "VIP" in den Kundendaten (CM-Version 6.8)

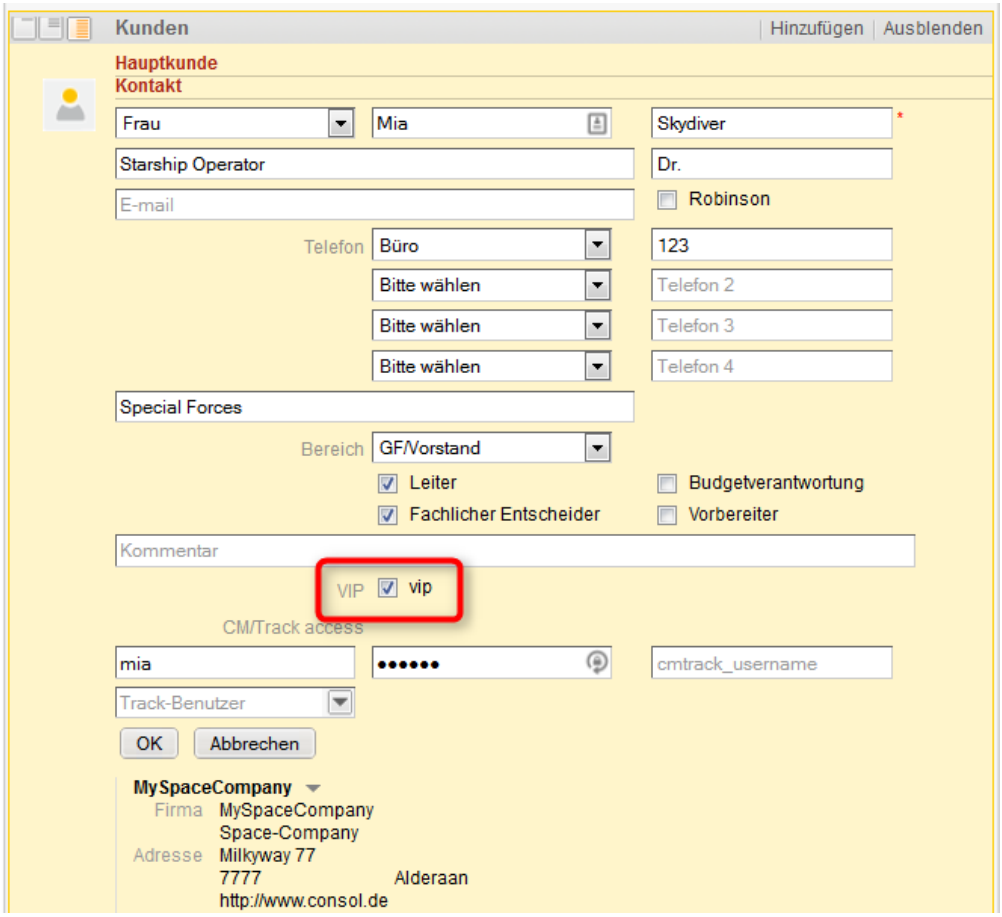


Fig. 4: ConSol*CM/Web Client - Benutzerdefiniertes Feld "VIP" für Kundendaten

2. Im Skript des Entscheidungsknotens muss überprüft werden, ob der Kunde ein *VIP* ist (zurückgegebener Wert: *true*) oder nicht (zurückgegebener Wert: *false*).

Beispiel für CM-Version 6.8

```
// Hole den Hauptkontakt des Tickets. Das Unit-Objekt (kann ein Kontakt oder einer Firma sein)
ist gegeben;
// hier muss es ein Kunde sein, d.h. ein Kontakt:
Unit contact = ticket.getMainContact()
// Ueberpruefe das Benutzerdefinierte Feld "vip" des Hauptkontakts. (Siehe naechstes Bild)
// Wenn es auf true gesetzt ist, gebe true zurueck, d.h. die Bedingung ist TRUE.
// Andernfalls gebe false zurueck, d.h. die Bedingung ist FALSE:
if (contact.get("VIP")) {
    return true
} else {
    return false
}
```

3. Wenn das Ticket automatisch den Entscheidungsknoten und die folgende automatische Aktivität, bei der ein *VIP*-Overlay hinzugefügt wird, durchlaufen hat, wird das Ticket-Icon im Web Client mit dem Overlay markiert, siehe folgendes Bild.

Fig. 5: ConSol*CM/Web Client - Ticket-Icon mit VIP-Overlay

4.7 ConSol CM Process Designer Handbuch - Adornments (Trigger und ACFs)

4.7.1 Adornments (Trigger und ACFs)

Die ConSol*CM-Workflow-Engine kann auf verschiedene Typen von Ereignissen reagieren. Dies wird über Trigger gesteuert. ACFs bieten dynamische Formulare an.

Adornment-Typ	Erklärung
Zeit-Trigger	Kontrollieren die Zeit, welche verstrichen ist, seit das Ticket in einen Scope oder eine Aktivität gelangt ist. Siehe Abschnitt Zeit-Trigger.
Mail-Trigger	Kontrollieren, ob eine E-Mail von einem Ticket, das sich in diesem Scope befindet, empfangen wurde. Siehe Abschnitt Mail-Trigger.
Business-Event-Trigger	Kontrollieren Ereignisse wie den Wechsel eines Bearbeiters oder das Hinzufügen eines Kommentars. Siehe Abschnitt Business-Event-Trigger.
ACF	Mittels Aktivitätsformularen (Activity Control Forms = ACFs) können Sie die Daten kontrollieren, die an einem bestimmten Prozessschritt vom Bearbeiter eingegeben werden müssen. Siehe Abschnitt Aktivitätsformulare (ACFs).

4.7.2 Zeit-Trigger

- Einleitung zu Zeit-Triggern
- Hinzufügen eines Zeit-Triggers zu einem Workflow
  - Hinzufügen eines Zeit-Triggers zu einem Scope
  - Hinzufügen eines Zeit-Triggers zu einer Aktivität
- Eigenschaften eines Zeit-Triggers
- Business-Logik und Initialisierung von Zeit-Triggern
- Beispiele für Zeit-Trigger
- Skripting mit Zeit-Triggern
  - Beispiel 1: Setze die Ablaufzeit eines Zeit-Triggers in Abhängigkeit zur Queue
  - Beispiel 2: Berechne die Eskalation als Warnung zwei Tage vor dem gewünschten Enddatum

Einleitung zu Zeit-Triggern

Ein Workflow kann mehrere Zeit-Trigger enthalten.

Fig. 1: ConSol*CM Process Designer - Zeit-Trigger

Ein Zeit-Trigger ist ein Mechanismus, der reagiert, wenn eine bestimmte Zeitspanne abgelaufen ist. Dies kann z.B. in den folgenden Situationen nötig sein:

- **Anwendungsfall 1:**
  Ein Bearbeiter möchte ein Ticket für eine bestimmte Zeit auf Wiedervorlage legen, weil er weiß, dass der Kunde bis dahin nicht erreichbar sein wird.
- **Anwendungsfall 2:**
  Das System soll automatisch die Eskalationszeit steuern, d.h. wenn ein Ticket hereingekommen ist und niemand sich um das Ticket kümmert, sollte es eine Warnung geben (dies kann ein Overlay am Ticket-Icon sein, eine E-Mail an den Teamleiter oder andere Aktionen).
- **Anwendungsfall 3:**
  Ein Ticket wurde gelöst und der Bearbeiter schließt es. Dies soll allerdings nur das vorläufige Ende sein und das Ticket soll erst nach einer definierten Zeitspanne technisch geschlossen werden.

Diese Anwendungsfälle können durch Zeit-Trigger implementiert werden.

Ein Zeit-Trigger kann so konfiguriert werden, dass er einen Arbeitszeitkalender verwendet, d.h. dass er nur die Stunden, die als Arbeitszeit definiert wurden, berücksichtigt.

Ein Zeit-Trigger kann hängen an ...

- **einem Scope**
  Dann kontrolliert der Trigger alle Tickets, die sich gerade in diesem Scope befinden.
- **einer Aktivität**
  Dann kontrolliert der Trigger nur die Tickets, die gerade in diese Aktivität eingetreten sind.

Ein Zeit-Trigger muss einem dieser beiden Typen angehören:

- manuell
- mit einer definierten Zeitspanne

> **Information:**
>
> Sie als Workflow-Entwickler müssen alles implementieren, was als Konsequenz auf das Feuern des Triggers erfolgt! Es gibt keine automatischen Aktionen. Der Zeit-Trigger sorgt lediglich für das Signal *Zeit abgelaufen*, genau wie ein Wecker.

Hinzufügen eines Zeit-Triggers zu einem Workflow

Hinzufügen eines Zeit-Triggers zu einem Scope

Klicken Sie auf das Zeit-Trigger-Icon in der Palette und ziehen Sie den Trigger in den gewünschten Scope. Er wird automatisch an den oberen Rand des Scopes hinzugefügt. Sie können seine Position danach verändern (bewegen Sie ihn nach links oder rechts, um die Reihenfolge von Triggern zu verändern oder einfach nur, um das Layout zu verbessern).

Ein Zeit-Trigger, der mit einem Scope verbunden wurde, kann nicht in einen anderen Scope oder an eine andere Aktivität verschoben werden. Wenn Sie den Zeit-Trigger mit einem anderen Scope oder einer anderen Aktivität verbinden möchten, entfernen Sie den Trigger, den Sie definiert haben, und erstellen Sie einen neuen für den gewünschten Scope.

Um die Eigenschaften des Triggers zu konfigurieren, wählen Sie ihn im Hauptarbeitsbereich aus und stellen Sie die gewünschten Werte im Eigenschaften-Editor ein. Siehe Abschnitt Eigenschaften eines Zeit-Triggers.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach einem Zeit-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

Hinzufügen eines Zeit-Triggers zu einer Aktivität

Klicken Sie auf das Zeit-Trigger-Icon in der Palette und ziehen Sie den Trigger in die gewünschte Aktivität. Er wird an die Ecke der Aktivität hinzugefügt.

Ein Zeit-Trigger, der mit einer Aktivität verbunden wurde, kann nicht in einen anderen Scope oder an eine andere Aktivität verschoben werden.Wenn Sie den Zeit-Trigger mit einem anderen Scope oder einer anderen Aktivität verbinden möchten, entfernen Sie den Trigger, den Sie definiert haben und erstellen Sie einen neuen für die gewünschte Aktivität.

Um die Eigenschaften des Triggers zu konfigurieren, wählen Sie ihn im Hauptarbeitsbereich aus und stellen Sie die gewünschten Werte im Eigenschaften-Editor ein. Siehe Abschnitt Eigenschaften eines Zeit-Triggers.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach einem Zeit-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

Eigenschaften eines Zeit-Triggers

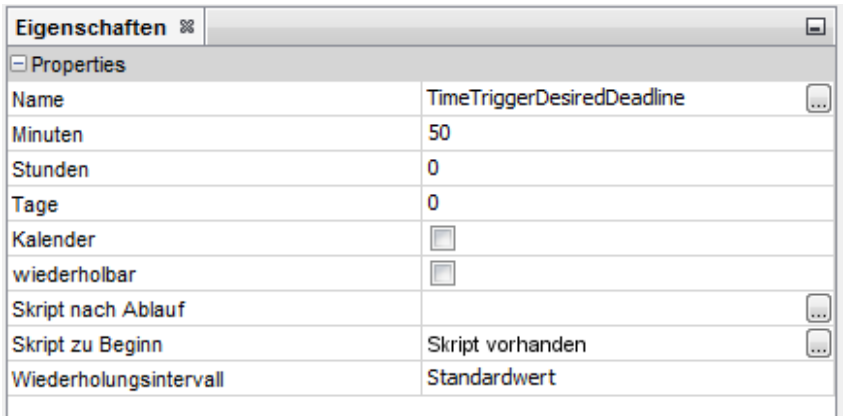


Fig. 2: ConSol*CM Process Designer - Eigenschaften eines Zeit-Triggers

Ein Zeit-Trigger besitzt die folgenden Eigenschaften:

- **Name**
  Notwendig. Technischer Name des Triggers. Er wird automatisch gesetzt, kann aber manuell geändert werden.
- **Minuten/Stunden/Tage**
  Hier geben Sie das Zeitintervall ein, nach dem der Trigger feuern soll. Die Anzeige bezieht sich immer auf einen 24-Stunden-Tag, d.h. wenn Sie *30 Stunden* als Reaktionszeit eingetragen haben und Sie den Workflow erneut öffnen, wird die Anzeige *1 Tag, 6 Stunden* anzeigen.

- **Kalender**
  Optional. Aktivieren Sie diese Checkbox, wenn ein Arbeitszeitkalender für die Berechnung des Zeitintervalls berücksichtigt werden soll.

  Vorsicht:

  Bitte beachten Sie, dass drei Schritte notwendig sind, um sicherzustellen, dass Zeitspannen unter Berücksichtigung eines Arbeitszeitkalenders berechnet werden:
  1. Definieren Sie einen Arbeitszeitkalender (siehe *ConSol*CM Administratorhandbuch*, Abschnitt *Arbeitszeitkalender).*
  2. Weisen Sie den korrekten Arbeitszeitkalender der gewünschten Queue zu (siehe *ConSol*CM Administratorhandbuch,* Abschnitt *Queue-Verwaltung*).
  3. Aktivieren Sie die Checkbox *Kalender* für jeden Trigger, der mit dem Arbeitszeitkalender arbeiten soll.

  Prinzip der Benutzung eines Arbeitszeitkalenders

  1 Tag bedeutet 24 Stunden absoluter Zeit, dies hat nichts mit der Benutzung eines Arbeitszeitkalenders zu tun. Der Arbeitszeitkalender spielt erst dann eine Rolle, wenn der Zeit-Trigger aktiviert ist, dann werden die 24 Stunden, d.h. 86400000 Millisekunden, als Arbeitszeitkalender-Input genommen (wenn der Kalender aktiviert ist).
  Beispiel:
  Wenn wir als Zeitspanne des Zeit-Triggers 1 Tag = 24 Stunden ohne Kalender haben, werden die 24 Stunden wie normale Zeit berechnet, d.h. die Eskalation feuert einen Tag später, um genau die gleiche Zeit.
  Im Gegensatz dazu: Wenn wir einen Kalender benutzen (mit z.B. 7 Arbeitsstunden pro Arbeitstag), werden die 24 Stunden dem Kalender entsprechend aufgeteilt. Dies führt dazu, dass das Feuern des Triggers mehr als 3 Tage später erfolgt (24 Stunden = 3 x 7 Stunden + 3 Stunden).

  Siehe dazu auch Abschnitt Arbeiten mit Kalendern und Zeiten.
- **wiederholbar**
  Optional. Aktivieren Sie diese Checkbox, um sicherzustellen, dass der Trigger mehr als einmal für ein Ticket feuern kann. Wenn ein Trigger *wiederholbar* ist, wird er sofort, nachdem er gefeuert hat, wieder zurückgesetzt, d.h. die Zeitmessung beginnt erneut.

  Info für Experten:

  Das Skript zu Beginn des Zeit-Triggers wird erneut ausgeführt. Das erste Feuern des Triggers wird durch den (technischen) Benutzer *admin* ausgelöst, alle folgenden Feuerungen des Triggers werden durch den *Job Executor* ausgelöst.

- **Skript nach Ablauf**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Zeitintervall, das durch den Trigger kontrolliert wird, abgelaufen ist, d.h. wenn der Trigger feuert.
- **Skript zu Beginn**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn der Zeit-Trigger anfängt, die Zeit zu messen, d.h. wenn das Ticket in den Scope bzw. die Aktivität eingetreten ist, mit dem/der der Trigger verbunden ist.
- **manuell**
  Optional, nur für Zeit-Trigger an Aktivitäten. Aktivieren Sie diese Checkbox, wenn der Bearbeiter im Web Client auswählen soll, nach welcher Zeit der Trigger feuern soll. Für den Bearbeiter wird in diesem Fall ein Datumsauswahl-Fenster (Web-Kalender) angezeigt.
- **Wiederholungsintervall**
  Die Zeit in Sekunden, nach der die Trigger-Ausführung erneut ausgeführt werden soll, für den Fall, dass ein Skript in einen Fehler gelaufen ist. Die Zeit kann im Admin-Tool konfiguriert werden (System-Property *cmas-workflow-engine*, *jobExecutor.timerRetryInterval.seconds*).

Business-Logik und Initialisierung von Zeit-Triggern

Die Zeitmessung eines Triggers beginnt (d.h. der Trigger wird initialisiert), wenn das Ticket in den Scope/die Aktivität eintritt. Sie hält an (d.h. der Trigger feuert), wenn das definierte Zeitintervall, das als fester Wert gesetzt wurde (Minuten/Stunden/Tage), oder die manuell definierte Zeit verstrichen ist.

Wenn Sie als Workflow-Entwickler einen Trigger mittels anderer Werte initialisieren möchten, muss dies über Skripte erfolgen. An dieser Stelle erhalten Sie kurze Beispiele dazu, bitte lesen Sie den Abschnitt Arbeiten mit Kalendern und Zeiten für eine detaillierte Erklärung zur Programmierung von Workflow-Zeit-Triggern. In diesen Kapiteln werden Ihnen auch Code-Beispiele gegeben.

- **Beispiel 1:**
  Die Reaktionszeit für ein Ticket soll auf Basis der Priorität berechnet werden. Im *Skript zu Beginn* werden die verschiedenen Reaktionszeiten verwendet (ein guter Weg, um dies zu implementieren, wäre über kundenspezifische System-Properties) und die Reaktionszeit wird berechnet. Danach wird der Trigger initialisiert, d.h. das Zeitintervall wird gesetzt.
  Beispiel 2:
  Wenn eine E-Mail für ein Ticket hereingekommen ist und nach drei Stunden noch kein Bearbeiter die E-Mail gelesen und sich um das Ticket gekümmert hat, soll eine Warnung ausgelöst werden. Um dies zu implementieren, besitzt der entsprechende Mail-Trigger (siehe Abschnitt Mail-Trigger) eine nachfolgende bzw. angehängte automatische Aktivität, welche den Zeit-Trigger mit drei Stunden re-initialisiert.

Ein Zeit-Trigger kann auch deaktiviert werden. Für *Beispiel 2* würde dies notwendig sein, um den Trigger davon abzuhalten, initial zu feuern, da er nicht initialisiert werden soll, bevor eine E-Mail eintrifft.

Beispiele für Zeit-Trigger

Die Implementierungen für die oben angegebenen Anwendungsfälle (siehe Einleitung zu Zeit-Triggern) wären:

- **Anwendungsfall 1: Wiedervorlage**
  Hängen Sie einen manuellen Zeit-Trigger an die Aktivität *Wiedervorlage*. Der Bearbeiter kann das gewünschte Enddatum mittels des Datumsauswahl-Fensters im Web Client auswählen. Normalerweise wird das Ticket dann zu den aktiven Tickets zurück geleitet.

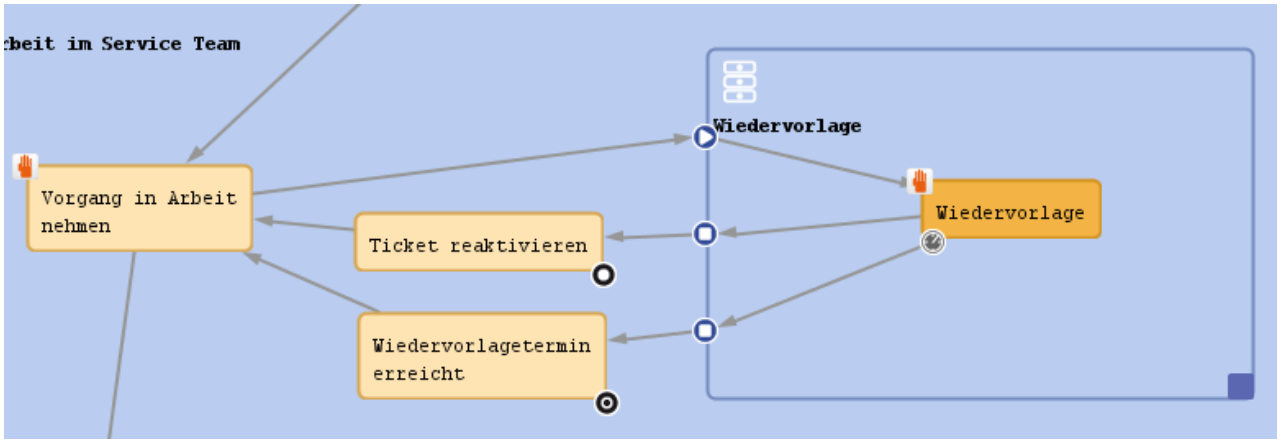


Fig. 3: ConSol*CM Process Designer - Anwendungsfall 1: Workflow

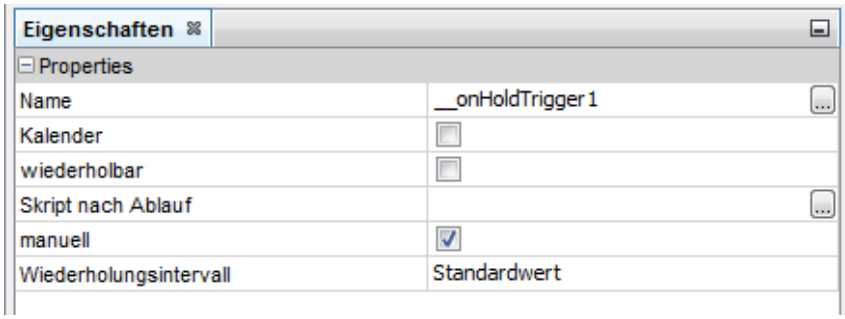


Fig. 4: ConSol*CM Process Designer - Anwendungsfall 1: Eigenschaften-Editor für den Zeit-Trigger

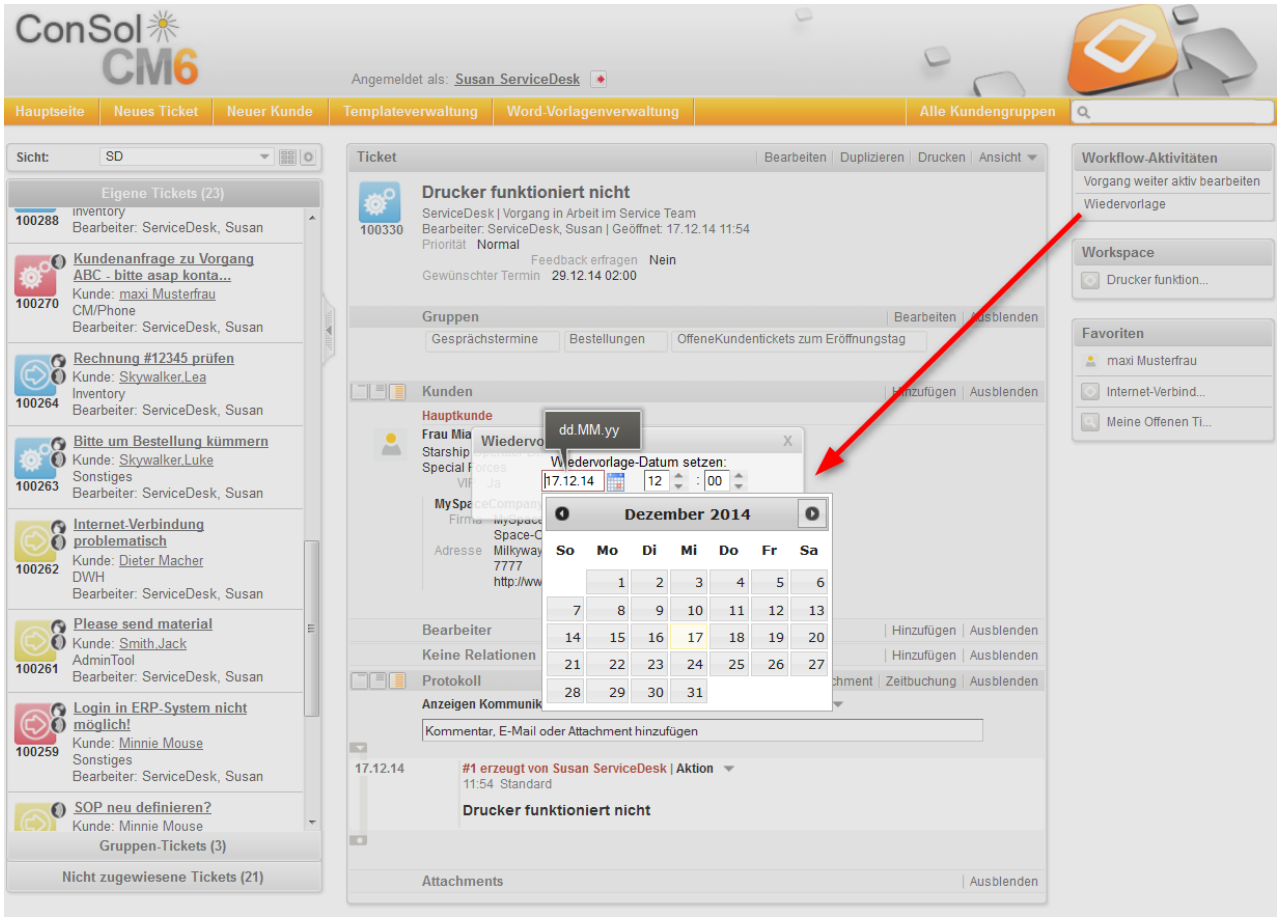


Fig. 5: ConSol*CM/Web Client - Anwendungsfall 1: Datumsauswahl-Fenster

- **Anwendungsfall 2: Bearbeiter gesetzt?**
  Setzen Sie einen Zeit-Trigger an den Scope, in dem neue Tickets ankommen. Definieren Sie die Zeit für den Trigger (dies kann von SLAs abhängen), z.B. vier Stunden. Setzen Sie eine Kontrolle (Entscheidungsknoten) hinter den Trigger, ob ein Bearbeiter sich um das Ticket gekümmert hat oder nicht. Wenn nicht, wird eine E-Mail an die Teamleitung gesendet.

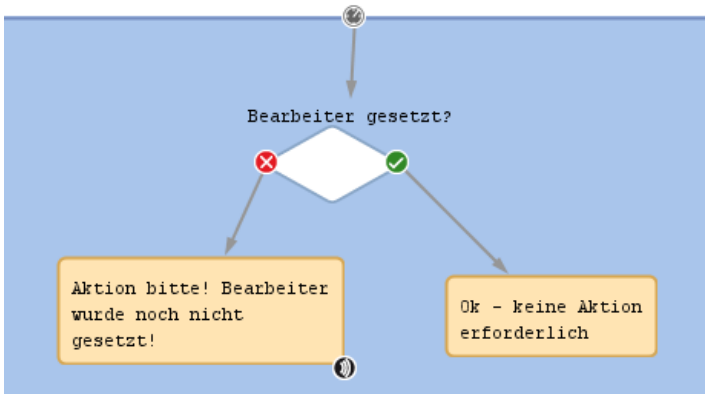


Fig. 6: ConSol*CM Process Designer - Anwendungsfall 2: Workflow

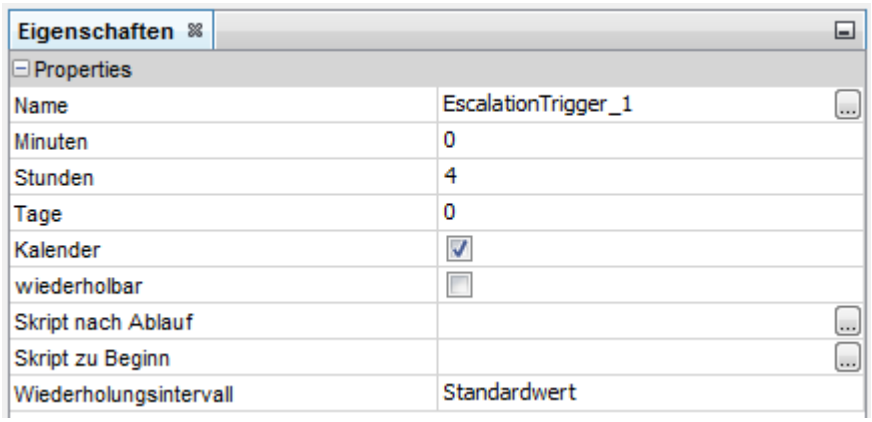


Fig. 7: ConSol*CM Process Designer - Anwendungsfall 2:Eigenschaften-Editor für den Zeit-Trigger

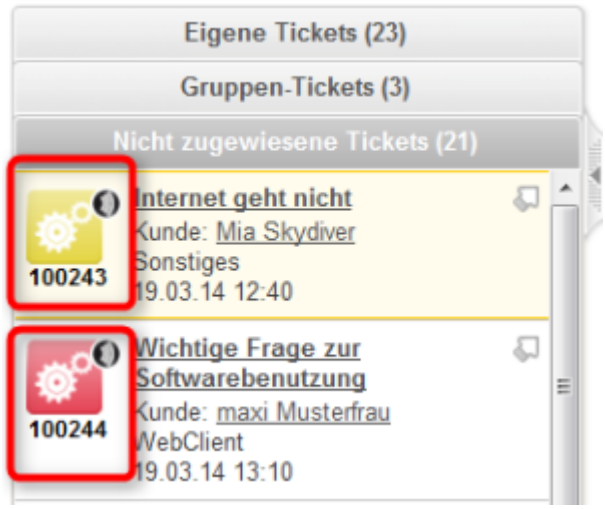


Fig. 8: ConSol*CM/Web Client - Anwendungsfall 2: Ticketliste

- **Anwendungsfall 3:Vorläufiger Abschluss**
  Setzen Sie einen Zeit-Trigger an die Aktivität *Ticket abschließen mit Lösung (positiv)* und legen Sie ein Zeitintervall für den Zeit-Trigger fest, z.B. fünf Tage. Hinter dem Trigger liegt der Endknoten des Prozesses. Für fünf Tage kann das Ticket noch bearbeitet werden, nach dieser Zeit wird es automatisch geschlossen.

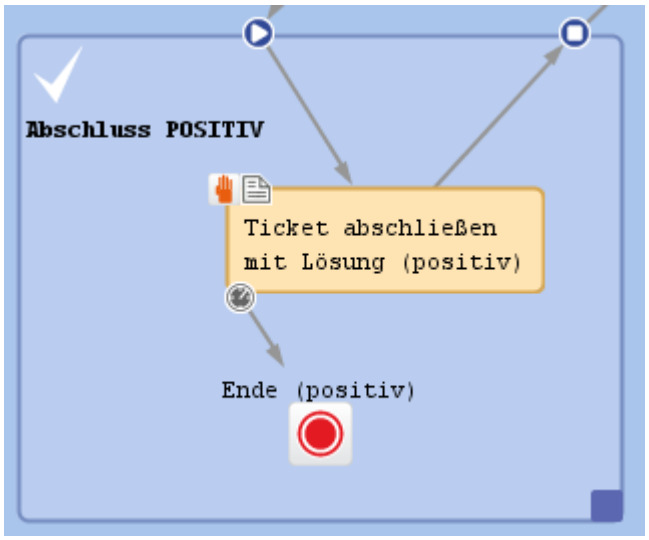


Fig. 9: ConSol*CM Process Designer - Anwendungsfall 3: Workflow

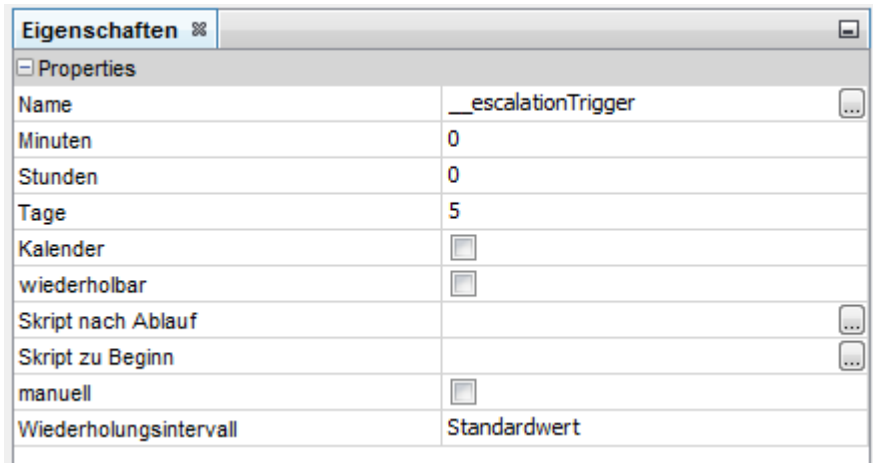


Fig. 10: ConSol*CM Process Designer - Anwendungsfall 3: Eigenschaften-Editor für Zeit-Trigger

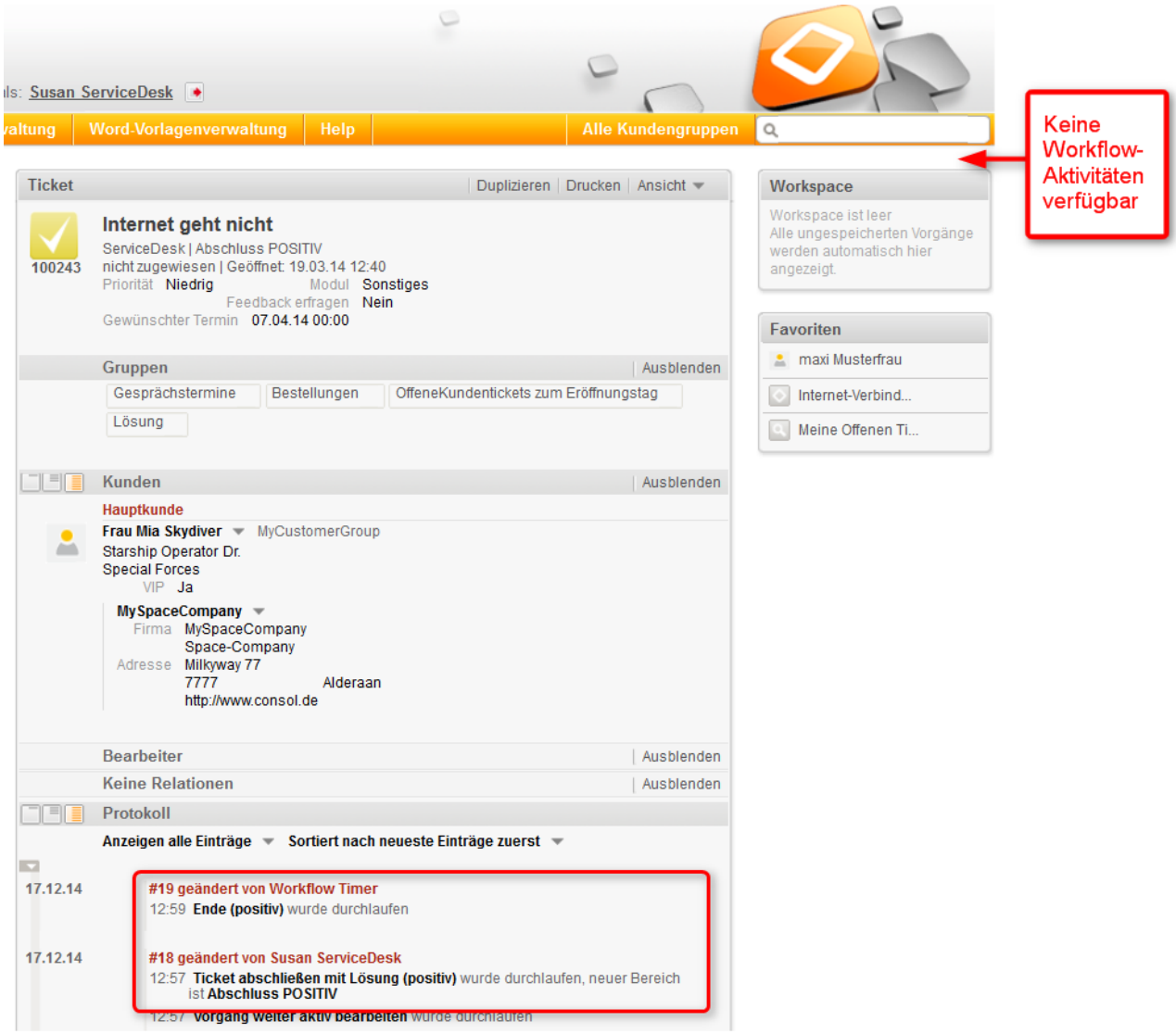


Fig. 11: ConSol*CM/Web Client - Geschlossenes Ticket

Skripting mit Zeit-Triggern

Die folgenden Methoden sind von großer Wichtigkeit, wenn Sie mit Zeit-Triggern arbeiten:

- **TimerTrigger.setDueTime(long pDueTime in millisecs)**
  Legt die Zeit fest, wann der Trigger feuern soll. Die Zeitaufnahme startet, wenn das Ticket in den Scope oder die Aktivität eintritt, an den/die der Trigger angehängt ist. *setDueTime()* definiert also die Zeitperiode in Millisekunden von der Eintrittszeit bis zur gewünschten Feuerung des Triggers.
- **workflowApi.reinitializeTrigger()**
  (verschiedene Methoden-Signaturen)
  Startet die Zeitaufnahme für den entsprechenden Trigger erneut, d.h. setzt seine Startzeit zurück.

- **workflowApi.deactivateTimer()**
  (verschiedene Methoden-Signaturen)
  Deaktiviert den entsprechenden Zeit-Trigger, d.h. der Trigger wird nicht feuern, bevor er nicht re-initialisiert wird.
  (Es gibt **keine** Methode *activateTimer().* Benutzen Sie *workflowApi.reinitializeTrigger()*, um den Trigger zu reaktivieren).

Bitte lesen Sie dazu auch den Abschnitt Arbeiten mit Kalendern und Zeiten.

Beispiel 1: Setze die Ablaufzeit eines Zeit-Triggers in Abhängigkeit zur Queue

Dieses Skript kann als *Skript zu Beginn* für einen Zeit-Trigger an einem Scope verwendet werden. Es initialisiert den Trigger für eine Eskalation in Abhängigkeit von der Queue, d.h. wenn das Ticket sich in der Queue *HelpDesk_1st_Level* befindet, ist die Zeitspanne, bis es zur Eskalation kommt, geringer, als in der Queue *HelpDesk_2nd_Level*.

Innerhalb der Skripte *Skript zu Beginn* und *Skript nach Ablauf* existiert das Objekt *trigger* als eine implizite Initialisierung von *TimerTrigger*. Daher können Sie mit Triggern ohne weitere vorherige Schritte arbeiten. In einem Admin-Tool-Skript müssen Sie jedoch die Klasse *TimerTrigger* oder das entsprechende Java-Package importieren.

Das folgende Skript kann in einer Servicedesk- und Helpdesk-Umgebung und im folgenden *TimerTrigger* verwendet werden:

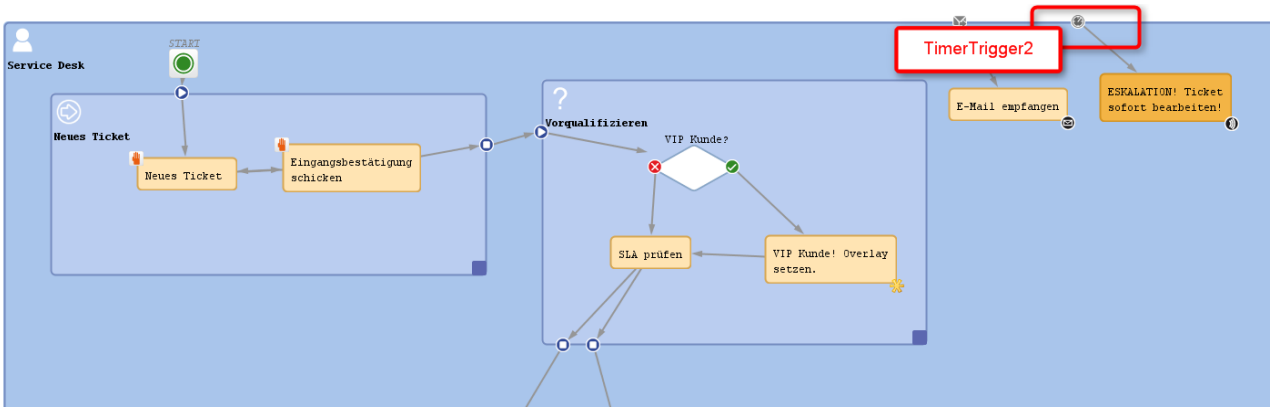


Fig. 12: ConSol*CM Process Designer - TimerTrigger im Workflow Service Desk

Bespiel für ein "Skript zu Beginn"

```
def addedEscalMillis = 0
switch (ticket.queue.name) {
  case "HelpDesk_1st_Level":
    addedEscalMillis = 12*60*60*1000L;
    break;
  case "HelpDesk_2nd_Level":
    addedEscalMillis = 24*60*60*1000L;
    break;
  case "ServiceDesk":
    addedEscalMillis = 4*60*60*1000L;
 }
trigger.setDueTime(addedEscalMillis)
```

Vorsicht:

Für dieses Beispiel war es sinnvoll, feste Werte für die Zeiten direkt im Skript-Code zu verwenden. In echten Umgebungen empfiehlt es sich, die Eskalationszeiten und dergleichen in System-Properties zu speichern und sie mittels des *configurationService* abzurufen. Auf diese Weise besitzt ein Administrator leichten Zugriff auf die Eskalationszeiten und kann diese bearbeiten, ohne die Workflow-Implementierung verändern zu müssen.

Im realen Arbeitsalltag würde vielleicht auch ein Arbeitszeitkalender benutzt werden - bitte schauen Sie sich dazu *Beispiel 2* an.

In der *server.log*-Datei können Sie sehen, zu welcher Zeit der Trigger feuern soll:

Fig. 13: Datei server.log mit der berechneten DueTime

Das gleiche Prinzip kann auch angewendet werden, um die Eskalationszeit in Abhängigkeit von der Ticket-Priorität, dem *VIP*-Status eines Kunden oder anderer Parameter zu berechnen.

Beispiel 2: Berechne die Eskalation als Warnung zwei Tage vor dem gewünschten Enddatum

Berechne und setze die Zeit für TimerTrigger mittels eines Arbeitszeitkalenders

```
def now = new Date()
def wunschTermin = ticket.get("helpdesk_standard", "date_test")
def twoWorkDays = -2*8*60*60*1000L
// Berechne Eskalationsdatum
def escalDate = BusinessCalendarUtil.getBusinessTime(wunschTermin, twoWorkDays, ticket.queue.
calendar)
// Berechne und setze Ablaufzeit
trigger.setDueTime(escalDate.time - now.time)
```

4.7.3 Mail-Trigger

- Einleitung zu Mail-Triggern
  - Mail-Trigger an einem Scope
  - Mail-Trigger an einer Aktivität
- Hinzufügen eines Mail-Triggers zu einem Workflow
  - Hinzufügen eines Mail-Triggers zu einem Scope
  - Hinzufügen eines Mail-Triggers zu einer Aktivität
- Eigenschaften eines Mail-Triggers
- Beispiele für Mail-Trigger
  - Anwendungsfall 1: Overlay für ein Ticket-Icon
  - Anwendungsfall 2: Overlay für ein Ticket-Icon und E-Mail-Bestätigung durch einen Bearbeiter
- Prozesslogik mit Mail-Triggern

Einleitung zu Mail-Triggern

Eine der Kernfunktionalitäten von ConSol*CM ist dessen Interaktion mit einer E-Mail-Infrastruktur. Dies ermöglicht es einem Bearbeiter, manuell E-Mails zu verschicken, und dem System, automatische E-Mails an Kunden oder Bearbeiter zu verschicken, so wie es im entsprechenden Business-Prozess-Schritt erforderlich ist. Natürlich muss ConSol*CM auch E-Mails empfangen. Dies geschieht durch das Abholen von E-Mails aus einem oder mehreren Postfächern mit ConSol*CM-zugehörigen Adressen. Für eine detaillierte Erklärung aller Interaktionen zwischen dem Mail-Server und ConSol*CM lesen Sie bitte das *ConSol*CM Administratorhandbuch und das ConSol*CM Betriebshandbuch*. An dieser Stelle werden nur die Workflow-Interaktionen erklärt.

Fig. 1: ConSol*CM Process Designer - Mail-Trigger

Mail-Trigger an einem Scope

Wenn eine E-Mail empfangen wird, die zu einem existierenden und aktiven (geöffneten) Ticket gehört, kann es notwendig sein, dass dieser Vorgang registriert wird und anschließend bestimmte Aktionen ausgeführt werden müssen. Dies kann durch den Einsatz von einem oder mehreren Mail-Triggern innerhalb eines Workflows erreicht werden.

> **Vorsicht:**
>
> Bitte beachten Sie, dass (in der Standard-Konfiguration, d.h. ohne Modifikation des Admin-Tool-Skripts *AppendToTicket.groovy*) die **einzige automatische Aktion**, die von ConSol*CM, nachdem eines der Postfächer eine E-Mail empfängt, ausgeführt wird, ist, diese E-Mail an das Ticket mit dem passenden Ticket-Tag im Betreff, z.B. *Ticket (<TicketNumber>)*, anzuhängen. Siehe dazu auch das *ConSol*CM Administratorhandbuch,* Abschnitt *Skripte vom Typ E-Mail*.
>
> **Alle anderen Aktionen,** die nach dem Empfang einer E-Mail ausgeführt werden sollen, müssen manuell im Workflow programmiert werden (und/oder in Admin-Tool-Skripten)!

Beispiele für den Gebrauch von Mail-Triggern sind:

Wenn eine E-Mail empfangen wurde ...

- soll der Bearbeiter eines Tickets ebenfalls eine E-Mail als Benachrichtigung erhalten.
- soll das Ticket-Icon (im Web Client) mit einem Overlay markiert werden.
- soll das Ticket in eine andere Aktivität verschoben werden, bei der der Bearbeiter bestätigen muss, dass er das Ticket gelesen hat.
- werden der Absender und der Betreff der E-Mail überprüft und geparsed. Wenn die E-Mail eine Bestätigung oder eine Ablehnung eines Genehmiger-Prozesses ist, wird das Ticket entsprechend der definierten Regeln und Aktivitäten im Workflow behandelt. Auf diese Weise kann die Genehmigung direkt mittels der E-Mail vollzogen werden, es ist kein Login des Genehmigers in den Web Client erforderlich.

Mail-Trigger an einer Aktivität

Wenn ein Mail-Trigger an eine Aktivität hinzugefügt wird, wird diese Aktivität nur ausgeführt, wenn eine E-Mail empfangen wird.

Fig. 2: ConSol*CM Process Designer - Mail-Trigger an einer Aktivität

Hinzufügen eines Mail-Triggers zu einem Workflow

Hinzufügen eines Mail-Triggers zu einem Scope

Klicken Sie auf das Mail-Trigger-Icon in der Palette und ziehen Sie den Trigger in den gewünschten Scope. Er wird automatisch an den oberen Rand des Scopes hinzugefügt. Sie können seine Position danach verändern (bewegen Sie ihn nach links oder rechts, um das Layout zu verbessern). Es kann nur ein Mail-Trigger pro Scope verwendet werden.

Ein Mail-Trigger, der mit einem Scope verbunden wurde, kann nicht in einen anderen Scope verschoben werden. Wenn Sie den Mail-Trigger mit einem anderen Scope verbinden möchten, entfernen Sie den Trigger, den Sie definiert haben, und erstellen Sie einen neuen für den gewünschten Scope.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach einem Mail-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

Hinzufügen eines Mail-Triggers zu einer Aktivität

In den sehr seltenen Fällen, in denen Sie einen Mail-Trigger mit einer Aktivität verbinden möchten (wir empfehlen dies nicht!), klicken Sie auf das Mail-Trigger-Icon in der Palette und ziehen Sie den Trigger in die gewünschte Aktivität. Er wird an die Ecke der Aktivität hinzugefügt.

Ein Mail-Trigger, der mit einer Aktivität verbunden wurde, kann nicht in einen anderen Scope oder in eine andere Aktivität verschoben werden. Wenn Sie den Mail-Trigger mit einem anderen Scope oder einer anderen Aktivität verbinden möchten, entfernen Sie den Trigger, den Sie definiert haben, und erstellen Sie einen neuen für den gewünschten Scope bzw. die gewünschte Aktivität.

Eigenschaften eines Mail-Triggers

Ein Mail-Trigger besitzt keine Eigenschaften.

Beispiele für Mail-Trigger

Anwendungsfall 1: Overlay für ein Ticket-Icon

Wenn eine E-Mail für ein Ticket empfangen wurde, das sich gerade in diesem Scope befindet, soll das Ticket-Icon im Web Client mit dem Overlay *mail* markiert werden.

Der Mail-Trigger ist mit dem Scope verbunden und das Overlay mit der angrenzenden automatischen Aktivität angefügt. Die Gültigkeit des Overlays ist *Aktivität*. Auf diese Weise wird das Ticket mit einem Overlay markiert, wenn eine E-Mail eingetroffen ist. Sobald ein Bearbeiter das Ticket in eine weitere Aktivität bewegt hat, verschwindet das Overlay wieder.

Bitte beachten Sie, dass das Ticket seinen Kontext nicht verlässt. Es wird lediglich ein Overlay zum Ticket-Icon hinzugefügt. Danach kehrt das Ticket wieder an seine Originalposition im Workflow zurück. Dies wird *Interrupt* genannt. Bitte lesen Sie für eine detaillierte Erklärung dazu den Abschnitt Prozesslogik.

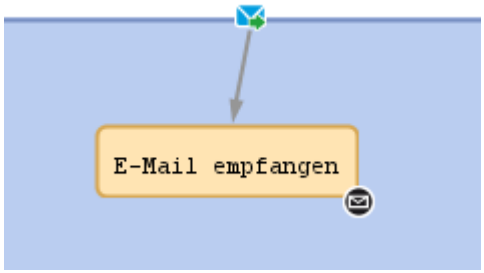


Fig. 3: ConSol*CM Process Designer - Anwendungsfall 1: Scope mit Mail-Trigger

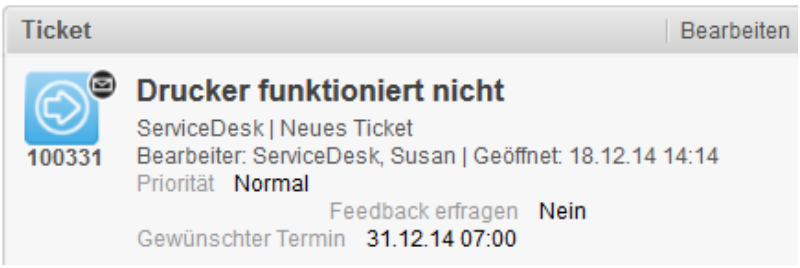


Fig. 4: ConSol*CM/Web Client - Anwendungsfall 1: Ticket-Icon mit Overlay

Anwendungsfall 2: Overlay für ein Ticket-Icon und E-Mail-Bestätigung durch einen Bearbeiter

Wenn eine E-Mail für ein Ticket empfangen wurde, das sich gerade in diesem Scope befindet, soll das Ticket-Icon im Web Client mit dem Overlay *mail* markiert werden. Zusätzlich soll das Ticket an eine Position verschoben werden, an der es so lange wartet, bis ein Bearbeiter bestätigt hat, dass er die E-Mail gelesen hat.

Der Mail-Trigger ist mit dem Scope verbunden und das Overlay mit der angrenzenden automatischen Aktivität angefügt. Die Gültigkeit des Overlays ist *Aktivität*. Auf diese Weise wird das Ticket mit einem Overlay markiert, wenn eine E-Mail eingetroffen ist.

In dem Skript, das auf den Mail-Trigger folgt, wird ein boolsches Feld *mail_to_read* auf *true* gesetzt. Im Workflow wird eine Aktivität *E-Mail gelesen* angeboten, wo immer diese benötigt wird. Sie wird nur angezeigt, wenn der Wert des boolschen Felds *mail_to_read true* ist. Dies stellt einen stärkeren Mechanismus als das alleinige Verwenden eines Overlays dar, um den Bearbeiter an eine eingetroffene E-Mail zu erinnern. Der Bearbeiter muss die E-Mail bestätigen, indem er die Workflow-Aktivität *E-Mail gelesen* explizit ausführt. Innerhalb dieser Workflow-Aktivität wird der Wert des boolschen Felds *mail_to_read* zurück auf *false* gestellt. Nun ist das Ticket bereit, eine weitere E-Mail zu erhalten und den Bearbeiter erneut darüber zu benachrichtigen.

Bitte beachten Sie, dass in diesem Fall das Ticket ebenfalls nicht als Konsequenz auf die Aktion, die nach dem Eintreffen der E-Mail ausgeführt wird, seinen Kontext verlässt. Es wird lediglich ein Overlay zum Ticket-Icon hinzugefügt und eine boolsche Variable verändert. Danach kehrt das Ticket wieder an seine Originalposition im Workflow zurück. Dies ist also ebenfalls ein Interrupt. Bitte lesen Sie für eine detaillierte Erklärung dazu den Abschnitt Prozesslogik.

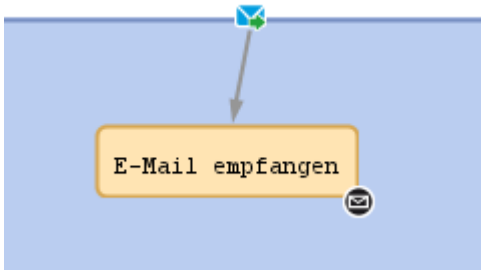


Fig. 5: ConSol*CM Process Designer - Anwendungsfall 2: Scope mit Mail-Trigger

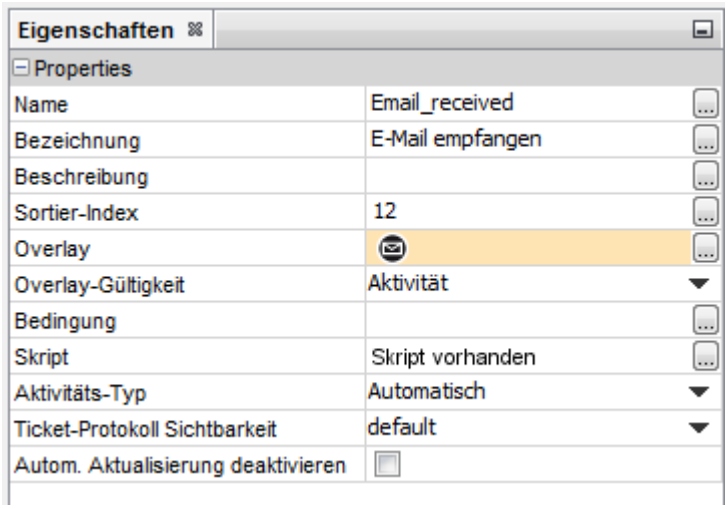


Fig. 6: ConSol*CM Process Designer - Eigenschaften der Aktivität "E-Mail empfangen"

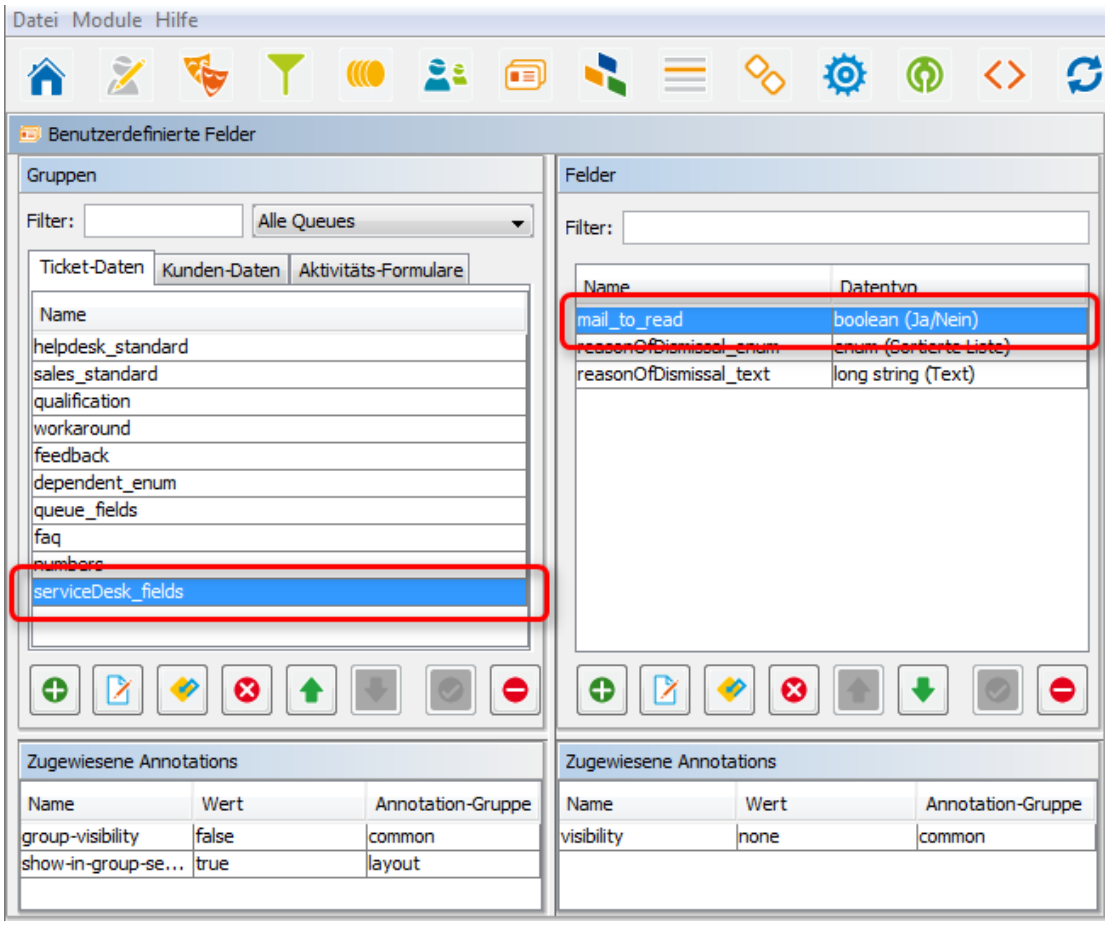


Fig. 7: ConSol*CM Admin-Tool - Anwendungsfall 2: Neues boolsches Feld zur Mail-Registrierung

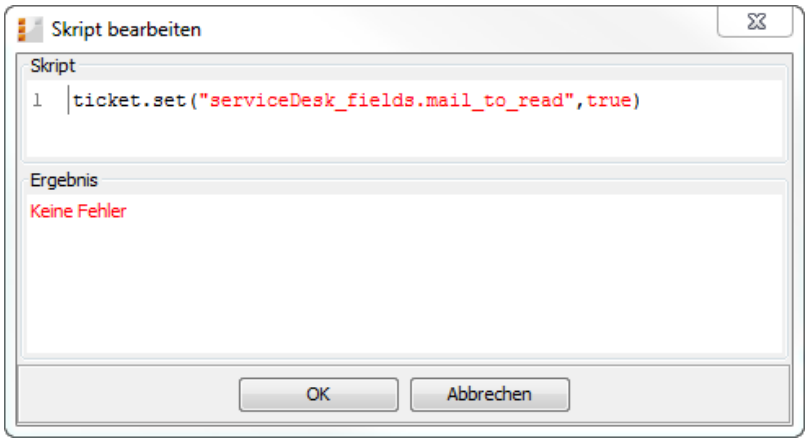


Fig. 8: ConSol*CM Process Designer - Anwendungsfall 2: Skript für die Aktivität "E-Mail empfangen"

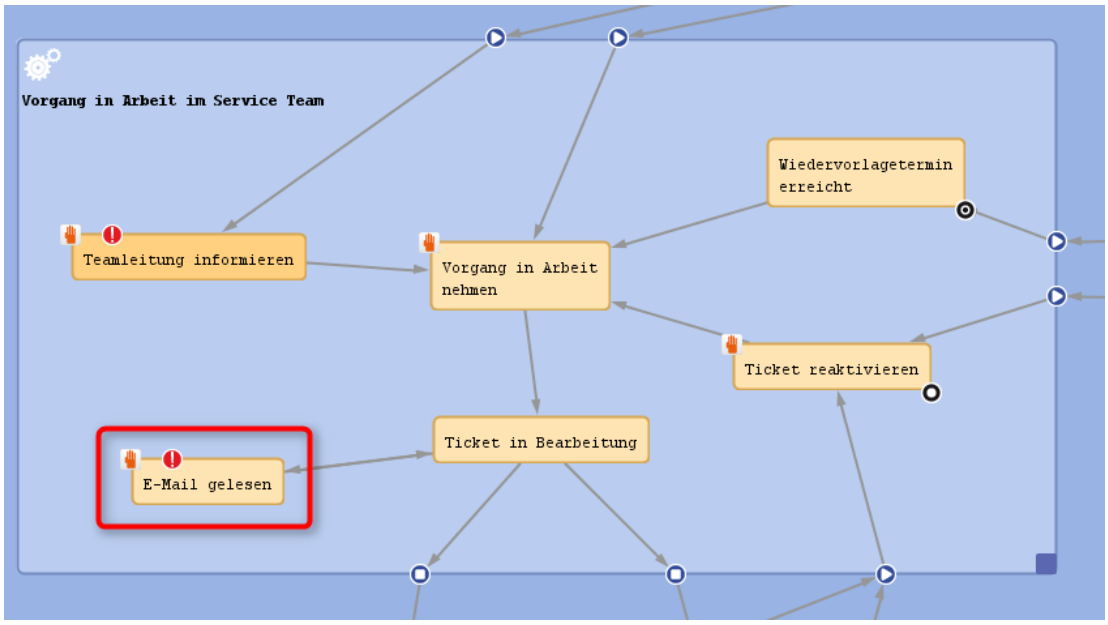


Fig. 9: ConSol*CM Process Designer - Anwendungsfall 2: Aktivität für die Bestätigung der E-Mail

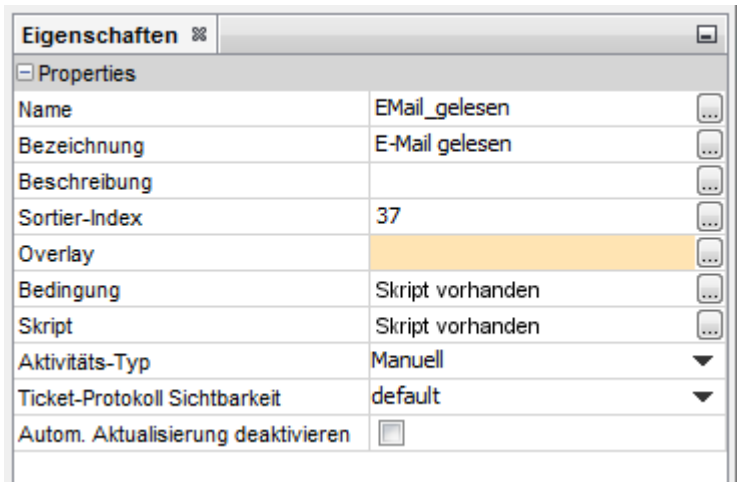


Fig. 10: ConSol*CM Process Designer - Anwendungsfall 2: Eigenschaften der Aktivität "E-Mail gelesen"

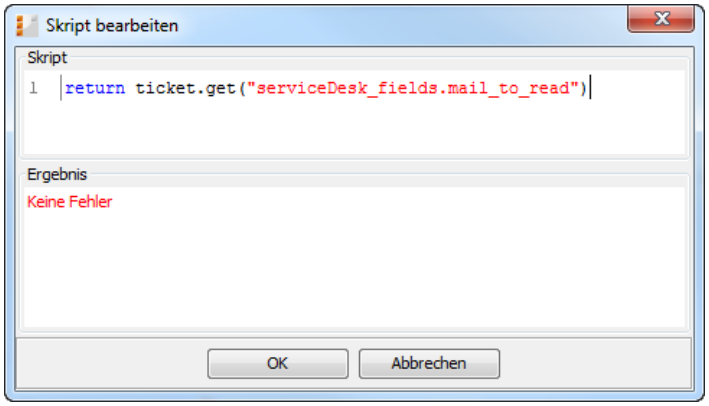


Fig. 11: ConSol*CM Process Designer - Anwendungsfall 2: Bedingungsskript für die Aktivität "E-Mail gelesen"

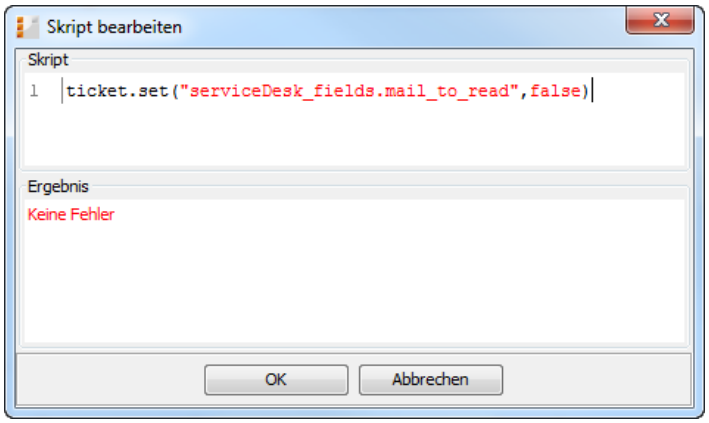


Fig. 12: ConSol*CM Process Designer - Anwendungsfall 2: Skript für die Aktivität "E-Mail gelesen"

Fig. 13: ConSol*CM/Web Client - Anwendungsfall 2: Workflow-Aktivität "E-Mail gelesen"

Prozesslogik mit Mail-Triggern

Wenn eine E-Mail empfangen wird, feuert der Mail-Trigger, der sich an dem innersten Scope befindet.

Beispiel 1:
Das Ticket befindet sich an Position **(1)** im Scope *Vorgang in Arbeit im Service Team*. Wenn eine E-Mail hereinkommt, feuert der Mail-Trigger, der sich an diesem Scope befindet **(2)**, und als Konsequenz daraus wird das Ticket in einen anderen Scope verschoben **(3)**.

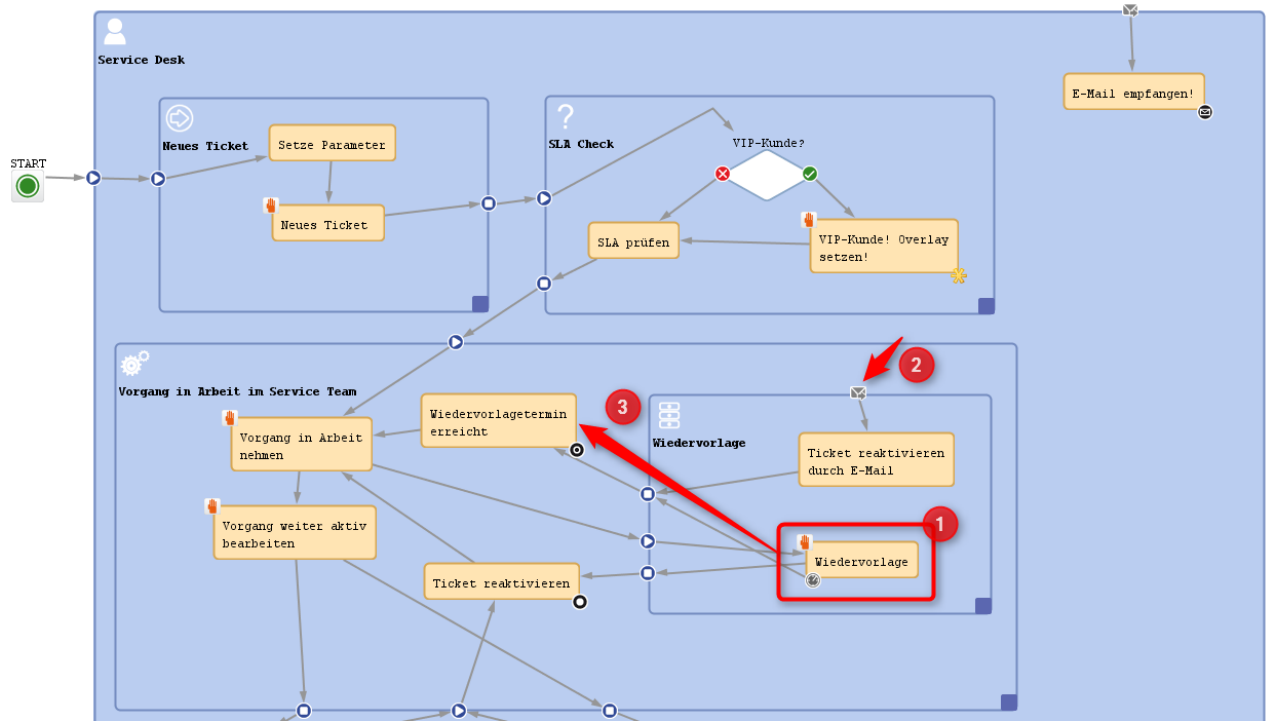


Fig. 14: ConSol*CM Process Designer - Beispiel 1: Mail-Trigger des Sub-Scopes aktiv

Beispiel 2:

Das Ticket befindet sich an Position **(1)** im Scope *Vorgang in Arbeit im Service Team*. Wenn eine E-Mail hereinkommt, feuert der Mail-Trigger des Haupt-Scopes **(2)** (da der Scope *Vorgang in Arbeit im Service Team* keinen Mail-Trigger besitzt). Daher wird die Ticketposition nicht verändert **(3)**.

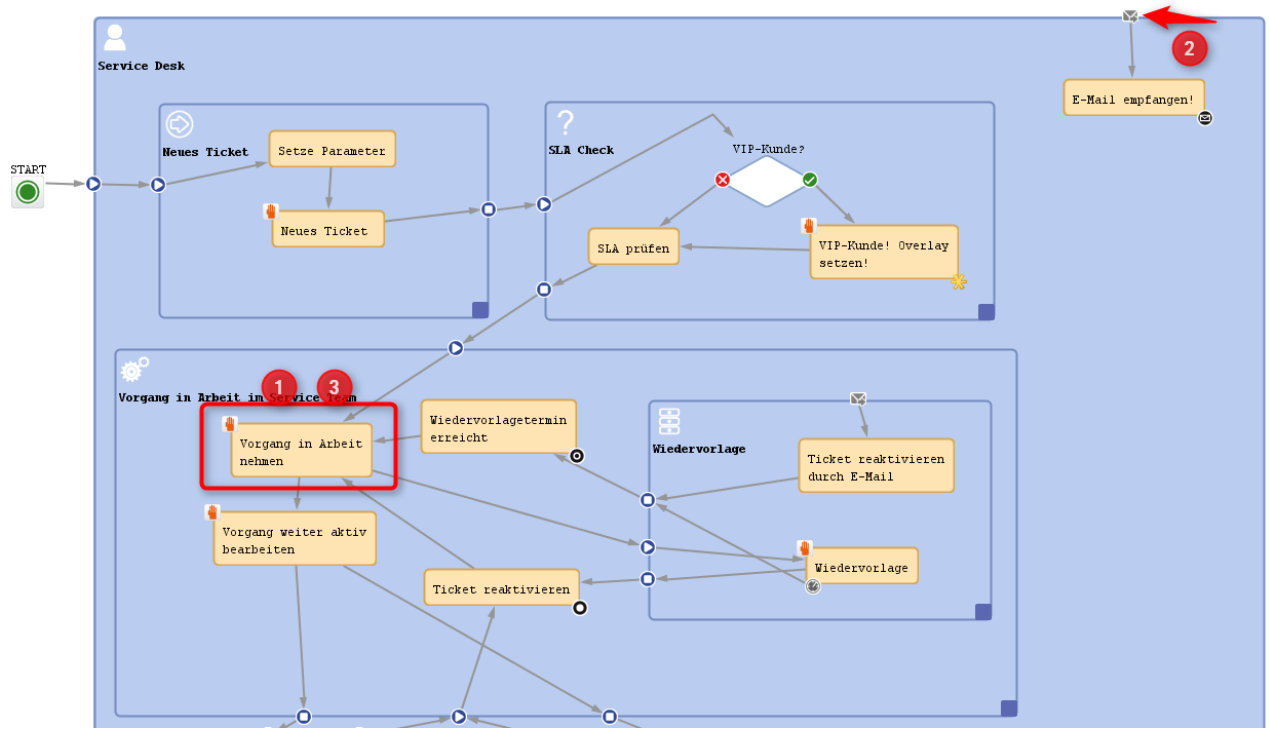


Fig. 15: ConSol*CM Process Designer - Beispiel 2: Mail-Trigger des Haupt-Scopes aktiv

4.7.4 Business-Event-Trigger

- Einleitung zu Business-Event-Triggern
- Hinzufügen eines Business-Event-Triggers zu einem Workflow
  - Hinzufügen eines Business-Event-Triggers zu einem Scope
- Eigenschaften eines Business-Event-Triggers
- Business-Logik von Business-Event-Triggern
  - Feuer-Reihenfolge von Business-Event-Triggern in serieller Reihenfolge
  - Feuer-Reihenfolge von Business-Event-Triggern in hierarchischen Scopes
    - Fall 1
    - Fall 2
    - Fall 3
- Beispiele für Business-Event-Trigger
  - Anwendungsfall 1: Überprüfe Kommentar des Bearbeiters
  - Anwendungsfall 2: Berechne die Ticketpriorität neu, wenn Auswirkung und/oder Dringlichkeit geändert wurden
  - Anwendungsfall 3: Setze den Auslieferungsprozess fort, wenn die Warenlieferung der Bestellung eingetroffen ist
- Best Practices: Verwendung von Business-Event-Triggern

Einleitung zu Business-Event-Triggern

In Business-Prozessen kommt es häufig während des regulären Prozesses zu Ereignissen, um die sich ein Mitarbeiter kümmern muss. Zum Beispiel kann es notwendig sein, den Teamleiter zu informieren, wenn jemand die Ticket-Priorität auf *Besonders Hoch* stellt. Oder es kann nach einem Bearbeiterwechsel notwendig sein zu überprüfen, ob der Bearbeiter eingeloggt ist (wenn sie oder er nicht eingeloggt ist, muss das Ticket an einen anderen Bearbeiter übergeben werden). Es gibt im Arbeitsalltag eine Vielzahl von Beispielen für solche Ereignisse.

Fig. 1: ConSol*CM Process Designer - Business-Event-Trigger

ConSol*CM kann durch Business-Event-Trigger Ereignisse registrieren und auf die folgenden Typen von Ereignissen reagieren:

- Änderung des Bearbeiters
- Änderung der Queue
- Änderung des Ticket-Themas
- Änderung des oder der zusätzlichen Bearbeiter
- Änderung (Hinzufügen) eines Kommentars d.h. Text-Kommentar oder einer E-Mail

- Änderung eines beliebigen Benutzerdefinierten Feldes, welches vom CM-System-Entwickler definiert wurde
  (dies kann z.b. die Priorität, eine Kategorie oder der Inhalt einer bestimmten Textbox sein)

Wenn das Ereignis eintritt, feuert der Business-Event-Trigger.

> **Information:**
>
> Sie als Workflow-Entwickler müssen alles, das als Konsequenz auf das Feuern des Business-Event-Triggers erfolgen soll, implementieren! Es gibt keine automatischen Aktionen. Der Business-Event-Trigger gibt lediglich das Signal *Ereignis eingetreten*.

Ein Workflow kann so viele Business-Event-Trigger wie benötigt enthalten. Sie müssen allerdings sicherstellen, dass es allen Business-Event-Triggern im Prozess potentiell möglich ist zu feuern (und dass keiner von einer Aktion abhängt, die niemals geschehen kann, da ein anderer Business-Event-(oder Zeit-) Trigger bereits vorher feuert). Bitte lesen Sie für mehr Informationen dazu den Abschnitt Prozesslogik.

Hinzufügen eines Business-Event-Triggers zu einem Workflow

Business-Event-Trigger können nur mit Scopes verbunden werden, niemals mit Aktivitäten.

Hinzufügen eines Business-Event-Triggers zu einem Scope

Klicken Sie auf das Business-Event-Trigger-Icon in der Palette und ziehen Sie den Trigger in den gewünschten Scope. Er wird automatisch an den oberen Rand des Scopes hinzugefügt. Sie können seine Position danach verändern (bewegen Sie ihn nach links oder rechts, um die Reihenfolge der Trigger zu verändern oder einfach nur, um das Layout zu verbessern).

Ein Business-Event-Trigger, der mit einem Scope verbunden wurde, kann nicht in einen anderen Scope verschoben werden. Wenn Sie den Business-Event-Trigger mit einem anderen Scope verbinden möchten, entfernen Sie den Trigger, den Sie definiert haben und erstellen Sie einen neuen für den gewünschten Scope.

Um die Eigenschaften des Triggers zu konfigurieren, wählen Sie ihn im Arbeitsbereich aus und setzen die gewünschten Werte im Eigenschaften-Editor. Siehe dazu den folgenden Abschnitt Eigenschaften eines Business-Event-Triggers.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach einem Business-Event-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

Eigenschaften eines Business-Event-Triggers

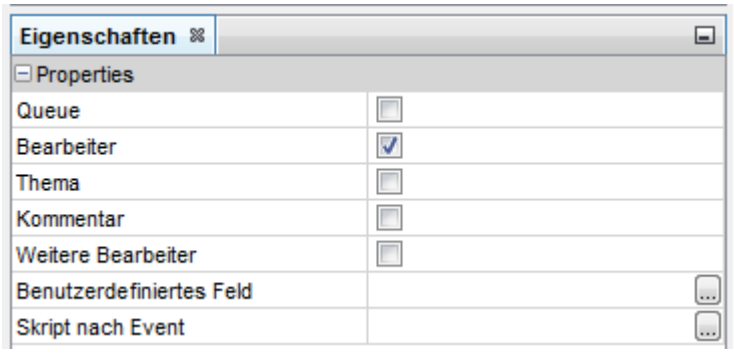


Fig. 2: ConSol*CM Process Designer - Eigenschaften eines Business-Event-Triggers

Ein Business-Event-Trigger besitzt die folgenden Eigenschaften

- **Queue**
  Aktivieren Sie diese Checkbox, wenn der Business-Event-Trigger auf den Wechsel der Queue reagieren soll, d.h. der Trigger feuert, wenn das Ticket in eine andere Queue verschoben wird. Es ist nicht relevant, ob dies aufgrund einer manuellen Aktion oder automatisch durch das System geschehen ist.
- **Bearbeiter**
  Aktivieren Sie diese Checkbox, wenn der Business-Event-Trigger auf den Wechsel des Bearbeiters des Tickets (Ticket-Owner) reagieren soll. Dies kann eine manuelle oder automatische Aktion sein. Es gibt drei mögliche Konstellationen:
  - Das Ticket besaß noch keinen Bearbeiter und es wird ein Bearbeiter gesetzt.
  - Das Ticket besitzt einen Bearbeiter und das Ticket wird an einen anderen Bearbeiter übergeben.
  - Das Ticket besitzt einen Bearbeiter und der Bearbeiter wird auf *null* gesetzt (kein Bearbeiter).
- **Thema**
  Aktivieren Sie diese Checkbox, wenn der Business-Event-Trigger auf eine Veränderung des Ticket-Themas reagieren soll.
- **Kommentar**
  Aktivieren Sie diese Checkbox, wenn der Business-Event-Trigger auf die Veränderung eines Kommentar reagieren soll, d.h.:
  - Ein Bearbeiter hat einen neuen (Text-)Kommentar hinzugefügt.
  - Ein Kunde hat einen neuen (Text-)Kommentar mittels seines ConSol*CM/Track-Zugangs hinzugefügt.
  - Eine E-Mail wurde für das Ticket empfangen.
  - Eine E-Mail wurde vom Ticket aus gesendet.
  - Ein oder mehrere Attachment(s) wurden zum Ticket hinzugefügt.

- **Weiterer Bearbeiter**
  Aktivieren Sie diese Checkbox, wenn das Ticket auf die Veränderung von zusätzlichen Bearbeitern, die dem Ticket in einer bestimmten Bearbeiterfunktion zugewiesen sind, reagieren soll (Ticketbereich *Bearbeiter*). Dabei kann es sich um eine der folgenden Situationen handeln (manuell oder automatisch vom System gesetzt):
  - Das Ticket besaß noch keinen zusätzlichen Bearbeiter und ein oder mehrere zusätzliche Bearbeiter wird/werden gesetzt.
  - Das Ticket besitzt einen oder mehrere zusätzliche Bearbeiter und einer oder mehrere von ihnen wird/werden auf *null* gesetzt oder es wird/werden dessen/deren Name(n) geändert.
  - Das Ticket besitzt einen oder mehrere zusätzliche Bearbeiter und alle diese Bearbeiter werden auf *null* gesetzt (kein Bearbeiter).

- **Benutzerdefiniertes Feld**
  Mit dem (...)-Button öffnen Sie das Pop-Up-Fenster *Event Trigger* (siehe nächstes Bild), in dem Sie das/die Benutzerdefinierte(n) Feld(er) auswählen können, welche überwacht werden sollen. Verwenden Sie die *Plus*- und *Minus*-Buttons, um mehr Felder hinzuzufügen oder die Anzahl der zu überwachenden Felder zu reduzieren. Wie auch bei der Definition der Benutzerdefinierten Felder (siehe *ConSol*CM Administratorhandbuch,* Abschnitt *Verwaltung von Benutzerdefinierten Feldern*), müssen Sie zuerst die Benutzerdefinierte Feldgruppe aus dem linken Drop-Down-Menü auswählen und können dann eines der Benutzerdefinierten Felder aus dieser Gruppe aus dem rechten Drop-Down-Menü auswählen. Sie können so viele Benutzerdefinierte Felder auswählen, wie Sie möchten.

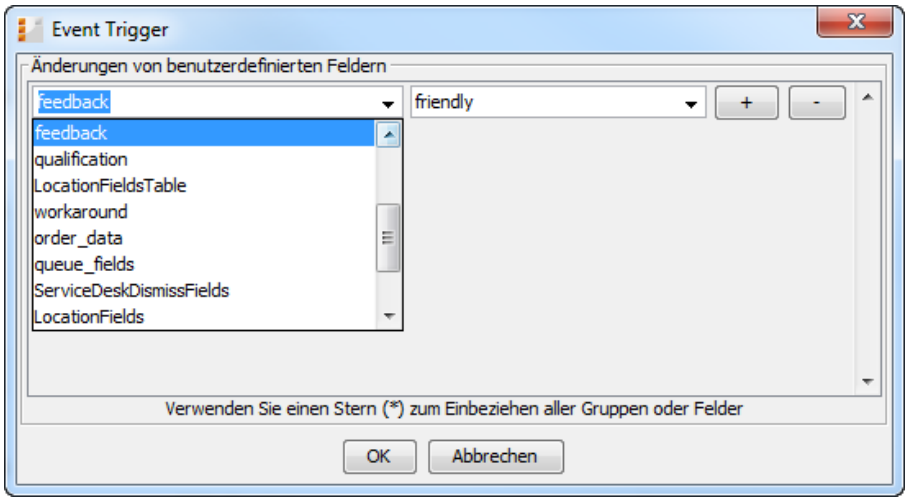


Fig. 3: ConSol*CM Process Designer - Parameter "Benutzerdefiniertes Feld" eines Business-Event-Triggers

Sie können mehrere Events für einen Business Event Trigger definieren, auf die dieser Trigger reagieren soll. Diese (potenziellen) Events werden mit ODER verbunden. Beispielsweise würde ein Trigger feuern, wenn sich der Bearbeiter ändert, ODER wenn das Thema des Tickets geändert wird.

- **Skript nach Event**
  Hier können Sie ein Skript definieren (mittels des ConSol*CM Skript-Editors), das ausgeführt wird, wenn der Business-Event-Trigger gefeuert hat. Es muss *true* oder *false* zurückgeben. Wenn es *true* zurückgibt, wird der Event wirklich gefeuert, d.h. die automatische Aktivität hinter dem Business-Event-Trigger wird ausgeführt. Wenn des Skript *false* zurückgibt, wird das Event blockiert und die automatische Aktivität wird nicht ausgeführt. Auf diese Weise können Sie exakt kontrollieren, wann die Aktion (Aktivität) ausgeführt werden soll, z.B. reagiert der Trigger auf die Änderung der Priorität, soll aber nur feuern, wenn die neue Priorität *Besonders Hoch* ist. Das Skript überprüft dann die neue Priorität und nur wenn der neue Wert *Besonders Hoch* ist, gibt das Skript *true* zurück, für alle anderen Werte gibt es *false* zurück.

> **Vorsicht:**
>
> Das *Skript nach Event* wird nur zur Steuerung und zur Feinabstimmung des Feuerns des Business-Event-Triggers verwendet! Jede Aktion, die ausgeführt werden soll, wenn der Trigger gefeuert hat, muss als automatische Aktivität hinter den Trigger gesetzt werden! Dies garantiert eine gute Prozesslogik und hilft dabei, den Prozess im Process Designer zu visualisieren!

Business-Logik von Business-Event-Triggern

Feuer-Reihenfolge von Business-Event-Triggern in serieller Reihenfolge

Wenn ein Ereignis eingetreten ist, das relevant für einen Business-Event-Trigger ist, feuert der Trigger. Danach wird das *Skript nach Event* ausgeführt. Wenn es *true* zurückgibt, wird die folgende automatische Aktivität oder der Entscheidungsknoten mit einer der beiden folgenden automatischen Aktivitäten ausgeführt.

Wenn der Bearbeiter mehr als einen Ticket-Parameter verändert und verschiedene Business-Event-Trigger für diese Parameter in diesem Scope definiert wurden, feuern die Business-Event-Trigger entsprechend ihrer Reihenfolge am Scope.

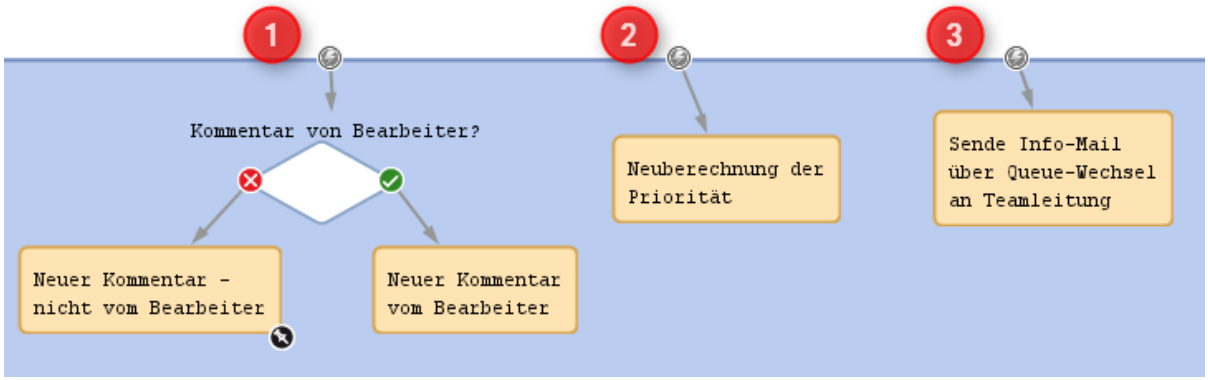


Fig. 4: ConSol*CM Process Designer - Feuer-Reihenfolge von Business-Event-Triggern (1)

Wenn einer der Business-Event-Trigger das Ticket zu einer neuen Position führt (d.h. es sich nicht länger in dem Scope befindet, an dem sich der nächste Business-Event-Trigger befinden würde), feuern die folgenden Business-Event-Trigger nicht. Im Beispiel im nächsten Bild wird Business-Event-Trigger **(3)** nicht feuern, wenn der *Neuberechnung der Priorität*-Trigger **(2)** gefeuert hat (siehe *Anwendungsfall 2* im Abschnitt Beispiele für Business-Event-Trigger), da die vorhergehende Aktion das Ticket in einen anderen Scope geführt hat.

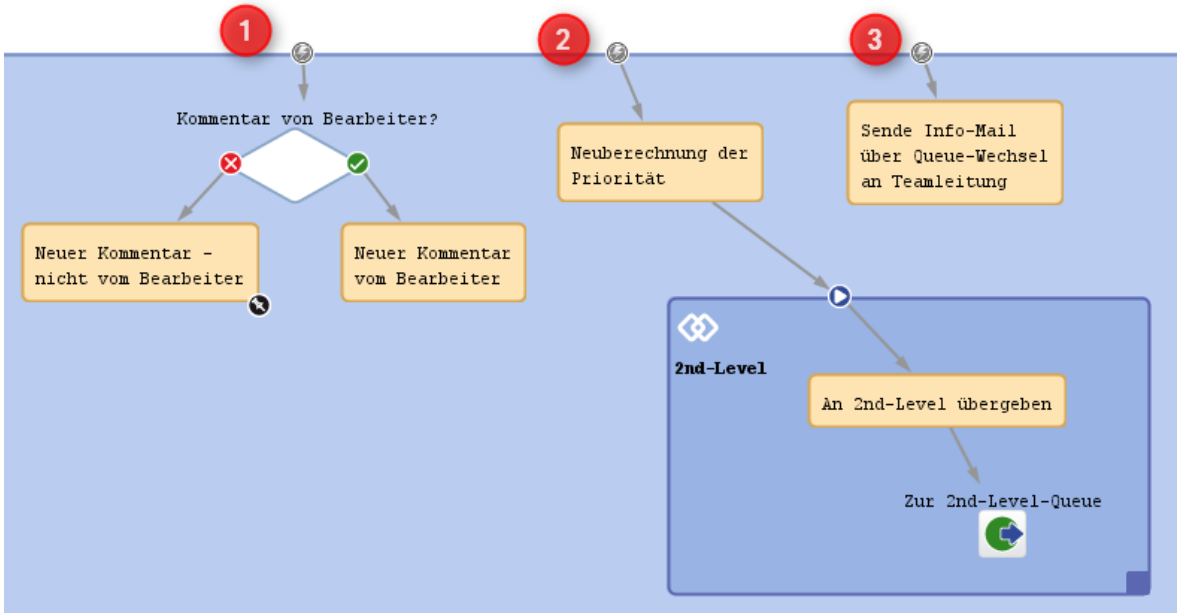


Fig. 5: ConSol*CM Process Designer - Feuer-Reihenfolge von Business-Event-Triggern (2)

Feuer-Reihenfolge von Business-Event-Triggern in hierarchischen Scopes

Wenn sich Business-Event-Trigger in hierarchischen Scopes befinden, wird das Event vom innersten Business-Event-Trigger *verbraucht*, d.h. vom Business-Event-Trigger des innersten Scopes. Alle Events, die dort nicht verbraucht wurden, werden in denn nächsten umgebenden Scope weitergeleitet, danach den diesen umgebenden usw.

Fall 1

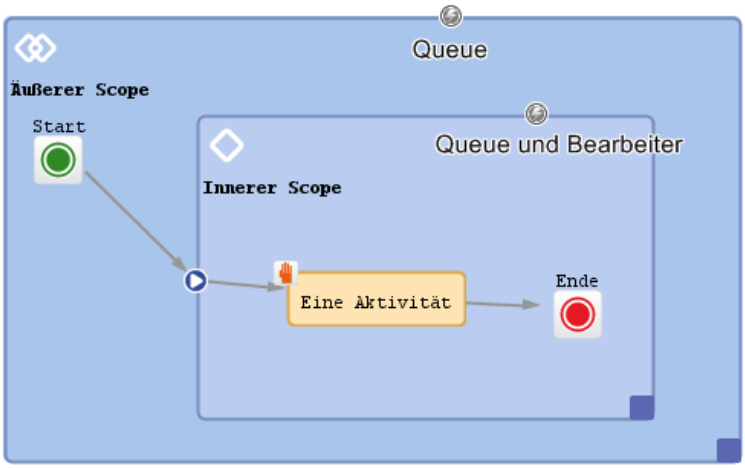


Fig. 6: ConSol*CM Process Designer - Hierarchische Business-Event-Trigger (1)

Gefeuerte Events:

Events	Trigger gefeuert
Queue	Innerer
Queue und Bearbeiter	Innerer für beide
Bearbeiter	Innerer

Fall 2

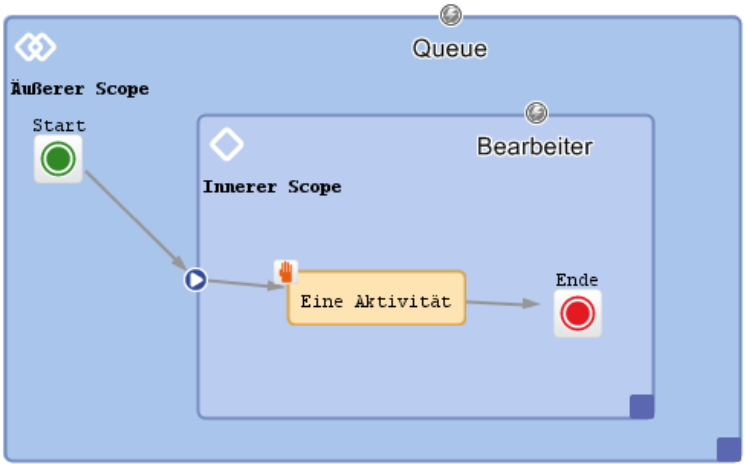


Fig. 7: ConSol*CM Process Designer - Hierarchische Business-Event-Trigger (2)

Gefeuerte Events:

Events	Gefeuerte Trigger
Queue	Äußerer
Bearbeiter	Innerer
Queue und Bearbeiter	Innerer und Äußerer

Fall 3

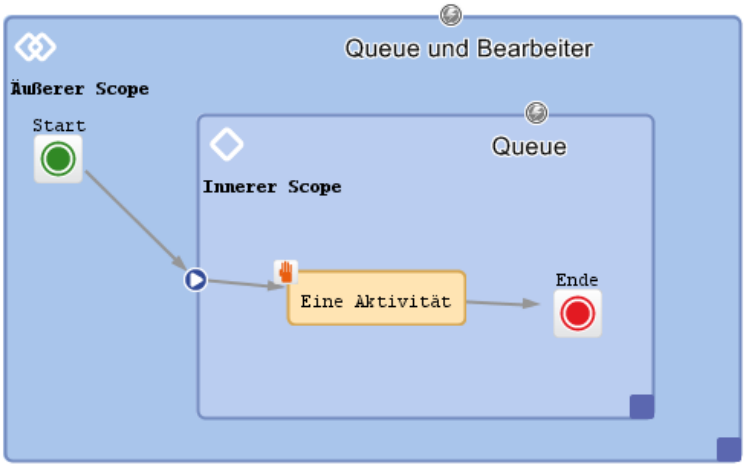


Fig. 8: ConSol*CM Process Designer - Hierarchische Business-Event-Trigger (3)

Gefeuerte Events:

Events	Gefeuerte Trigger
Queue	Innerer
Bearbeiter	Äußerer
Queue und Bearbeiter	Innerer (Queue) und Äußerer (nur Bearbeiter)

Beispiele für Business-Event-Trigger

Anwendungsfall 1: Überprüfe Kommentar des Bearbeiters

Wenn ein neuer Kommentar von jemand anderem als dem derzeitigen Bearbeiter zum Ticket hinzugefügt wurde, soll ein Overlay zum Ticket-Icon hinzugefügt werden. Auf diese Weise wird das Ticket markiert und der Bearbeiter kann in der Ticketliste sehen, dass es einen neuen Kommentar zu einem seiner Tickets gibt. Der Kommentar kann hinzugefügt werden von einem anderen Bearbeiter, der Schreibberechtigungen für die Queue besitzt oder von einem Kunden, der Kommentare mittels seines ConSol*CM/Track-Zugangs hinzufügen kann. Oder eine E-Mail könnte empfangen worden sein.

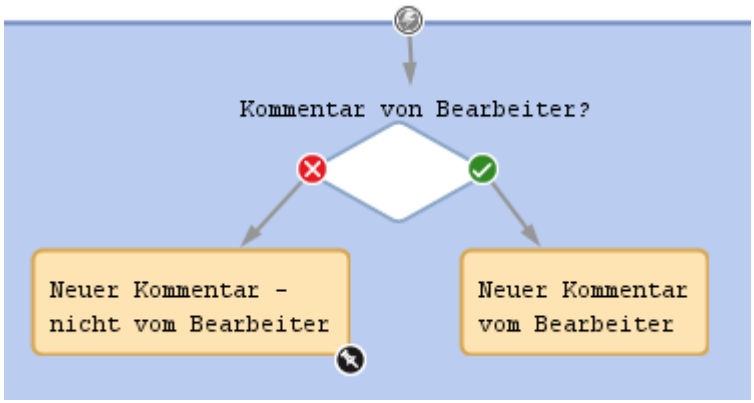


Fig. 9: ConSol*CM Process Designer - Business-Event-Trigger mit Folgeaktivitäten

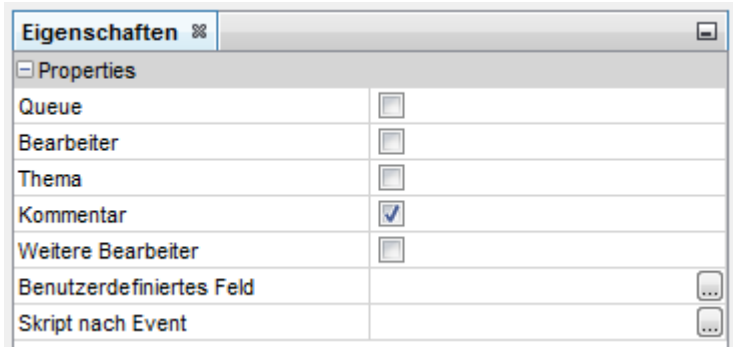


Fig. 10: ConSol*CM Process Designer - Eigenschaften eines Business-Event-Triggers (1)

Code des Skripts des Entscheidungsknotens

```
return (workflowApi.getCurrentEngineer() == ticket.getEngineer())
```

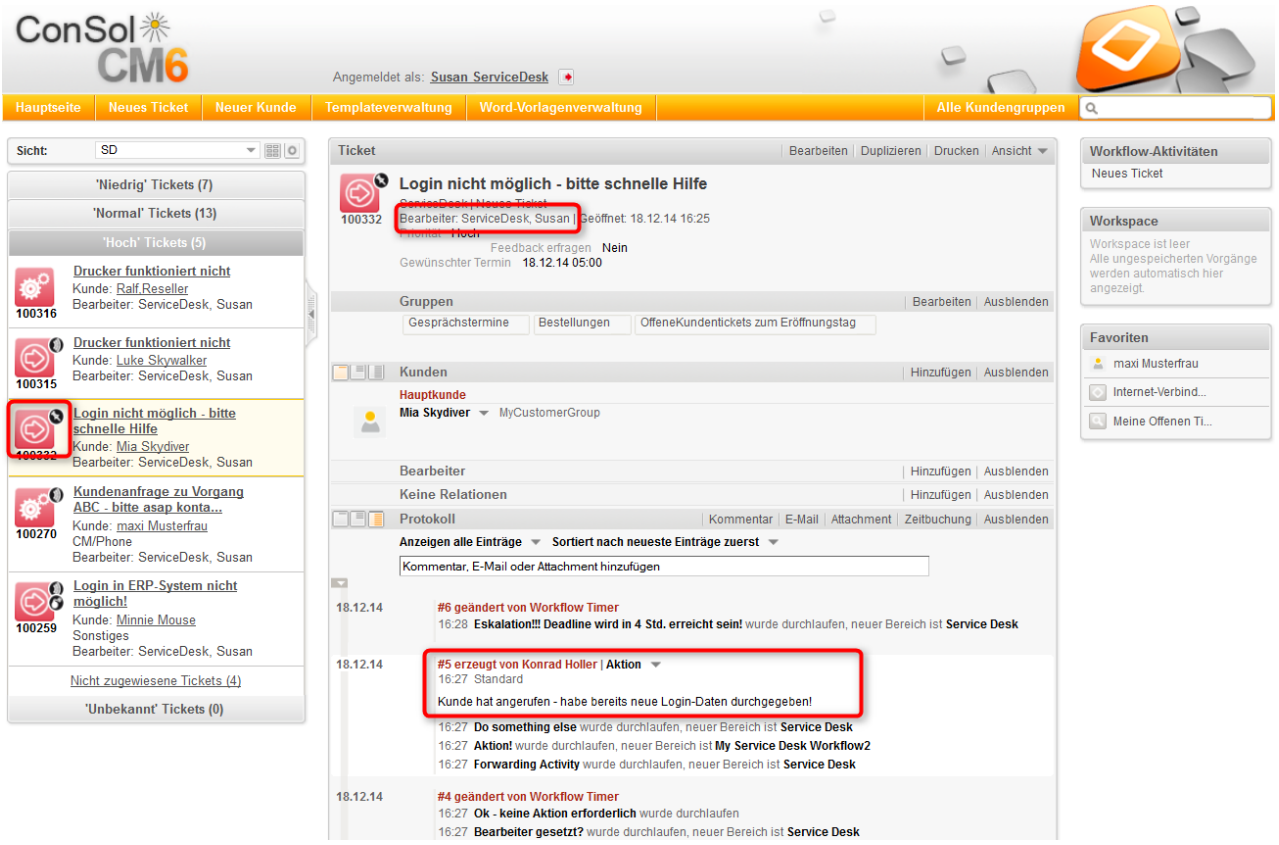


Fig. 11: ConSol*CM Web Client- Ticket markiert mit neuem Overlay

Anwendungsfall 2: Berechne die Ticketpriorität neu, wenn Auswirkung und/oder Dringlichkeit geändert wurden

Dies ist ein Beispiel aus einer *ITIL-Servicedesk*-Umgebung. Entsprechend der *ITIL*-Standards wird die Ticketpriorität aus zwei Werten berechnet: *Auswirkung* und *Dringlichkeit*. Das bedeutet, dass es im Ticket zwei Felder gibt, die vom Bearbeiter modifiziert werden können, und die Priorität wird automatisch aus diesen zwei Werten berechnet. Die Priorität kann dann als Ticketfarbe oder als Read-Only-Liste (oder beides) dargestellt werden.

Dieses Prinzip erfordert die Neuberechnung der Priorität, wenn mindestens eines der beiden Felder (Auswirkung/Dringlichkeit) geändert wurde. Dies wird durch einen Business-Event-Trigger mit einer benachbarten Aktivität, bei der die Neuberechnung durchgeführt wird, erreicht.

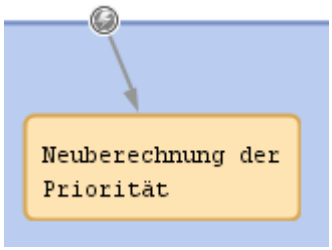


Fig. 12: ConSol*CM Process Designer - Business-Event-Trigger mit automatischer Folgeaktivität

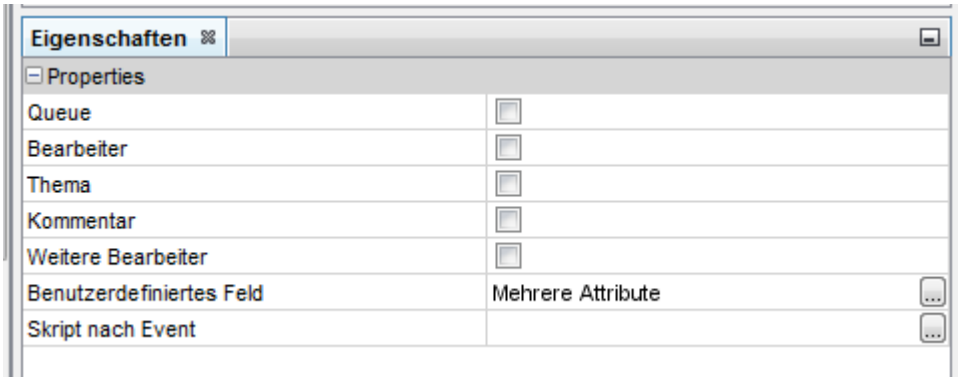


Fig. 13: ConSol*CM Process Designer - Eigenschaften eines Business-Event-Triggers (2)

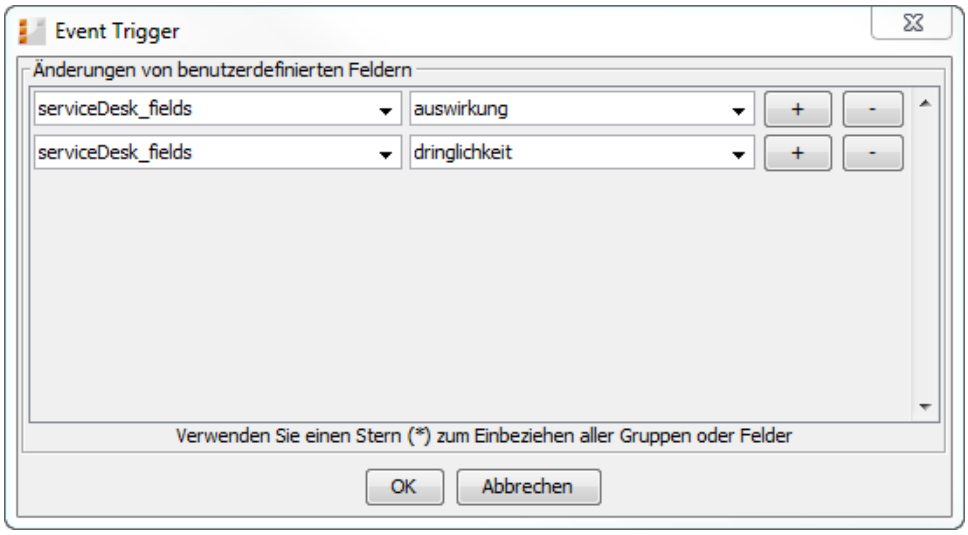


Fig. 14: ConSol*CM Process Designer - Parameter "Benutzerdefinertes Feld" eines Business-Event-Triggers (2)

Code des Skripts der automatischen Aktivität "Neuberechnung der Priorität"

```
// Neuberechnung der Prioritaet:
String imp_value = ticket.get("service_desk_fields.impact").getName()
String urg_value = ticket.get("service_desk_fields.urgency").getName();
ScriptProvider scriptProvider = scriptProviderService.createDatabaseProvider("calculatePriority.
groovy")
//Inhalt des calculatePriority.groovy wurde hier ausgelassen, da es für den aktuellen Kontext
nicht relevant ist
```

Anwendungsfall 3: Setze den Auslieferungsprozess fort, wenn die Warenlieferung der Bestellung eingetroffen ist

Dies ist ein Beispiel aus einem Versand- und Lieferungsprozess: Neue Komponenten (z.B. Hardware) werden bestellt. Das Ticket wartet im Scope *Bestellung: Auf Lieferung warten*. Wenn die Warenlieferung eingetroffen ist, registriert ein Bearbeiter eines anderes Teams den Wareneingang und setzt den Tag *Ware eingegangen*. Diese Veränderung der Ticketdaten (*Ware eingegangen* von *false* auf *true*) wird vom Business-Event-Trigger registriert, der auf den entsprechenden *boolean* Wert reagiert (die Checkbox). Nachdem der Business-Event-Trigger gefeuert hat , wird die Checkbox kontrolliert (im Entscheidungsknoten) und wenn der Wert auf *true* gesetzt ist, wird das Ticket in den nächsten Scope *Lieferung Komponenten* weitergeleitet. Die Bearbeiter, die für die Auslieferung sehen nun das Ticket in ihrer Sicht *Für Auslieferung bereite Komponenten* und können der Auslieferung nachdem sie fertig sind mit *Alle Komponenten geliefert* bestätigen.

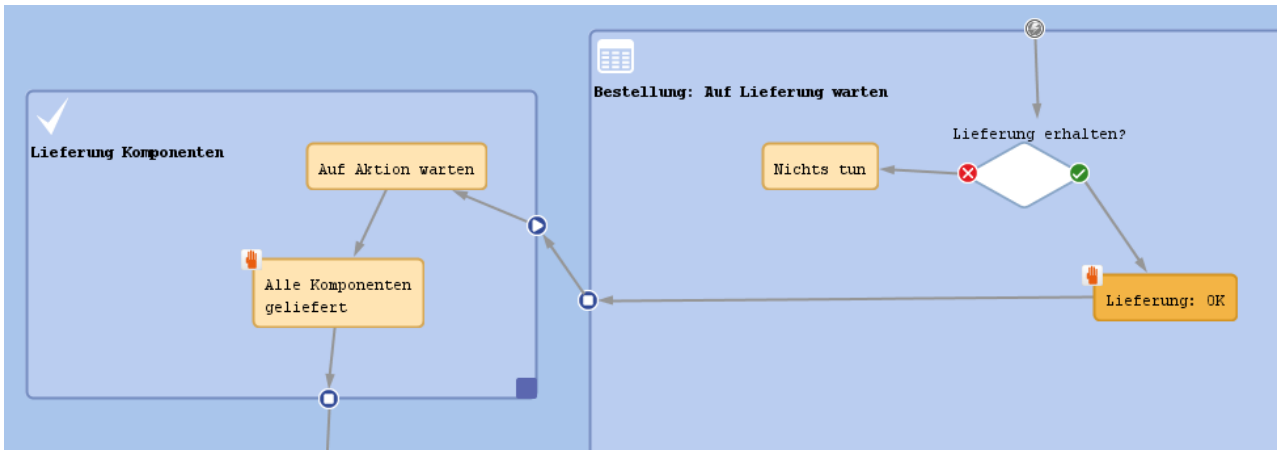


Fig. 15: ConSol*CM Process Designer - Workflow für Anwendungsfall 3

Best Practices: Verwendung von Business-Event-Triggern

Siehe Abschnitt Best Practices - Vermeiden Sie selbstauslösende Business-Event-Trigger.

4.7.5 Aktivitätsformulare (ACFs)

- Einleitung zu ACFs
- Hinzufügen eines ACF zu einem Workflow
  - Variante A: Die ACF-Definition mit dem Admin-Tool beginnen
  - Variante B: Die ACF-Definition mit dem Process Designer beginnen
- Eigenschaften eines ACF
- Business-Logik von ACFs
  - ACF an einer manuellen Aktivität
  - ACF an einer manuellen Aktivität mit Bedingung
- Beispiel für die Verwendung von ACFs
  - Anwendungsfall 1: ACF für das Ablehnen einer Kundenanfrage
  - Anwendungsfall 2: Ausfüllen der Sales-Informationen, wenn ein Angebot erstellt wurde

Einleitung zu ACFs

Ein Aktivitätsformular (*Activity Control Form = ACF)* ist ein Webformular, welches einem Bearbeiter an einem oder mehreren Prozessschritten angezeigt wird. Ein ACF wird im Admin-Tool definiert und dann im Process Designer an eine Workflow-Aktivität angehängt. Mittels ACFs kann die Dateneingabe strikt kontrolliert werden.

Fig. 1: ConSol*CM Process Designer - Aktivitätsformular (Activity Control Form=ACF) im Process Designer

Wenn ein Helpdesk-Agent beispielsweise eine Beschwerde ablehnen möchte, kann dies nicht geschehen, ohne dass ein Grund angegeben wird. Im Prozess wird dies mit einem ACF implementiert, das angezeigt wird, wenn der Bearbeiter auf die Workflow-Aktivität *Beschwerde ablehnen* klickt. Es wird ein Formular geöffnet, in dem der Bearbeiter einer Kategorie für die Ablehnung auswählen muss und eine Textbox mit einem Kommentar befüllen kann. Oder, wenn man das Beispiel eines Sales-Prozesses betrachtet, wenn ein Bearbeiter (in diesem Fall ein Sales-Agent) auf *Termin mit potentiellem Kunden* vereinbaren klickt, wird ein Formular angezeigt, in das das Budget, die Größe der Firma des Kunden und das Produkt, für das sich der Kunde interessiert, eingetragen werden muss.

Ein ACF kann optionale Felder und Pflichtfelder enthalten.

Information:

Wir empfehlen, ein "..." hinter den Namen von Aktivitäten zu setzen, die automatisch ein ACF öffnen. Dies hilft den Bearbeitern, zwischen Aktivitäten mit ACF und einfachen Aktivitäten zu unterscheiden.

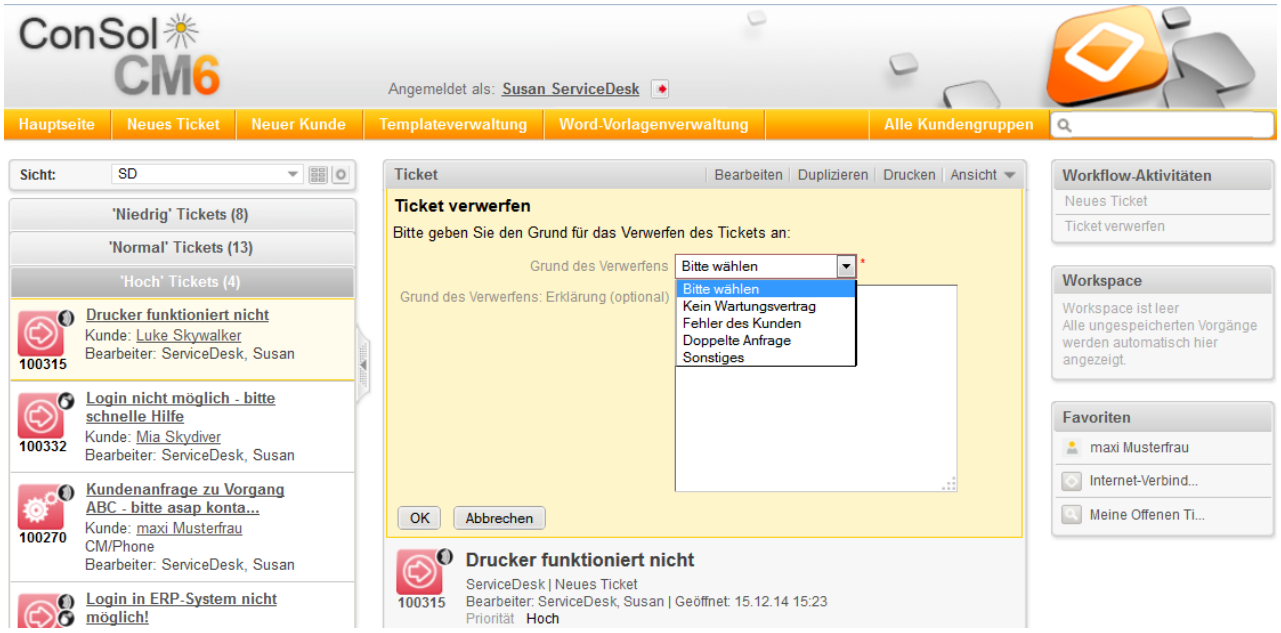


Fig. 2: ConSol*CM Web Client - Geöffnetes ACF

Hinzufügen eines ACF zu einem Workflow

Variante A: Die ACF-Definition mit dem Admin-Tool beginnen

Bevor Sie ein ACF zu einem Workflow hinzufügen können, müssen Sie es im Admin-Tool definieren. Bitte lesen Sie für eine detaillierte Erklärung das *ConSol*CM Administratorhandbuch,* Abschnitt *Verwaltung von Benutzerdefinierten Feldern.* An dieser Stelle nehmen wir an, dass Sie bereits ein ACF definiert haben und es zum Workflow hinzufügen möchten.

Ein ACF wird immer zu einer manuellen Aktivität hinzugefügt. Um ein ACF zur gewünschten Aktivität hinzuzufügen, klicken Sie auf das ACF-Icon in der Palette und ziehen Sie es in die gewünschte Aktivität. Dann können Sie die Eigenschaften des ACF konfigurieren. Wenn Sie ein ACF zu einer automatischen Aktivität hinzufügen, wird diese automatisch zu einer Aktivität vom Typ *Manuell* geändert.

Im Web Client wird das ACF geöffnet, wenn der Bearbeiter auf die Workflow-Aktivität klickt, die im Workflow mit dem ACF verbunden ist. Siehe Bild oben.

Variante B: Die ACF-Definition mit dem Process Designer beginnen

Sie können auch ein leeres ACF zu einer Workflow-Aktivität hinzufügen und währenddessen den Namen festlegen. Dadurch wird ein leeres ACF im Admin-Tool erstellt und Sie müssen in einem späteren Schritt diesem ACF die Benutzerdefinierten Felder zuweisen.

> **Vorsicht:**
>
> Vergessen Sie nicht, die Admin-Tool-Daten neu zu synchronisieren! Wenn Sie das ACF im Process Designer definiert haben, gibt es keinen automatischen Transfer der Daten zum Admin-Tool.

Eigenschaften eines ACF

Dies sind die Eigenschaften eines ACF::

- **Name**
  Der Name des ACF. Wählen Sie einen Namen aus dem Drop-Down-Menü aus. Alle ACFs, die im Admin-Tool definiert wurden, sind verfügbar.
- **Pflichtfelder**
  Öffnet ein Pop-Up-Fenster (siehe Bild unten), in dem Sie die Pflichtfelder definieren können. Die Standardeinstellung ist, dass alle ACF-Felder optional sind, d.h. wenn das Formular im Web Client geöffnet wird, kann der Bearbeiter Daten eingeben, aber auch den Prozess ohne Dateneingabe fortsetzen. Bei Pflichtfeldern kann der Prozess nur fortgesetzt werden, wenn das Pflichtfeld ausgefüllt wurde.
- **Initialisierungsskript**
  Hier können Sie ein Skript definieren, das ausgeführt wird, bevor das ACF geladen wird. Normalerweise werden solche Skripts dazu genutzt, um Standardwerte in den Benutzerdefinierten Feldern von ACFs einzustellen.
- **Vorbedingungsskript**
  Hier können Sie ein Skript definieren, das ausgeführt wird um zu bestimmen, ob das ACF angezeigt wird (Rückgabewert *true*) oder nicht (Rückgabewert *false*).

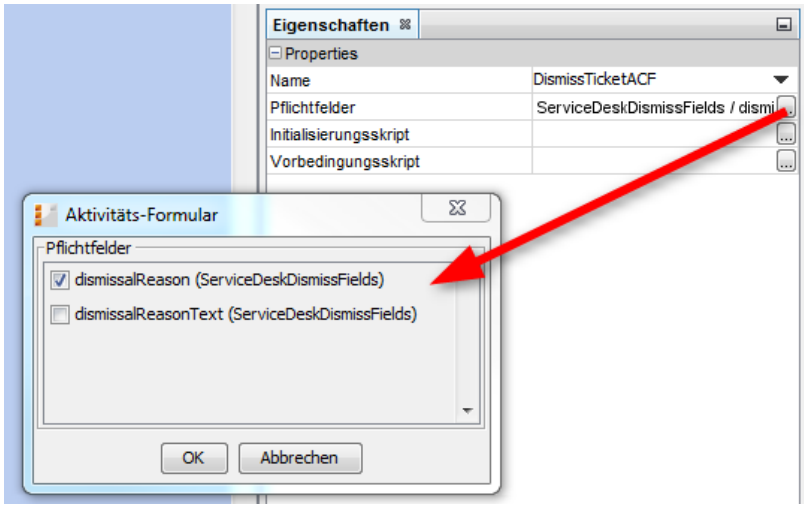


Fig. 3: ConSol*CM Process Designer - Eigenschaften eines ACF

Vorsicht:

Alle Benutzerdefinierten Felder, die Teil eines ACF sind, müssen in der Ziel-Queue verfügbar sein, d.h. die entsprechende Benutzerdefinierte Feldgruppe muss der Queue zugewiesen werden,die den Workflow benutzt! Es gibt zwei Möglichkeiten, um dies zu erreichen:

1. Sie weisen die Benutzerdefinierte Feldgruppe manuell der Queue zu.
2. Sie erstellen das ACF und benutzen es in einem Workflow. Wenn Sie den Workflow installieren, weist ConSol*CM die benötigten Gruppen automatisch den Queues zu, die den Workflows benutzen.

Für eine detaillierte Erklärung der Queue-Verwaltung lesen Sie bitte das *ConSol*CM Administratorhandbuch*.

Business-Logik von ACFs

ACF an einer manuellen Aktivität

ACFs sind nur für manuelle Aktivitäten möglich. Wenn ein Bearbeiter eine Workflow-Aktivität im Web Client auswählt, wird das ACF-Skript ausgeführt (wenn es ein solches Skript gibt). Dann wird das ACF im Web Client geöffnet (mit optionalen und Pflichtfeldern). Wenn Felder, die Teil des ACF sind, auch in den normalen Ticketdatenfeldern verfügbar sind, ist es möglich, dass diese Felder bereits von einem Bearbeiter editiert/ausgefüllt wurden, bevor das ACF verwendet wird. Daher können solche Felder bereits im ACF ausgefüllt sein. Der Bearbeiter kann diese lassen, wie sie sind (und das ACF lediglich zur Kontrolle verwenden) oder den Inhalt der Felder bearbeiten (sofern es sich nicht um read-only Felder handelt, die hier vielleicht zur Info angezeigt werden).

Wenn die Daten eines ACF nicht sichtbar sein sollen, bevor der Prozess einen bestimmten Schritt erreicht hat, können die Daten in eine (oder mehrere) separate Benutzerdefinierte Feldgruppe(n) gestellt werden, welche zu Beginn des Prozesses *unsichtbar* sind. Im Schritt nach der Aktivität, die mit dem ACF verbunden ist, werden die Benutzerdefinierten Feldgruppen mit einem Skript einer Workflow-Aktivität eingeblendet. Bitte lesen Sie für mehr Empfehlung bezüglich der Verwendung von ACFs den Abschnitt Best Practices.

Wenn ein ACF abgebrochen wird, kehrt das Ticket zu dem Scope seiner letzten Aktivität zurück, da das Ticket immer **hinter** der letzten Aktivität wartet und **nicht** vor der nächsten.

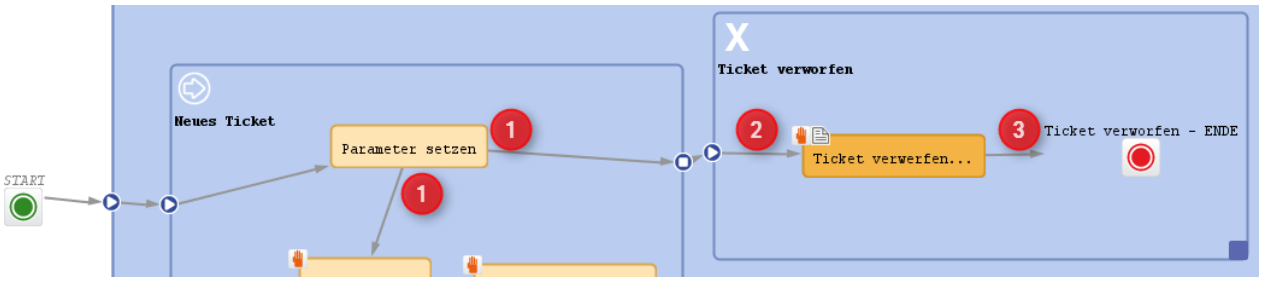


Fig. 4: ConSol*CM Process Designer - ACF Prozesslogik

Beispiel:

- Ein Ticket wird erstellt und läuft durch die automatische Aktivität *Parameter setzen*.
- Es wartet hinter dieser Aktivität an Position **(1)** im Scope *Neues Ticket*. Die nächsten Aktivitäten *Ticket verwerfen...* und *Neues IT-Ticket* (nicht im Bild) werden im Web Client angezeigt.
- Der Bearbeiter wählt *Ticket verwerfen...* .
- Das Skript für das ACF an *Ticket verwerfen...*wird ausgeführt **(2)**.
- Das ACF wird im Web Client angezeigt.
    1. **Variante 1:**
        a. Das ACF wird abgebrochen.
        b. Das Ticket geht zurück zu **(1)**.
    2. **Variante 2:**
        a. Das ACF wird ausgefüllt und bestätigt.
        b. Die Aktivität *Ticket verwerfen...* wird ausgeführt (falls es ein Skript für diese Aktivität gibt, wird dieses ausgeführt), das Ticket läuft durch den Knoten und und setzt seinen Weg fort **(3)**. In dem obigen Beispiel wird das Ticket geschlossen.

ACF an einer manuellen Aktivität mit Bedingung

Falls eine manuelle Aktivität eine Bedingung besitzt, wird die Aktivität nur angezeigt, wenn das Bedingungsskript *true* zurückgibt, d.h. das ACF wird ebenfalls nur angezeigt, wenn das Bedingungsskript *true* zurückgibt.

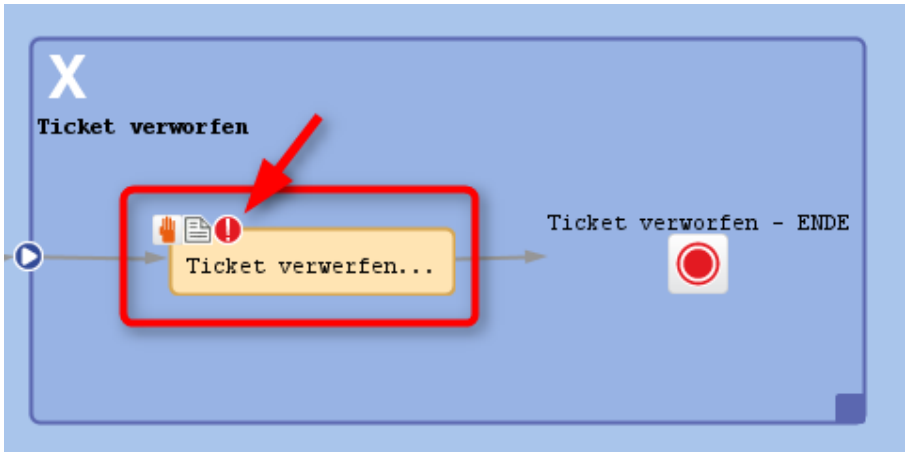


Fig. 5: ConSol*CM Process Designer - Manual Activity with ACF and Condition

Beispiel für die Verwendung von ACFs

Anwendungsfall 1: ACF für das Ablehnen einer Kundenanfrage

Dieses Beispiel wurde auch in den vorangegangenen Abschnitten als Beispiel genutzt. Der Bearbeiter kann eine Kundenanfrage nur ablehnen, wenn dafür eine Begründung gegeben wurde. Diese wird aus einem Drop-Down-Menü ausgewählt. Zusätzlich kann der Bearbeiter in ein Textfeld einen Kommentar eingeben.

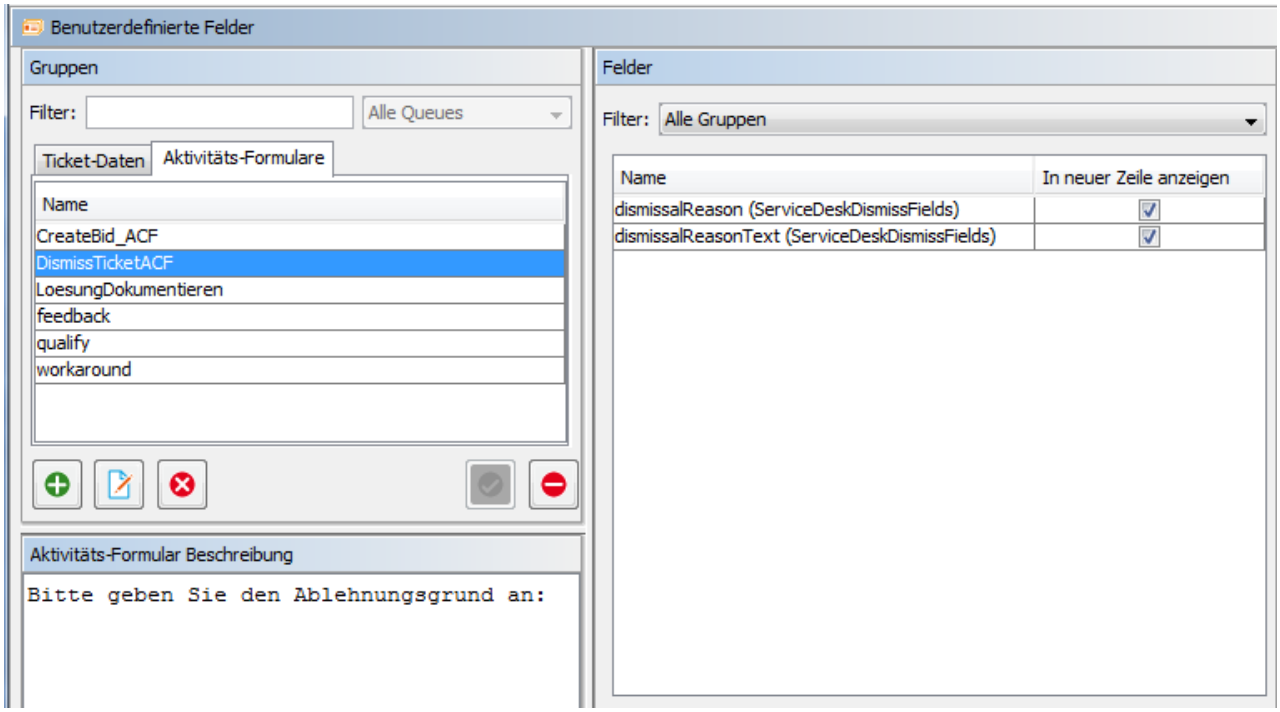


Fig. 6: ConSol*CM Admin-Tool - ACF-Definition

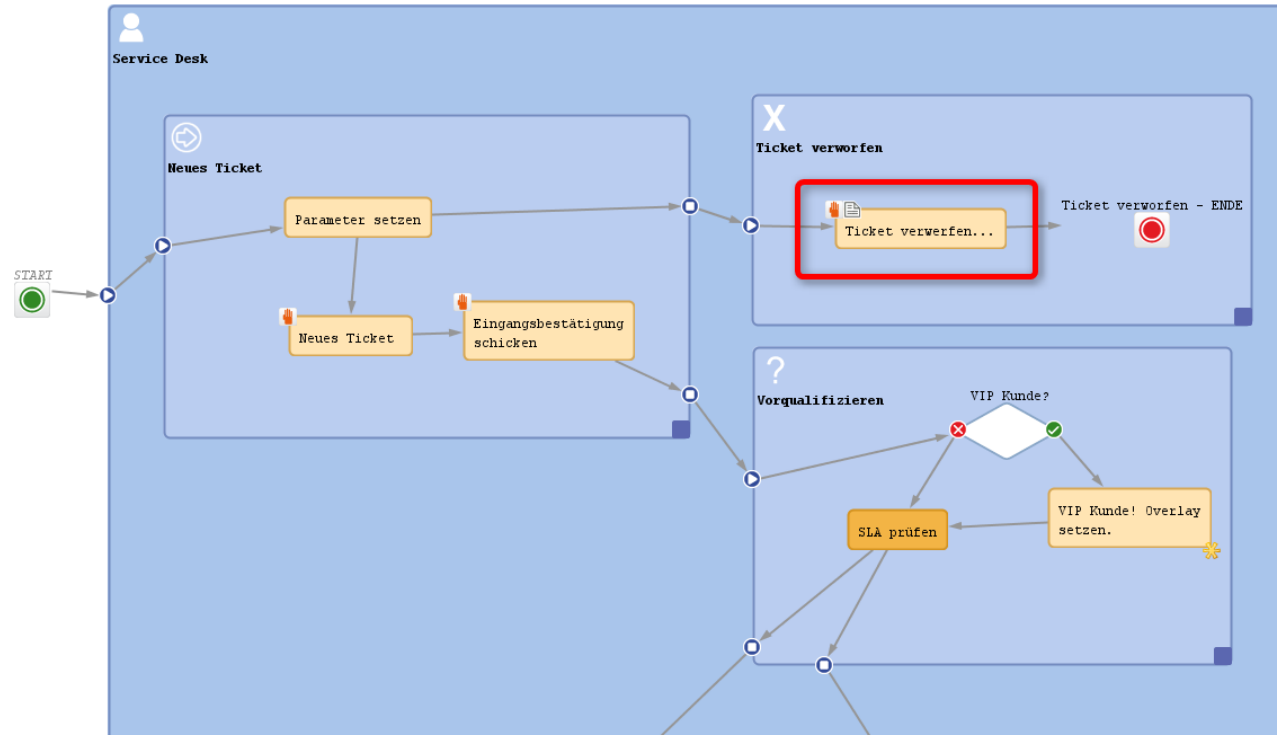


Fig. 7: ConSol*CM Process Designer - ACF im Workflow

Der Web Client und die Eigenschaften des ACF werden in den Bildern der vorangegangenen Kapitel gezeigt.

Anwendungsfall 2: Ausfüllen der Sales-Informationen, wenn ein Angebot erstellt wurde

Wenn ein Sales-Mitarbeiter die Workflow-Aktivität *Angebot erstellen...* im Web Client auswählt, öffnet sich ein ACF, in dem mehrere Felder angeboten werden. Das Feld *Initiator dieses Angebotes* ist ein Drop-Down-Menü, und ein Standardwert ("Mr. Miller") ist mittels eines ACF-Skripts gesetzt worden. Die anderen Felder sind optional. Das Feld *Produkt* wurde in den vorangegangenen Prozessschritten für das Ticket bereits gefüllt, daher wird dieses Feld mit dem ausgewählten Wert angeboten. Es kann entweder unverändert gelassen oder verändert werden.

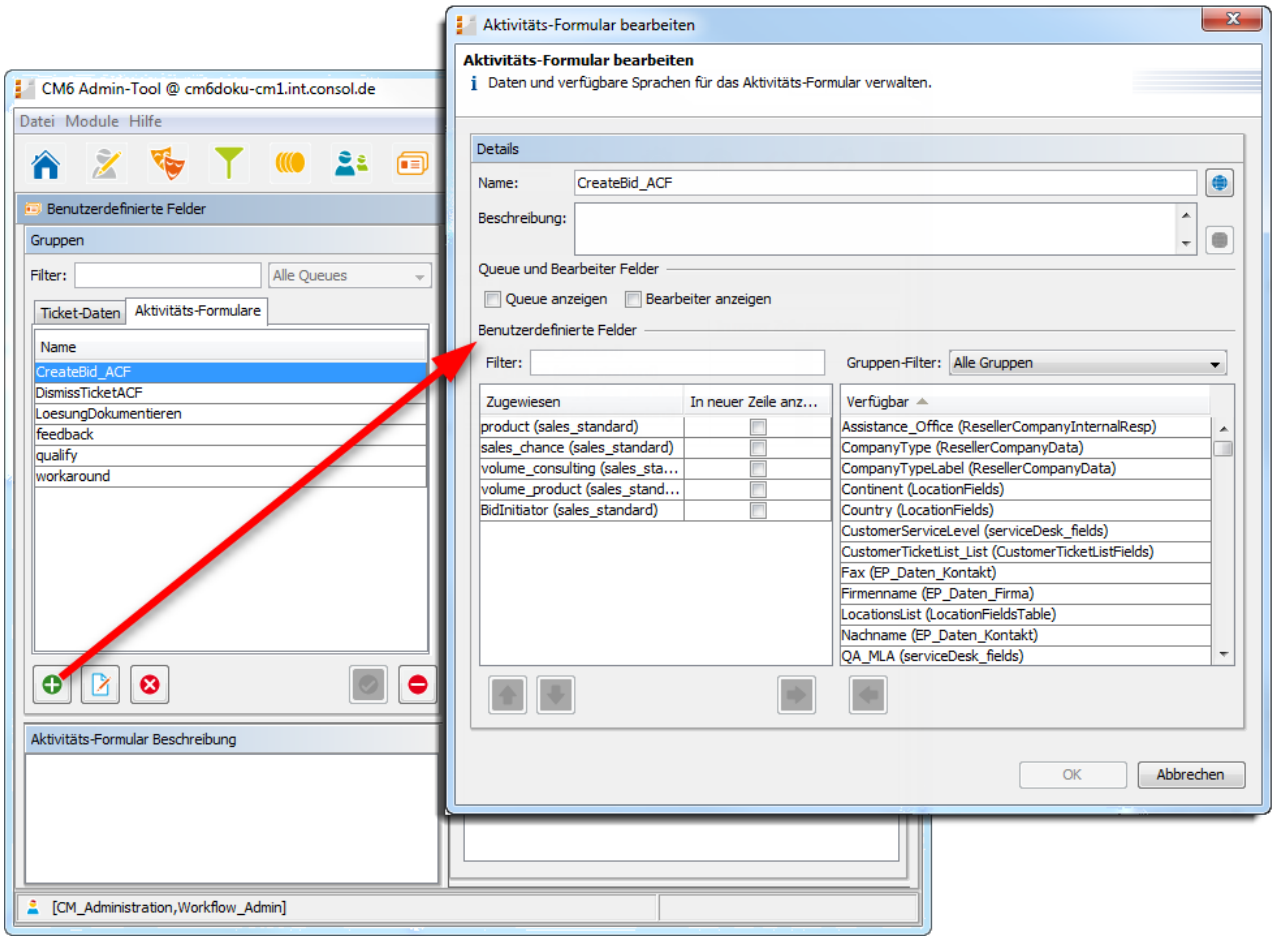


Fig. 8: ConSol*CM Admin-Tool - ACF für Sales-Workflow

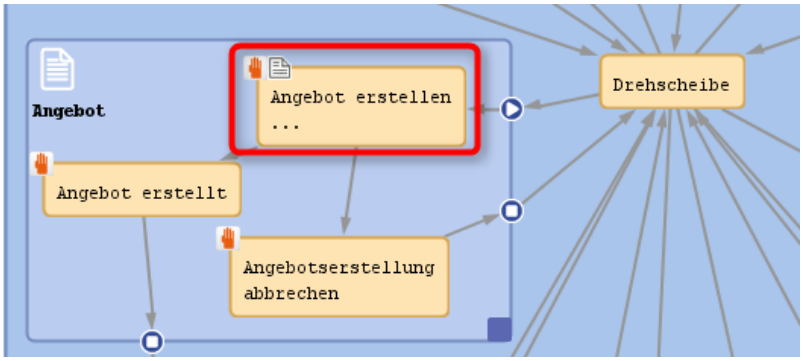


Fig. 9: ConSol*CM Process Designer - ACF für Sales-Workflow

Process Designer: Initialisierungs-Skript für ACF "Angebot erstellen"

```
ticket.set("sales_standard.BidInitiator","Mr. Miller")
```

Fig. 10: ConSol*CM/Web Client - Sales-Prozess-ACF

4.8 Aussprung- und Einsprungknoten

- Einleitung
- Aussprungknoten
    - Eigenschaften eines Aussprungknotens
- Einsprungknoten
    - Eigenschaften eines Einsprungknotens

4.8.1 Einleitung

Ein Prozess beinhaltet oft einen oder mehrere Teilprozesse, z.B. kann es in einem IT-Helpdesk ein First-Level-Team geben, das die Ticket akzeptiert und qualifiziert, ein Second-Level-Team, das einige Probleme lösen kann und ein Third-Level-Team mit Spezialisten. Wenn Sie diesen Prozess abbilden möchten, müssen Sie einen Workflow für jeden Teilprozess (First-Level, Second-Level, Third-Level) erstellen. Dann müssen die Teilprozesse miteinander verknüpft werden, damit das Übergeben eines Tickets von einem Team zum nächsten über den richtigen Weg im Prozess geschieht.

Ein Ticket kann vom First-Level zum Second-Level übergehen, darauf weiter zum Third-Level-Team, zurück zum Second-Level-Team mit einer weiteren Frage, zurück zu einem weiteren Third-Level-Team und dann zurück zum First-Level-Team, das den Kunden kontaktiert. Daher werden Verbindungen von einem Teilprozess zum nächsten benötigt, d.h. Knoten, an denen das Ticket den aktuellen Workflow verlässt, einen **Aussprungknoten**, und das Gegenstück im folgenden Workflow, den **Einsprungknoten**. Wenn ein Ticket am START-Knoten des neuen Prozesses starten soll, wird kein Einsprungknoten benötigt.

Im Process Designer werden Aussprung- und Einsprungknoten durch Drag-and-Drop in den Workflow eingefügt, indem sie aus der Palette gezogen werden und mit den anderen Workflow-Elementen verbunden werden, abhängig vom gewünschten Prozess.

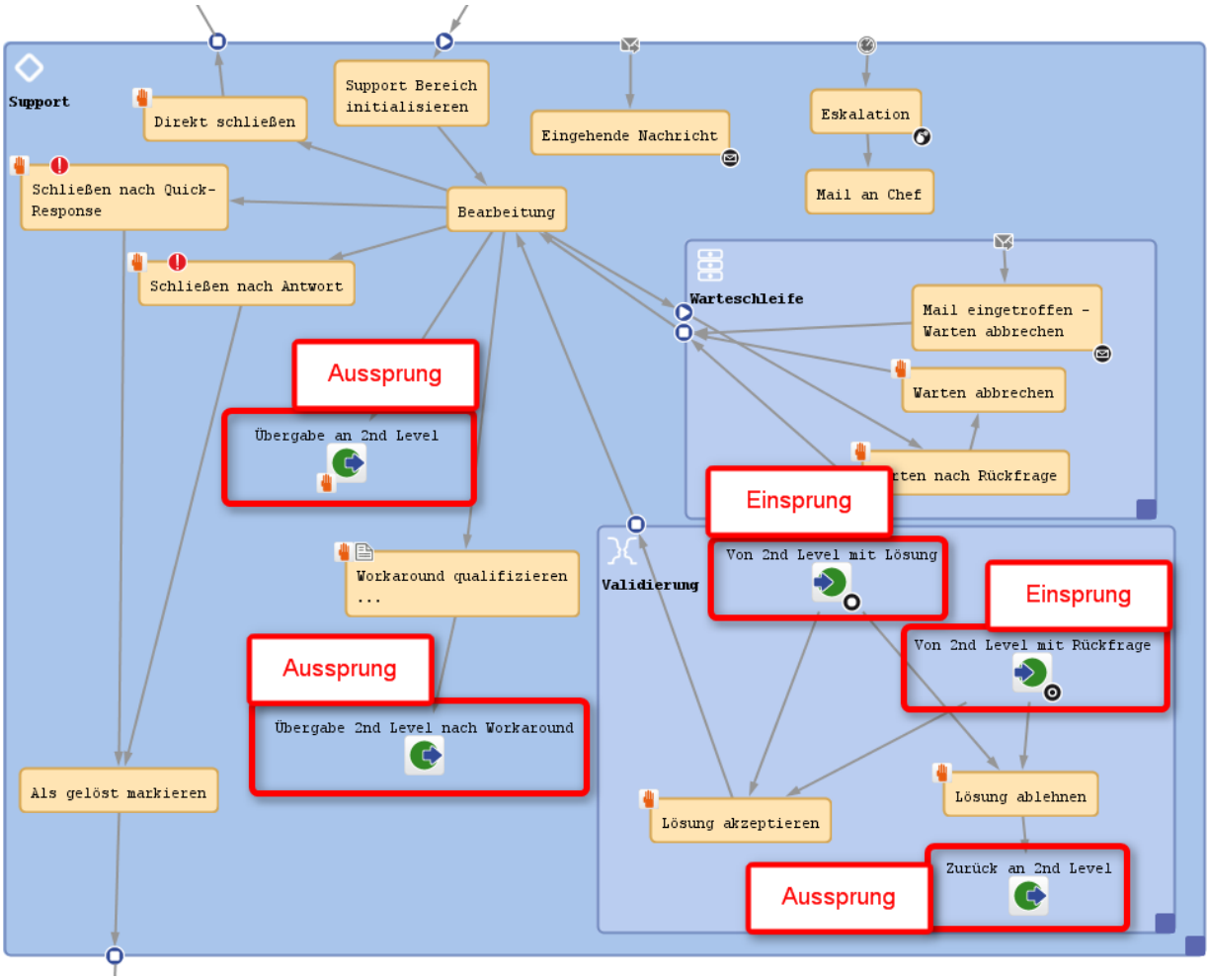


Fig. 1: ConSol*CM Process Designer - Beispiel für Aussprung und Einsprungknoten

4.8.2 Aussprungknoten

Ein Aussprungknoten definiert eine Position, an der das Ticket den (Teil-)Prozess verlassen und den nächsten (Teil-)Prozess betreten soll.

Fig. 2: ConSol*CM Process Designer - Aussprungknoten

Eigenschaften eines Aussprungknotens

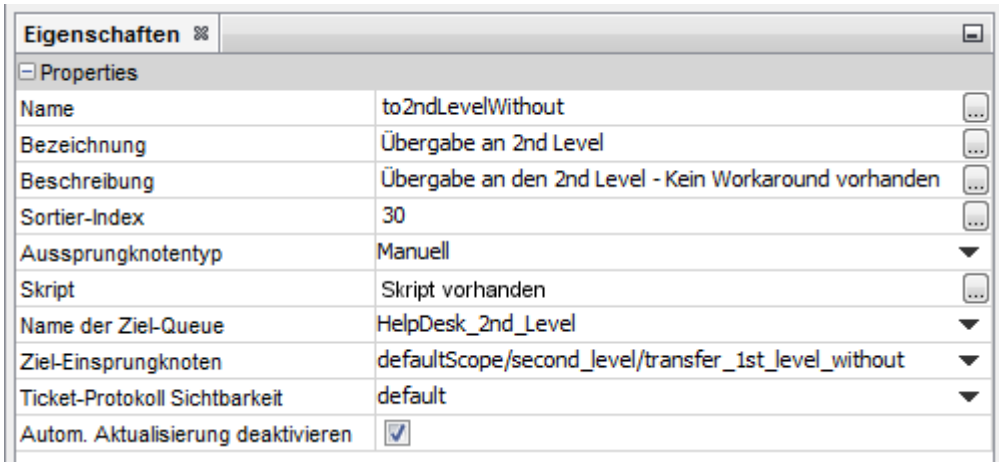


Fig. 3: ConSol*CM Process Designer - Ausprungknoten: Eigenschaften-Editor

Für einen Aussprungknoten können folgende Eigenschaften definiert werden:

- **Name**
  Notwendig. Technischer Name des Objekt.
- **Bezeichnung**
  Optional. Lokalisierter Name (wenn nicht gesetzt, wird der technische Name verwendet), der im Web Client angezeigt wird.
- **Beschreibung**
  Optional. Sie wird als Mouseover im Web Client angezeigt.
- **Sortier-Index**
  Legt die Reihenfolge der Aktivitäten im Web Client fest.
- **Aussprungknotentyp**
  Notwendig. Es muss entweder *Automatisch* oder *Manuell* ausgewählt werden. Falls es sich um einen manuellen Knoten handelt, wird er mit dem *Hand/Manuell*-Icon im Process Designer markiert.
- **Skript**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Ticket in den Knoten eintritt.
- **Name der Ziel-Queue**
  Wählen Sie den Namen der Queue aus, in welche das Ticket wechseln soll.
- **Ziel-Einsprungknoten**
  Wählen Sie den Ziel-Einsprungknoten aus dem Drop-Down-Menü aus. Alle Einsprungknoten des Workflows der gewählten Queue werden angeboten. Wenn kein Einsprungknoten ausgewählt wird, wird das Ticket den anderen Prozess, d.h. die Ziel-Queue, am START-Knoten betreten.

> **Information:**
>
> Wenn Sie mit dem Design von Workflows beginnen, stehen Sie möglicherweise vor einem *Henne-Ei-Problem*, wenn Sie beginnen, die Aussprung- und Einsprungknoten zu definieren, da Sie natürlich mit einem Workflow anfangen müssen, während der andere Workflow noch nicht existiert. Wir empfehlen, mit Dummy-Queues ohne spezifischen Einsprungknoten zu arbeiten. Fügen Sie dann den korrekten Namen der Ziel-Queue und den Namen des Einsprungknotens später hinzu.

- **Ticket-Protokoll Sichtbarkeit**
- Siehe Abschnitt Ticket-Protokoll-Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
- Siehe Abschnitt Automatische Aktualisierung deaktivieren.

4.8.3 Einsprungknoten

Ein Einsprungknoten ist ein Knoten, der die Position festlegt, an der ein Ticket aus einem anderen Prozess (Queue) in eine Queue mit dem aktuellen Workflow eintreten kann. Alle Einsprungknoten eines Workflows werden als Ziel-Einsprungknoten angeboten, wenn die Queue mit dem entsprechenden Workflow als Ziel-Queue für einen Aussprungknoten ausgewählt wurde.

Fig. 4: ConSol*CM Process Designer - Einsprungknoten

Eigenschaften eines Einsprungknotens

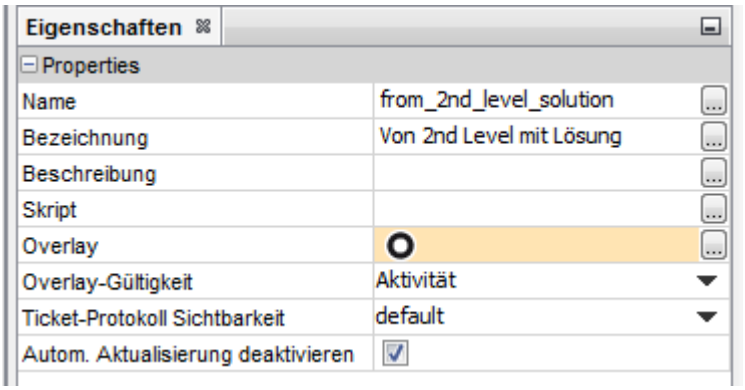


Fig. 5: ConSol*CM Process Designer - Einsprungknoten: Eigenschaften-Editor

Für einen Einsprungknoten können folgende Eigenschaften definiert werden:

- **Name**
  Notwendig. Technischer Name des Objekt.
- **Bezeichnung**
  Optional. Lokalisierter Name (wenn nicht gesetzt, wird der technische Name verwendet), der im Web Client angezeigt wird.
- **Beschreibung**
  Optional. Sie wird als Mouseover im Web Client angezeigt.
- **Skript**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Ticket in den Knoten eintritt.
- **Overlay**
  Optional. Klicken Sie in den orangen Bereich um eines der ConSol*CM Standard-Overlays laden oder benutzen Sie den Datei-Explorer (...), um ein anderes Icon aus Ihrem Dateisystem hochzuladen.
- **Overlay-Gültigkeit**
  Wird nur angezeigt, wenn ein Overlay gesetzt wurde.
  - **Aktivität**
    Das Overlay bleibt so lange am Ticket, wie das Ticket hinter der Aktivität steht. Sobald die nächste Aktivität ausgeführt wird, wird das Overlay vom Ticket entfernt.
  - **Bereich**
    Das Overlay wird entfernt, wenn das Ticket den Scope verlässt.
  - **Prozess**
    Das einmal zum Ticket-Icon hinzugefügt Overlay bleibt für den Rest des Prozesses am Ticket.
  - **Nächster Overlay**
    Das Overlay bleibt so lange am Ticket, wie kein neues Overlay hinzugefügt wird. Sobald ein neues Overlay hinzugefügt wird, wird das alte entfernt.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll-Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Automatische Aktualisierung deaktivieren.

5 Prozesslogik

- Aktivitäten
- Interrupts und Exceptions
  - Einleitung zu Interrupts und Exceptions
  - Interrupts
  - Exceptions
- Schleifen (Fehler in Workflows)
- Prozesslogik von Zeit-Triggern
- Prozesslogik von Business-Event-Triggern

Wenn Sie Workflows erstellen und bearbeiten, ist es wichtig, die Grundprinzipien der Workflow-Engine zu kennen, die für das Verhalten des Tickets während des Prozesses sorgen. Daher geben wir Ihnen einen kurzen Überblick über die Grundregeln der ConSol*CM-Ticketverarbeitung.

5.1 Aktivitäten

Grundregeln:

- Während des Durchlaufens eines Workflows wartet ein Ticket immer **hinter** der letzten Aktivität, **nicht** vor der nächsten!
- Danach schaut es nach der nächsten Aktivität, die ausgeführt/durchlaufen werden kann.
- Wenn die nächstmögliche Aktivität eine manuelle Aktivität ist, bleibt das Ticket an der Position hinter der vorangegangenen Aktivität stehen (Nummer **(1)** und **(2)** im folgenden Bild).
- Wenn die nächstmögliche Aktivität eine automatische Aktivität ist, wird die Aktivität ausgeführt, d.h. das Ticket durchläuft diese Aktivität (Nummer **(3)** im folgenden Bild).
- Eine Aktivität kann **eine oder mehrere manuelle** Aktivitäten als Vorgänger-Aktivitäten besitzen **oder** eine Aktivität kann (nur) **eine automatische** Aktivität als Vorgänger-Aktivität besitzen.
- Wenn Sie einen Workflow speichern, führt der Process Designer automatisch eine Konsistenzprüfung durch. Wenn es Inkonsistenzen gibt (z.B. zwei automatische Aktivitäten), wird eine Fehlermeldung angezeigt, und der Workflow kann nicht gespeichert werden.

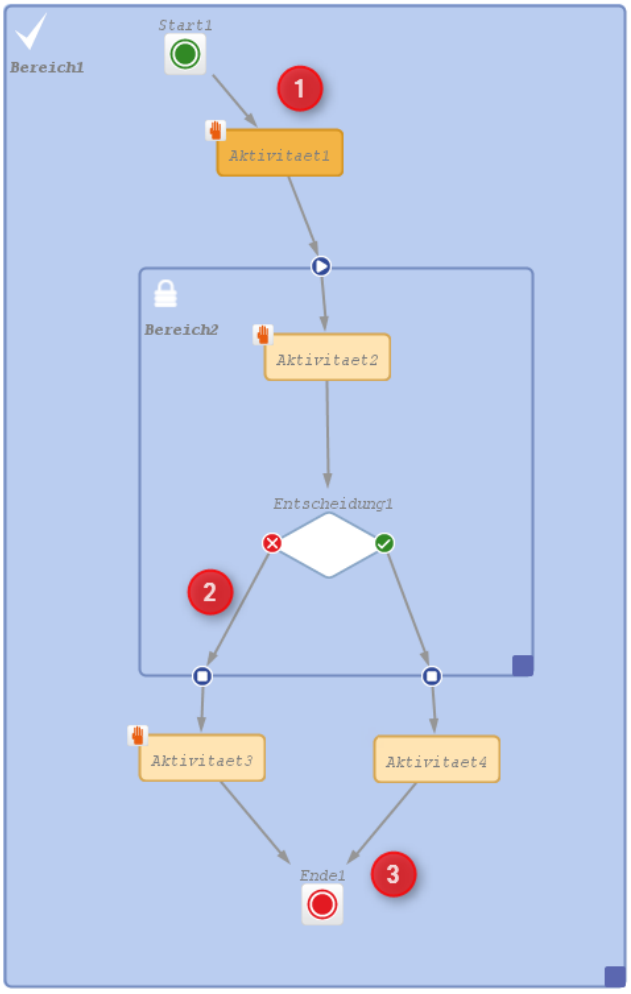


Fig. 1: ConSol*CM Process Designer - Prozesslogik 1

5.2 Interrupts und Exceptions

5.2.1 Einleitung zu Interrupts und Exceptions

Während eines Prozessverlaufs, d.h. während der Zeit, in der ein Ticket geöffnet ist und Bearbeiter daran arbeiten, können Ereignisse auftreten, um die sich ein Mitarbeiter kümmern muss. Beispielsweise, wenn eine E-Mail empfangen wird oder wenn ein Zeitfenster für ein SLA abgelaufen ist, ist es wichtig, dass diese Ereignisse registriert werden und entsprechend zu reagieren.

Es gibt zwei Arten, um die Reaktion und das Verhalten der Tickets zu definieren. Es kann Folgendes implementiert werden:

- **Interrupt**
  Dies ist eine Workflow-Architektur, bei der das Ereignis registriert wird, anschließend eine oder mehrere Aktivitäten ausgeführt werden und das Ticket anschließend direkt an seine vorherige Position im Workflow zurückkehrt.
- **Exception**
  Dies ist eine Workflow-Architektur, bei der das Ereignis registriert wird und das Ticket durch die folgenden manuellen oder automatischen Aktivitäten seine vorherige Position verlässt und zu einer neuen Position innerhalb des Workflows oder eines anderen Workflows springt.

5.2.2 Interrupts

Interrupts ...

- werden durch Trigger aktiviert.
- bewirken eine kurze Unterbrechung des Prozesses, um auf das Ereignis des Triggers zu reagieren.
- verwenden automatische Aktivitäten (eine oder mehrere anschließende automatische Aktivitäten).
- setzen das Ticket anschließend zurück an seine vorherige Position im Workflow, d.h. zurück an die Position, an der es sich befand, bevor das Interrupt-Ereignis gefeuert hat.
- werden häufig genutzt, um das Ticket-Icon mit einem Overlay zu markieren, z.B. wenn eine E-Mail empfangen wurde (siehe Bild unten) oder wenn ein Eskalationszeitpunkt erreicht wurde.

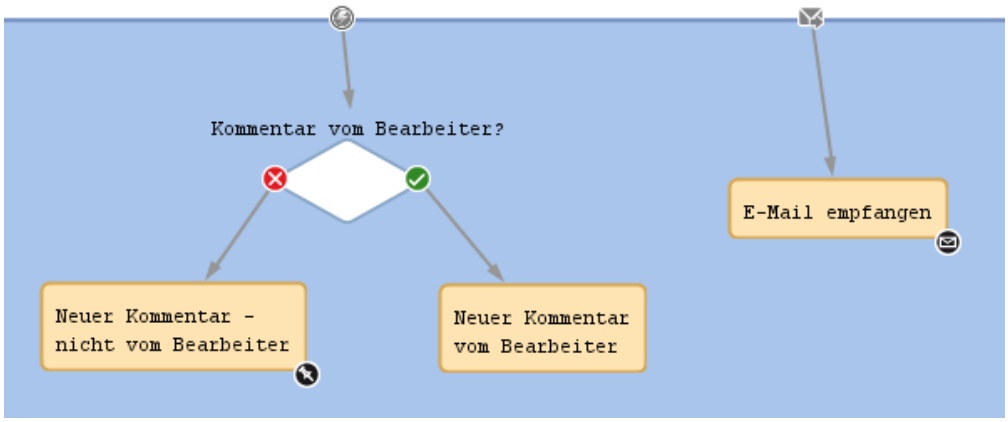


Fig. 2: ConSol*CM Process Designer - Zwei Interrupts

5.2.3 Exceptions

Exceptions ...

- werden durch Trigger aktiviert.
- bewegen das Ticket von seiner alten Position im Workflow zu einer neuen Position. Letztere kann sich im gleichen oder in einem anderen Workflow befinden.
- bewirken, dass der Prozess an der neuen Position fortgesetzt wird.

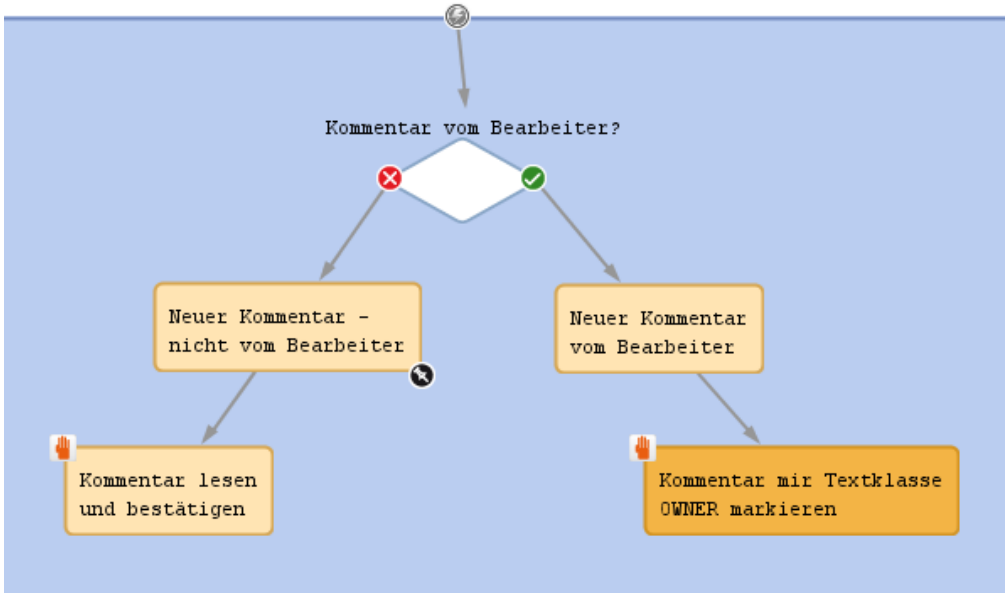


Fig. 3: ConSol*CM Process Designer - Exception

5.3 Schleifen (Fehler in Workflows)

(Endlos-)Schleifen sorgen für Fehler im Prozess. Sie können vom Process Designer nicht entdeckt werden, daher können Sie einen Workflow installieren, der eine Schleife enthält, wie im unteren Bild gezeigt.

Die Prozess-Engine entdeckt jedoch solche Schleifen während der Laufzeit und wirft eine *InfiniteWorkflowLoopException*, um einen vollständigen Systemausfall zu verhindern. Sie können diese Exception selbstverständlich in der *server.log*-Datei sehen. Im Web Client wird eine Fehlermeldung angezeigt.

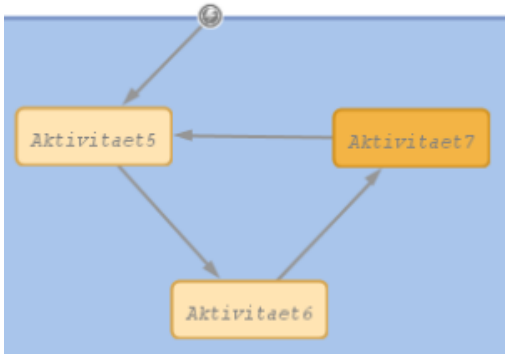


Fig. 4: ConSol*CM Process Designer - Endlosschleife im Workflow

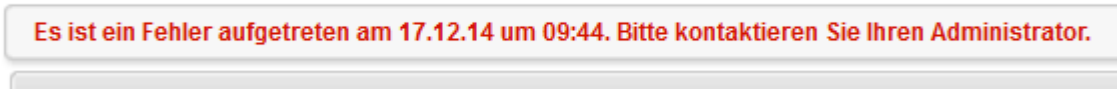


Fig. 5: ConSol*CM Web Client -Fehlermeldung, wenn Schleife erkannt wurde

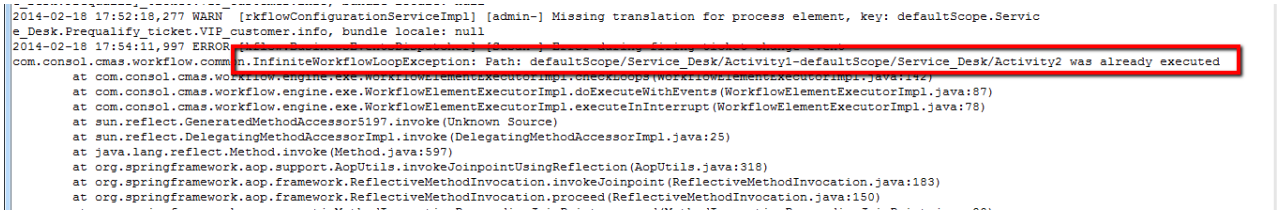


Fig. 6: Console - File server.log: Fehlermeldung aufgrund von Workflow-Schleife

Business-Event-Trigger können ebenfalls Schleifen verursachen, wenn die automatische Aktivität, die mit dem Trigger verbunden ist, die Parameter verändert, auf die der Trigger reagiert. Siehe Abschnitt Best Practices - Vermeiden Sie selbstauslösende Business-Event-Trigger.

5.4 Prozesslogik von Zeit-Triggern

Siehe Abschnitt Business-Logik von Zeit-Triggern.

5.5 Prozesslogik von Business-Event-Triggern

Siehe Abschnitt Business-Logik von Business-Event-Triggern.

6 ConSol CM Process Designer Handbuch - Workflow-Programmierung

6.1 Workflow-Programmierung

- Einleitung
- Zusätzliche Werkzeuge für die Workflow-Programmierung
- Anmerkungen über die Syntax von Methoden
  - Getter-Methoden können häufig ausgelassen werden
  - Setter-Methoden können häufig ausgelassen werden

6.1.1 Einleitung

Die Prozesslogik von ConSol*CM-Workflows wird mittels der zwei Grundpfeiler der ConSol*CM-Prozessintelligenz implementiert:

1. Die Logik, welche sich aus der Reihenfolge von Scopes, Aktivitäten und anderen **Workflow-Elementen** ergibt.
2. Die **Workflow-Skripte** (welche die eigentliche Programmierintelligenz enthalten).

Bisher haben wir uns in diesem Handbuch auf die Erklärung der Workflow-Elemente konzentriert, die über die grafisch gesteuerten Funktionalitäten des Process Designers implementiert werden. Im folgenden Kapitel werden wie einen tieferen Einblick in die Workflow-Entwicklung liefern und die Workflow-Programmierung erklären.

Sie sollten grundlegende Kenntnisse in der *Java und Groovy* Programmierung besitzen, da ConSol*CM-Skripte in Groovy geschrieben werden. Wir werden keine Einleitung zur Programmierung im Allgemeinen geben.

In ConSol*CM-Workflows werden Skripte in den folgenden Kontexten verwendet:

- Als Aktivitätsskript für eine Aktivität.
- Als Vorbedingungsskript für eine Aktivität, das *true* oder *false* zurückgeben muss.
- Als Skript für einen Entscheidungsknoten, das *true* oder *false* zurückgeben muss.
- Als Skript für einen Business-Event-Trigger, das ausgeführt wird, bevor der Trigger feuert
- Als Skript für einen Zeit-Trigger
  - das ausgeführt wird, wenn der Zeit-Trigger initialisiert wird, d.h. wenn das Ticket den Scope betritt, mit dem der Trigger verbunden ist.
  - das ausgeführt wird, wenn der Zeit-Trigger feuert, d.h. wenn eine definierte Zeit abgelaufen ist.
- Als Skript für Endknoten.
- Als Skript für Einsprung- oder Aussprungknoten.
- Als Skripte for ACFs.

Bitte lesen Sie die entsprechenden Abschnitte in diesem Handbuch für eine Erklärung, wie man die Skripte jeweils einfügt.

6.1.2 Zusätzliche Werkzeuge für die Workflow-Programmierung

Um Skripte für Workflow-Elemente zu schreiben, verwenden Sie den Workflow-Skript-Editor, der im Abschnitt Der Skript-Editor erklärt wird.

Als wichtiges Werkzeug werden Sie außerdem die *ConSol*CM Java API Dokumentation* verwenden. Bitte fragen Sie Ihren ConSol*CM Kundenberater oder CM-Consultant nach der entsprechenden *JAR*-Datei. Es handelt sich dabei um eine Standard Java/Groovy API Doc, daher werden Sie sich als erfahrener Java /Groovy-Programmierer schnell zurechtfinden.

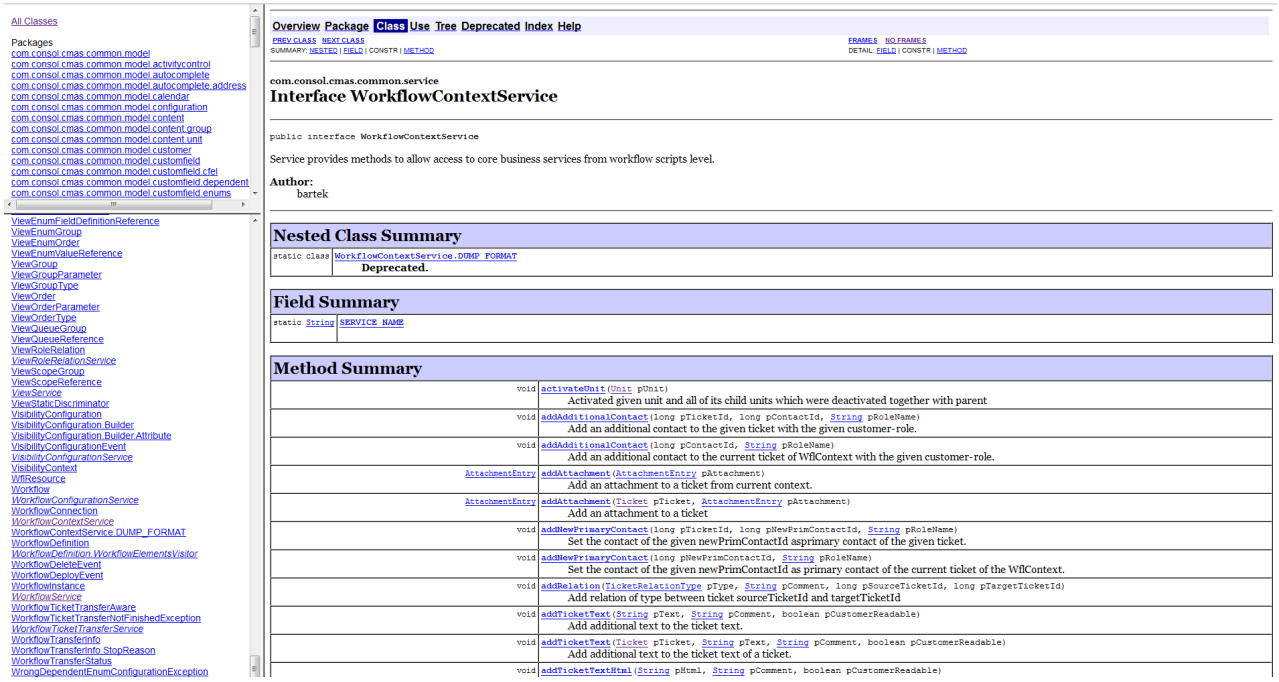


Fig. 1: ConSol*CM Java API Doc

6.1.3 Anmerkungen über die Syntax von Methoden

Wie oben erwähnt, benutzen Sie Groovy-Syntax für ConSol*CM-Skripte. Dabei kann es verschiedene Möglichkeiten geben, den gleichen Inhalt auszudrücken oder zu coden. In den folgenden Abschnitten werden wir Ihnen einige Hinweise geben und eine Beispiele, wie man mit der Groovy-API arbeitet, bereitstellen.

Getter-Methoden können häufig ausgelassen werden

Die meisten Java-Objekte besitzen eine Vielzahl von *Getter*-Methoden, um Werte von Objektattributen abzufragen. In ConSol*CM können Sie entweder die vollständige *Getter*-Methode oder die Kurzform (Convenience) verwenden. Bitte beachten Sie die folgenden Beispiele für Workflow-Skripte.

Anwendungsfall	Java-gemäße Syntax (lange Version)	Groovy-Syntax (Kurzversion)
Hole das Thema eines Tickets.	String mysubject = ticket.getSubject()	def mysubject = ticket.subject
Hole den Bearbeiter eines Tickets.	Engineer myeng = ticket.getEngineer()	def myeng = ticket.engineer
Hole den Hauptkontakt eines Tickets.	Unit mymaincontact = ticket.getMainContact()	def mymaincontact = ticket.mainContact
Hole den Wert eines bestimmten Benutzerdefinierten Felds von einem Ticket.	String myprio = ticket.get("helpdesk_fields", "prio")	def myprio = ticket.get("helpdesk_fields.prio")
Hole den Unit-Typ für den Hauptkontakt des Tickets.	Unit mycustomer = workflowApi.getPrimaryContact() UnitDefinition myunitdef = mycustomer.getDefinition() UnitDefinitionType mydeftype = myunitdef.getType()	def mycustomer = workflowApi.primaryContact def myunitdef = mycustomer.definition def mydeftype = mycustomer.definition.type

Der Zugriff auf Benutzerdefinierte Felder kann nicht abgekürzt werden, da es für diese Felder keine Getter-Methoden gibt. Bitte lesen Sie den Abschnitt Arbeiten mit Datenfeldern für Details zur Arbeit mit Daten aus Benutzerdefinierten Feldern.

Setter-Methoden können häufig ausgelassen werden

Die meisten Java-Objekte besitzen eine Vielzahl von *Setter*-Methoden, um Werte von Objektattributen zu setzen.In ConSol*CM können Sie entweder die vollständige *Setter*-Methode oder die Kurzform (Convenience) verwenden. Bitte beachten Sie die folgenden Beispiele für Workflow-Skripte.

Anwendungsfall	Java-gemäß Syntax (lange Version)	Groovy-Syntax (Kurzversion)
Setze das Thema eines Tickets.	ticket.setSubject("asd")	ticket.subject = "asd"

6.2 Wichtige Klassen und Objekte

- Einleitung
- Wichtige Objekte
  - Ticket
  - workflowAPI
- Convenience-Klassen und -Methoden
  - Beispiel 1: Verwendung des ConfigurationService, um System-Properties abzurufen
  - Beispiel 2: Verwendung des EngineerService, um das Ticket einem Genehmiger zuzuweisen
  - Beispiel 3: Verwendung des EnumService, um einen Enum-Wert nach dessen Namen abzurufen
  - Beispiel 4: Verwendung des TicketService, um alle Tickets einer bestimmten Sicht abzurufen
  - Beispiel 5: Verwendung des EngineerRoleRelationService, um allen Bearbeitern mit einer bestimmten Rolle eine E-Mail zu senden

6.2.1 Einleitung

Um die ConSol*CM-Skriptprogrammierung zu vereinfachen, bietet die CM Workflow API einfachen Zugriff auf häufig genutzte Objekte. Des weiteren ermöglichen Convenience-Klassen und -Methoden einen kurzen Weg zu verschiedenen Objekten und Methoden.

6.2.2 Wichtige Objekte

Manche Objekte sind implizit in Workflow-Skripten vorhanden.

> ⚠ **Vorsicht:**
>
> Die gleichen Objekte sind **nicht** in Admin-Tool-Skripten vorhanden, d.h. in Admin-Tool-Skripten müssen Sie *import*-Befehle verwenden!

Ticket

In jedem Workflow-Skript kann auf das aktuelle Ticket einfach durch das Objekt *ticket* zugegriffen werden. Es ist aus der Klasse *Ticket* abgeleitet und implizit vorhanden. Es wird kein Import und keine Instantiierung benötigt.

Beispiel:

Verwendung des Ticket-Objekts

```
def myId = ticket.getId()
```

workflowAPI

Das Objekt *workflowApi* ist ebenfalls implizit vorhanden. Es ermöglicht einfachen Zugriff auf das Interface *WorkflowContextService,* welches für verschiedene Operationen verwendet wird.

Beispiele:

Verwendung von workflowApi für das Senden einer E-Mail

```
workflowApi.sendEmail(contact_e,subj,text,replyto,null)
```

Verwendung von workflowApi, um das Ticket dem aktuellen Bearbeiter zuzuweisen

```
def curr_eng = workflowApi.getCurrentEngineer()
ticket.setEngineer(curr_eng)
```

Verwendung von workflowApi, um einen Trigger zu deaktivieren

```
workflowApi.deactivateTimer("defaultScope/Service_Desk/TimeTrigger1")
```

Verwendung von workflowAppi, um dem Bearbeiter eine Nachricht im Web Client anzuzeigen

```
 workflowApi.addValidationError("1", "Das Ticket kann nicht geschlossen werden, bevor eine
Lösung angegeben wird. Bitte geben Sie eine Lösung ein und markieren Sie diese mit der
Textklasse LÖSUNG.") }
```

6.2.3 Convenience-Klassen und -Methoden

Die ConSol*CM API stellt verschiedene Convenience-Interfaces und -Methoden bereit, die den Zugriff auf die meisten Objekte des alltäglichen CM-Programmierens stark vereinfachen. Die meisten dieser Convenience-Interfaces sind Teil des Packages *com.consol.cmas.common.service* und dessen Sub-Packages. Details entnehmen Sie bitte der *ConSol*CM Java API Documentation*. An dieser Stelle werden wir Ihnen einige Beispiele zeigen, die sich für die meisten CM-Programmierer als nützlich erweisen können.

Die implementierende Instanz eines Interfaces ist immer verfügbar, indem der erste Buchstabe, der ein Großbuchstabe ist, im Klassennamen durch einen Kleinbuchstaben ersetzt wird, z.B. das Objekt (Singleton) mit dem Interface *EngineerService* ist verfügbar mit dem Objekt *engineerService*, siehe *Beispiel 2*.

Beispiel 1: Verwendung des ConfigurationService, um System-Properties abzurufen

Verwendung von ConfigurationService, um die Nummer des Bearbeitermanagement-Tickets abzurufen

```
def tic_nr = configurationService.getValue("custom-mycompany-properties","engineer_management.
ticket.nr")
// dann: ... tue etwas mit dem Bearbeitermanagement-Ticket, z.B. finde den Namen des naechsten
Bearbeiters heraus, dem ein Service-Ticket zugewiesen werden soll.
```

Verwendung von ConfigurationService, um die Basis-URL des Systems abzurufen

```
def baseUrl = configurationService.getValue("custom-mycompany-properties","base.url.mycompany")
def url = baseUrl + "/cm-client/ticket/ticket_name/" + ticket.getName()
def itComplete = url + " " + ticket.getName()
// ... tue etwas mit der Ticket-URL, z.B. platziere den Link zu einem Child-Ticket in einer
Tabelle des Parent-Tickets
```

Beispiel 2: Verwendung des EngineerService, um das Ticket einem Genehmiger zuzuweisen

Beispiel mit Verwendung des EngineerService

```
// Skript tut Folgendes:
// Übergibt das Tickt erst an einen Genehmiger, wenn der Genehmiger im Ticket als zusaetzlicher
Bearbeiter gesetzt wurde.
// Import-Package, da Klassen andernfalls nicht im Workflow verfuegbar sind:
import com.consol.cmas.common.model.ticket.user.function.*
// Hole den Namen des Genehmigers, der in einem Benutzerdefinierten Feld steht/gespeichert ist,
// nämlich im Feld mit dem Namen „CF_ApproverName“ in der Benutzerdefinierten Feldgruppe
// „CF_GroupApproverData“. Der Wert koennte z.B. „Mr. Miller“ sein:
def gen = ticket.get("CF_GroupApproverData.CF_ApproverName").getName()
// Hole das Bearbeiterobjekt, in dem der Name „Mr. Miller“ gesetzt ist, d.h. das
Bearbeiterobjekt
// des gewuenschten Genehmigers:
def gen_eng = engineerService.getByName(gen)
// Hole das ticketFunction-Objekt, das die ticketFunction (Bearbeiterfunktion) „Approver“
repraesentiert:
TicketFunction tf = ticketFunctionService.getByName("Approver")
// Füge das Bearbeiterobjekt von Mr. Miller als Genehmiger hinzu. d.h. in der ticketFunction
// (Bearbeiterfunktion) „Approver“ zum Ticket hinzu. Einer der Parameter ist ticket. Dies
// muss nicht instantiiert werden, da es implizit in Workflow-Skripten vorhanden ist:
def tu = ticketUserService.addTicketUser(ticket, gen_eng, tf, "Approver")
// Weise das Ticket dem Bearbeiter zu, d.h. setze den Bearbeiter Mr. Miller auch als Ticket-
Owner.
def tic2 = workflowApi.assignEngineer(ticket, gen_eng)
```

Wir haben hier zwei Zuweisungen:

1. Mr. Miller wird als zusätzlicher Bearbeiter mit der Bearbeiterfunktion *Approver* gesetzt.
2. Mr. Miller wird als (regulärer) Ticket-Bearbeiter gesetzt

Beispiel 3: Verwendung des EnumService, um einen Enum-Wert nach dessen Namen abzurufen

Verwendung von EnumService, um einen Enum-Wert nach dessen Namen abzurufen

```
def enumValueMLA = enumService.getValueByName( "priority", "REGULAR" )
ticket.set( "helpdesk_fields.prio", enumValueMLA )
```

Beispiel 4: Verwendung des TicketService, um alle Tickets einer bestimmten Sicht abzurufen

Verwendung von TicketService, um alle Tickets einer bestimmten Sicht abzurufen

```
List<Ticket> mylist = ticketService.getByView(new ViewCriteria(
    viewService.getByName("helpdesk_active_tickets"),
    ViewAssignmentParameter.allAssignedTickets(),
    ViewGroupParameter.allTickets(),
    viewOrderParameter.addByName(true)))
```

Beispiel 5: Verwendung des EngineerRoleRelationService, um allen Bearbeitern mit einer bestimmten Rolle eine E-Mail zu senden

Verwendung von EngineerRoleRelationService, um allen Bearbeitern mit einer bestimmten Rolle eine E-Mail zu senden

```
// Sende eine E-Mail an alle Bearbeiter mit einer bestimmten Rolle
def mail = new Mail()
mail.setTo(engineerRoleRelationService.getEngineersWithRoles(roleService.getByName("Supervisor")
)*.email.join(","))
mail.setSubject("Ticket (${ticket.name}) -- Eskalation!")
mail.setText(workflowApi.renderTemplate("Ticket escalation note to supervisor"))
mail.send()
```

6.3 Arbeiten mit Datenfeldern

- Einleitung zu Datenfeldern
  - ConSol*CM-Version 6.8 und niedriger
  - ConSol*CM-Version 6.9 und höher
- Datentypen für Datenfelder
- Benutzerdefinierte Felder für Ticketdaten
  - Die wichtigsten Methoden für den Zugriff auf Benutzerdefinierte Felder für Ticketdaten
  - Abrufen von Werten aus Benutzerdefinierten Feldern für Ticketdaten
    - Einfache Datentypen
    - Enum-Werte
    - Listen
      - Listen mit einfachen Datentypen
      - Lists of Structs (Tabellen)
  - Setzen von Werten in Benutzerdefinierten Feldern für Ticketdaten
    - Setzen von Werten in Benutzerdefinierten Feldern mit einfachen Datentypen
    - Setzen von Enum-Werten
    - Setzen von Listen-Werten
      - Setzen von Werten in Listen mit einfachen Datentypen
      - Setzen von Werten in Lists of Structs
  - Ein- und Ausblenden von Benutzerdefinierten Feldgruppen
- Datenfelder für Kundendaten
  - Benutzerdefinierte Felder für Kundendaten (CM-Version 6.8 und niedriger)
    - Abrufen von Werten
    - Setzen von Werten für Kundendaten in CM-Version 6.8 und niedriger
  - Datenobjektgruppenfelder für Kundendaten (CM-Version 6.9 und höher)
    - Die wichtigsten Methoden für den Zugriff auf Datenobjektgruppenfelder für Kundendaten
    - Abrufen von Werten für Kundendaten in CM-Version 6.9 und höher
    - Setzen von Werten für Kundendaten in CM-Version 6.9 und höher
      - Setzen von Werten für Datenobjektgruppenfelder mit einfachen Datentypen
      - Listen
        - Setzen von Werten in einer List of Structs für Kundendaten
  - Convenience-Methoden für den Zugriff auf Kundendaten in CM-Version 6.9 und höher
- Datenfelder für (unsichtbare) Variablen nutzen

6.3.1 Einleitung zu Datenfeldern

Der Zugriff auf Datenfelder ist ein essentieller Bestandteil der ConSol*CM-Programmierung. Er kann in allen Skripten des Systems benötigt werden, sowohl in Workflow- als auch in Admin-Tool-Skripten, unabhängig davon, welchem Typ diese angehören. Hier legen wir den Fokus auf die Workflow-Programmierung, der Zugriff auf Datenfelder funktioniert jedoch in allen Skripten grundsätzlich gleich.

ConSol*CM-Version 6.8 und niedriger

In ConSol*CM-Versionen 6.8 und niedriger werden alle Datenfelder **Benutzerdefinierte Felder** (Custom Fields = **CFs**) genannt. CFs werden verwendet, um das CM-Datenmodell zu definieren, welches aus **Ticketdaten** und **Kundendaten** besteht. Das Layout des Web Clients wird ebenfalls mittels CFs definiert, die bestimmte Annotationen (z.B. *position*) besitzen.

Beispiel für Benutzerdefinierte Felder für Ticketdaten sind:

- Priorität des Tickets
- Eskalationsdatum entsprechend eines SLA
- Drucker-Modell
- Vertragsnummer

Beispiel für Benutzerdefinierte Felder für Kundendaten sind:

- Name des Kunden
- Postleitzahl
- Telefonnummer
- E-Mail-Adresse

Für eine detaillierte Beschreibung der Arbeit mit Benutzerdefinierten Feldern für Ticketdaten lesen Sie bitte das *ConSol*CM Administratorhandbuch 6.8*, Abschnitt *Verwaltung der Benutzerdefinierten Felder*.

Regeln für die Arbeit mit Benutzerdefinierten Feldern für CM Version 6.8 oder niedriger:

Es gibt zwei Hauptregeln, die Sie beachten müssen, wenn Sie mit Benutzerdefinierten Feldern arbeiten:

1. Benutzerdefinierte Felder werden immer durch Benutzerdefinierten Feldgruppen verwaltet und referenziert, wenn Sie z.B. den Wert eines CF abrufen möchten, verwenden Sie <CF GroupName>.<CF FieldName>
2. Sie verwenden immer den technischen, eindeutigen Namen, um auf ein Benutzerdefiniertes Feld oder eine Benutzerdefinierte Feldgruppe zu referenzieren, nicht den lokalisierten Wert.

ConSol*CM-Version 6.9 und höher

Beginnend mit ConSol*CM-Version 6.9.0 gibt es zwei Typen von Datenfeldern:

- **Benutzerdefinierte Felder**
  Werden verwendet, um Ticketdaten zu definieren und in Benutzerdefinierten Feldgruppen verwaltet, so wie aus vorherigen CM-Versionen bekannt.

- **Datenobjektgruppenfelder**
  Werden verwendet, um Kundendaten zu definieren, als Teil *des FlexCDM*, dem neuen CM-Kundendatenmodell. Verwaltet in Datenobjektgruppen.

Die Arbeit mit Benutzerdefinierten Feldern des neuen (Version 6.9 und höher) Kundendatenmodells (*FlexCDM*) wird detailliert erklärt im *ConSol*CM Administratorhandbuch, Abschnitt Das CM-Kundendatenmodell:FlexCDM*.

Regeln für die Arbeit mit Benutzerdefinierten Feldern CM Version 6.9 und höher:

Es gibt drei Hauptregeln, die Sie beachten müssen, wenn Sie mit Benutzerdefinierten Feldern und Datenobjektgruppenfelder arbeiten:

1. Benutzerdefinierte Felder werden immer durch Benutzerdefinierten Feldgruppen verwaltet und referenziert, wenn Sie z.B. den Wert eines CF abrufen möchten, verwenden Sie <CF GroupName>.<CF FieldName>
2. Datenobjektgruppenfelder werden immer durch *Datenobjektgruppen* verwaltet und referenziert, wenn Sie z.B. den Wert eines Datenobjektgruppenfelds abrufen möchten, verwenden Sie <Data Object GroupName>:<Data Object Group FieldName>
3. Sie verwenden immer den technischen, eindeutigen Namen, um auf ein Benutzerdefiniertes Feld, eine Benutzerdefinierten Feldgruppe, eine Datenobjektgruppe oder auf ein Datenobjektgruppenfeld zu referenzieren, nicht den lokalisierten Wert.

6.3.2 Datentypen für Datenfelder

Ein Datenfeld gehört immer einem bestimmten Datentyp an. Wie für jede Variable in der Programmierung, hängt es vom Datentyp ab, wie mit dem Wert dieses Felds umzugehen ist, z.B: kann ein *string*-Feld nicht zum Rechnen mit Zahlen verwendet werden und ein *enum*-Feld benötigt eine spezielle Zugriffsmethode.

Die folgenden Datentypen sind in ConSol*CM verfügbar:

- **boolean (Ja/Nein)**
  Werte: *true* oder *false*. Abhängig von der Annotation *boolean-type*, wird der Wert entweder als Checkbox, Radiobuttons oder Drop-Down-Liste dargestellt.
- **date (Datum)**
  Format und Genauigkeit können über Annotationen gesetzt werden.
- **enum (Sortierte Liste)**
  Für *Sortierte Listen*. Der Bearbeiter kann einen der enum-Werte im Web Client auswählen. Enums und Werte müssen zuvor in der *Verwaltung von Sortierten Listen* im Admin-Tool erstellt werden *(* siehe *ConSol*CM Administratorhandbuch)*.
- **list (Liste)**
  Ein Datenfeld dieses Typs ist die Basis für eine *Liste* (eine Spalte) oder eine *Tabelle* (mehrere

Spalten) aus mehreren Eingabefeldern im Web Client. Eine Tabelle enthält Zeilen, jede davon mit dem Datentyp *struct* (siehe unten). Jede Zeile (*struct*) enthält individuelle Datenfelder. Eine einfache Liste besteht aus einem list-Feld, das die Benutzerdefinierten Felder enthält.

- **struct (Struktur)**
  Ein Datenfeld dieses Typs definiert eine Datenstruktur (Zeile oder Tabelle), welche ein oder mehrere Feld(er) gruppiert.
- **number (Zahl)**
  Für Ganzzahl-Werte.
- **fixed point number (Festkommazahl)**
  Für Zahlen mit einer festen Kommastelle, z.B. Währungen. Sie müssen die Gesamtanzahl der Stellen (Genauigkeit) und die Anzahl der Stellen, die sich hinter dem Komma befinden sollen (Skalierung) festlegen.
- **string (Text)**
  Für bis zu 4000 alphanumerische Zeichen.
- **long string (Text)**
  Für lange Texte, keine Einschränkung in der Länge.
- **short string (Text)**
  Für bis zu 255 alphanumerische Zeichen.
- **contact data reference (Referenz auf ein Kontaktdatenfeld)** (bis zu Version 6.8)
  Dieser spezielle Datentyp wird intern genutzt, um von einem Kontakt auf die Firma, zu der der Kontakt gehört, zu referenzieren. Für diesen Datentyp muss zusätzlich noch der *Kontaktdatentyp* (Kunde oder Firma) im entsprechenden Feld ausgewählt werden.
- **MLA field (Baum Sortierte Liste)**
  Dieser Datentyp wird für Benutzerdefinierte Felder benutzt, die hierarchische Listen mit einer Baumstruktur beinhalten, genannt *MLA* (Multi Level Attributes). Der Name des Benutzerdefinierten Felds wird als Name für das neue MLA gesetzt, das in der MLA-Verwaltung definiert werden muss. Die Gruppe des Benutzerdefinierten Felds muss referenziert werden, wenn das MLA erstellt wird.

6.3.3 Benutzerdefinierte Felder für Ticketdaten

Im Admin-Tool werden die Benutzerdefinierte Felder für Ticketdaten in der *Verwaltung von Benutzerdefinierte Feldern* definiert, Tab *Ticketdaten*.

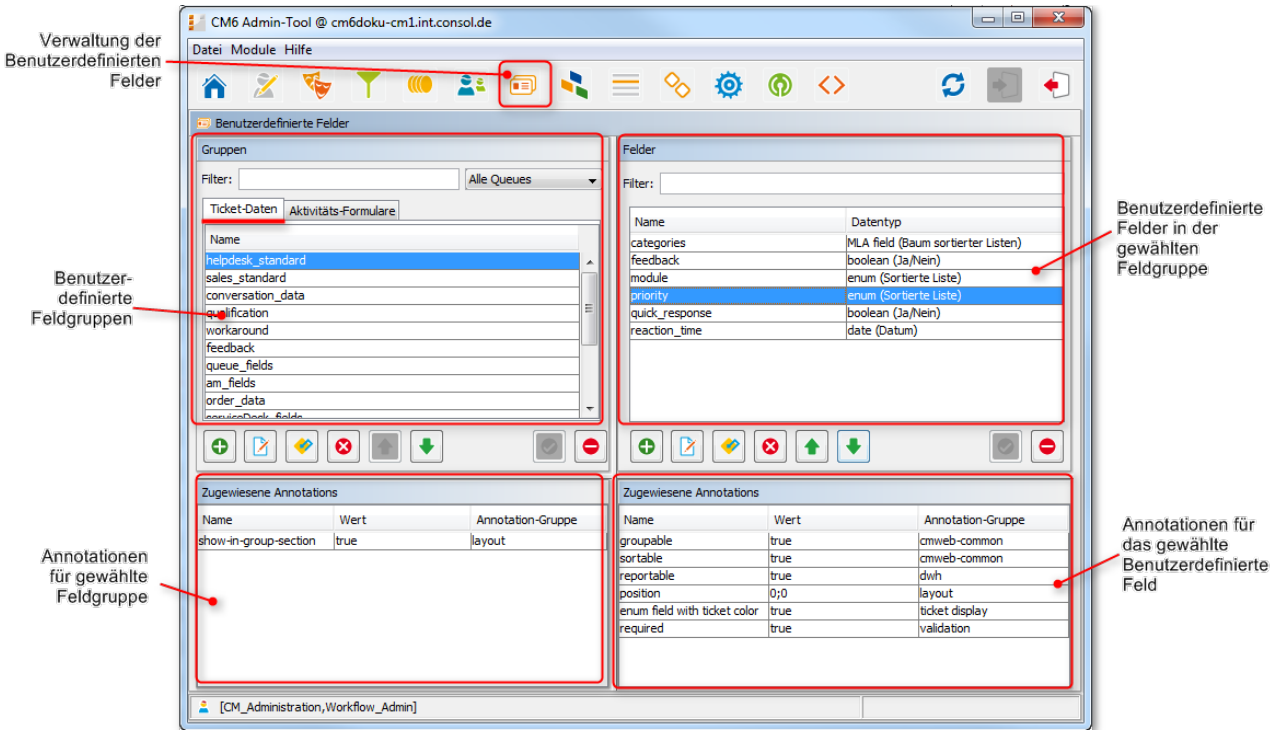


Fig. 1: ConSol*CM Admin-Tool: Verwaltung der Benutzerdefinierten Felder für Ticket-Daten (CM ab Version 6.9)

Die wichtigsten Methoden für den Zugriff auf Benutzerdefinierte Felder für Ticketdaten

Drei Methoden sind bei der Programmierung von größter Wichtigkeit für den Zugriff auf CFs in CM-Skripten. Sie sind alle Methoden der Klasse *Ticket*:

- **Ticket.get()**
  - Für das Abrufen von Daten aus einem CF.
- **Ticket.set()**
  - Für das Setzen von Daten in einem bereits existierenden CF.
- **Ticket.add()**
  - Für das Rechnen mit einem Wert innerhalb eines CF, d.h. um eine bestimmte Zeitspanne zu einem *Datum*-Feld hinzuzufügen.
  - Um eine neue Zeile zu *list*-Feldern hinzuzufügen (einfache Listen und Tabellen).

Eine weitere Methode kann benutzt werden, wenn ein Feld geleert werden soll, d.h. wenn der Wert auf *null* gesetzt werden soll:

- **Ticket.remove()**
  - Setzt den Wert des Feldes auf *null*.

Abrufen von Werten aus Benutzerdefinierten Feldern für Ticketdaten

Um Daten aus einem Benutzerdefinierten Feld in einem Skript abzurufen, müssen Sie dieses mittels des technischen Namens der Benutzerdefinierten Feldgruppe und des Benutzerdefinierten Felds referenzieren. Die Methode, die verwendet werden muss, kann abhängig vom Datentyp des CF unterschiedlich sein.

Einfache Datentypen

Die folgenden Beispiele beziehen sich auf die Benutzerdefinierten Felder im Bild oben. Folgende Methode sollte verwendet werden (da diese die einfachste Möglichkeit darstellt):

```
ticket.get("<Group_name>.<CF_name>")
```

Vorsicht:

Bitte beachten Sie, dass die *Getter*-Methoden für Attribute das *Attribut* (ein *Objekt*) zurückgeben und nicht den *Wert* des Objekts!

Zum Beispiel:

ticket.getField("helpdesk_standard","reaction_time") gibt ein *AbstractField* zurück.

Wenn Sie mit dem *Wert* des Felds arbeiten möchten, verwenden Sie:

def myvalue = ticket.get("helpdesk_standard", "reaction_time")

Oder:

def myfield = ticket.getField("helpdesk_standard", "reaction_time");

def myvalue = myfield.getValue();

Am besten:
(diese Version empfehlen wir für den Standardgebrauch)

def myvalue = ticket.get("helpdesk_standard.reaction_time")

Den Wert eines boolean-CF abrufen

```
def fedb = ticket.get("helpdesk_standard.feedback")
// wird TRUE oder FALSE oder NULL zurueckgeben da es ein BOOLEAN-Feld ist
```

Ein *Bedingungs*-Skript für eine Workflow-Aktivität könnte wie der folgende Code aussehen:

Bedingungs-Skript, bei dem ein boolean-Wert überprüft wird

```
boolean vip_info = ticket.get("am_fields.vip");
if(vip_info == true){
  return true;
}
else {
  return false;
}
```

Oder kürzer::

Bedingungs-Skript, bei dem ein boolean-Wert überprüft wird. Kurzversion

```
return ticket.get("am_fields.vip")
```

Enum-Werte

Ein enum-Feld (Sortierte Liste) ist ein Feld, bei dem der Wert einer von verschiedenen möglichen Listenwerten ist. Zum Beispiel ist die Liste mit Prioritäten die Basis für ein enum-Feld. Um den Wert eines enum-Felds abzurufen, können Sie die gleiche Syntax wie für einfache Datentypen verwenden. Die *get*-Methode liefert den enum-Listenwert, die *getName()*-Methode liefert das *string*-Attribut mit dem Namen des Werts.

Abrufen eines enum-Werts für ein CF

```
def prio = ticket.get("helpdesk_standard.priority")
println "Priority is now " + prio.getName()
```

Listen

Listen mit einfachen Datentypen

Eine Liste mit einfachen Datentypen besteht aus einer Liste (= array), welche in jeder Zeile einen Wert eines einfachen Datentyps besitzt, ein *Datum* in unserem Beispiel. Das CF vom Typ *date* (Datum) muss den Parameter *Gehört zu* besitzen, welcher auf die Liste verweist.

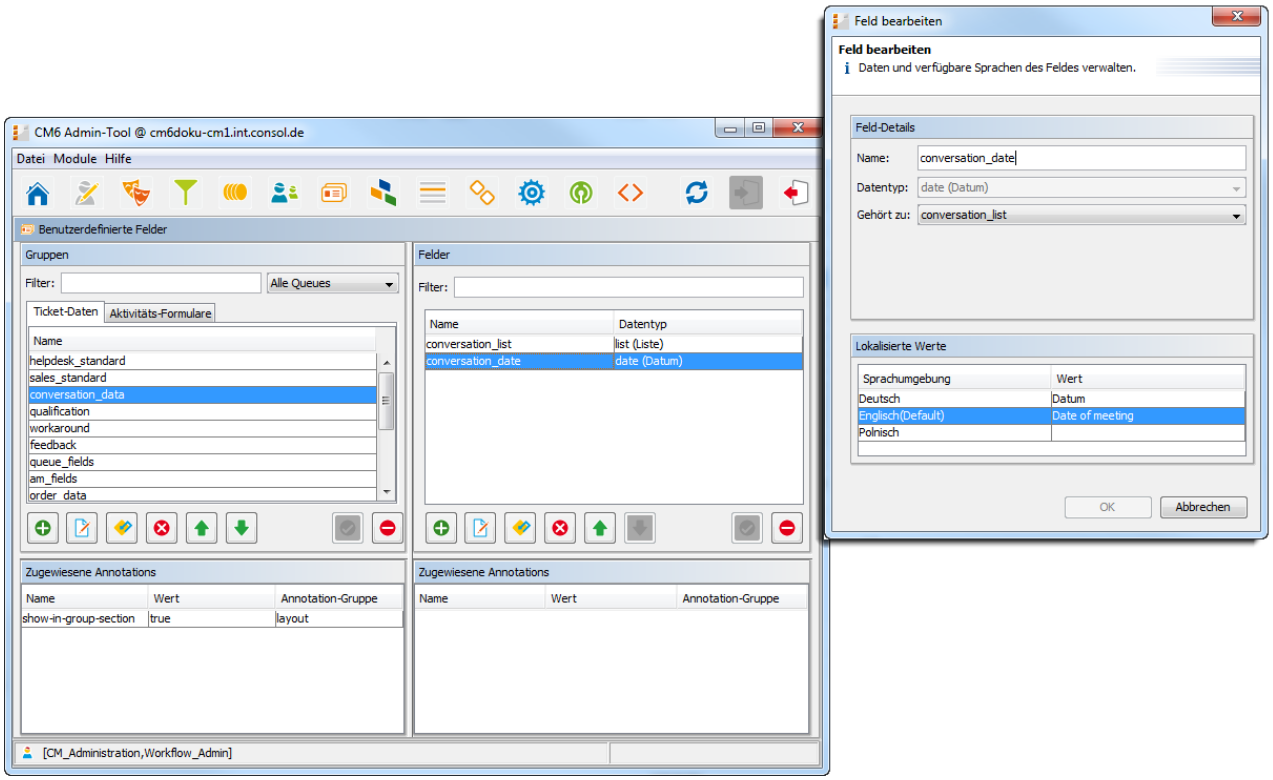


Fig. 2: ConSol*CM Admin-Tool - CFs für eine Liste mit Datumsfeldern

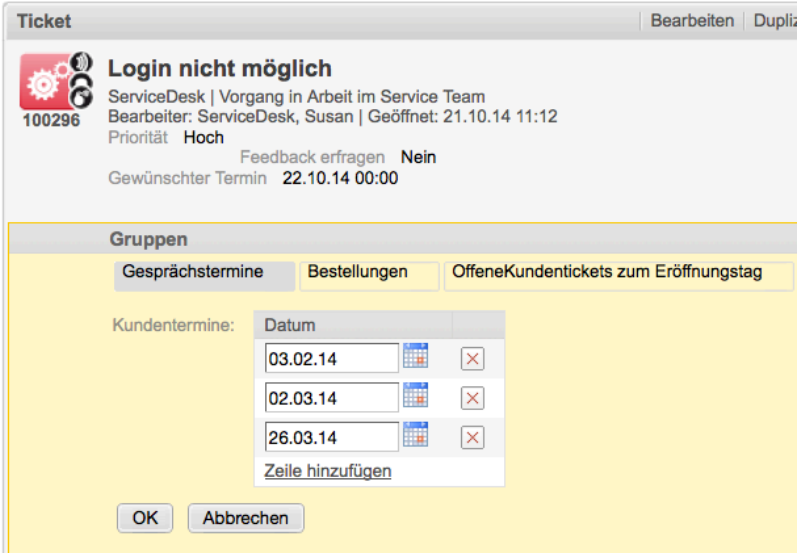


Fig. 3: ConSol*CM/Web Client - Liste mit Datumsfeldern in einem Ticket (Bearbeiten-Modus)

Für den Zugriff auf jedes *Datum*-CF innerhalb der Liste verwenden Sie den folgenden Code:

Anzeige des Inhalts einer Liste von Datumsobjekten

```
def convs = ticket.get("conversation_data.conversation_list").each() { conv ->
    println "NEXT DATE is :" + conv
    println "CLASS of NEXT DATE is " + conv.getClass()
}
```

```
2014-03-27 12:21:45,274 INFO  [STDOUT                    ] NEXT DATE is :2014-02-03 00:00:00.0
2014-03-27 12:21:45,279 INFO  [STDOUT                    ] CLASS of MEXT DATE is class java.sql.Timestamp
2014-03-27 12:21:45,280 INFO  [STDOUT                    ] NEXT DATE is :2014-03-02 00:00:00.0
2014-03-27 12:21:45,280 INFO  [STDOUT                    ] CLASS of MEXT DATE is class java.sql.Timestamp
2014-03-27 12:21:45,281 INFO  [STDOUT                    ] NEXT DATE is :2014-03-26 00:00:00.0
2014-03-27 12:21:45,281 INFO  [STDOUT                    ] CLASS of MEXT DATE is class java.sql.Timestamp
```

Fig. 4: Log File - Skript-Output

Um auf eine bestimmte Zeile zuzugreifen, können Sie die folgende Syntax verwenden:

Rufe einen bestimmten Wert aus einer Liste von einfachen Datentypen ab

```
def mydate = ticket.get("conversation_data.conversation_list[1]")
```

Lists of Structs (Tabellen)

Das Datenkonstrukt *list of structs* ist die technische Basis für eine Tabellen-Struktur im Web Client. Die *Liste* (*list*) ist das Parent-Objekt, welches Zeilen enthält. Jede Zeile ist eine Instanz eines *structs*. Jede Zeile (*struct*) enthält so viele Benutzerdefinierte Felder (Tabellenspalten), wie benötigt.

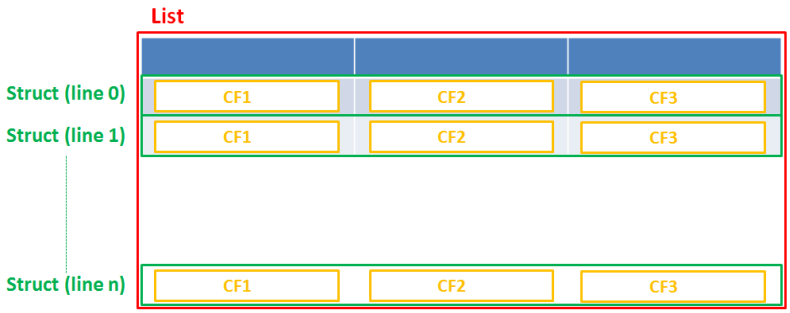


Fig. 5: List of Structs - Logisches Prinzip

Technisch gesehen ist die Liste ein *array*, das in jedem Feld eine *map* (=key:value pairs) enthält.

Fig. 6: List of Structs - Technisches Prinzip

Um Daten aus einer List of Structs abzurufen, können Sie mit einer Iteration über die Zeilen (= *structs*) arbeiten. Im folgenden Beispiel (aus einem Bestellsystem, nicht im Bild oben dargestellt) arbeiten wir mit einer Tabelle, in der ...

- das CF *orders_list* die Liste (*list*) repräsentiert.
- sich CF *orders_list* in der CF-Gruppe *order_data* befindet.
- der Iterator *str* jeweils die Zeile (das *struct*) repräsentiert .
- die Zeile (struct) besitzt drei Felder:

- *orders_hardware*
  welches den Artikel, der bestellt werden soll, enthält (*enum*).
- *orders_contact*
  welches die Kontaktperson enthält (*string*).
- *orders_number*
  welches die Anzahl der Artikel, die bestellt werden sollen, enthält (*integer*).

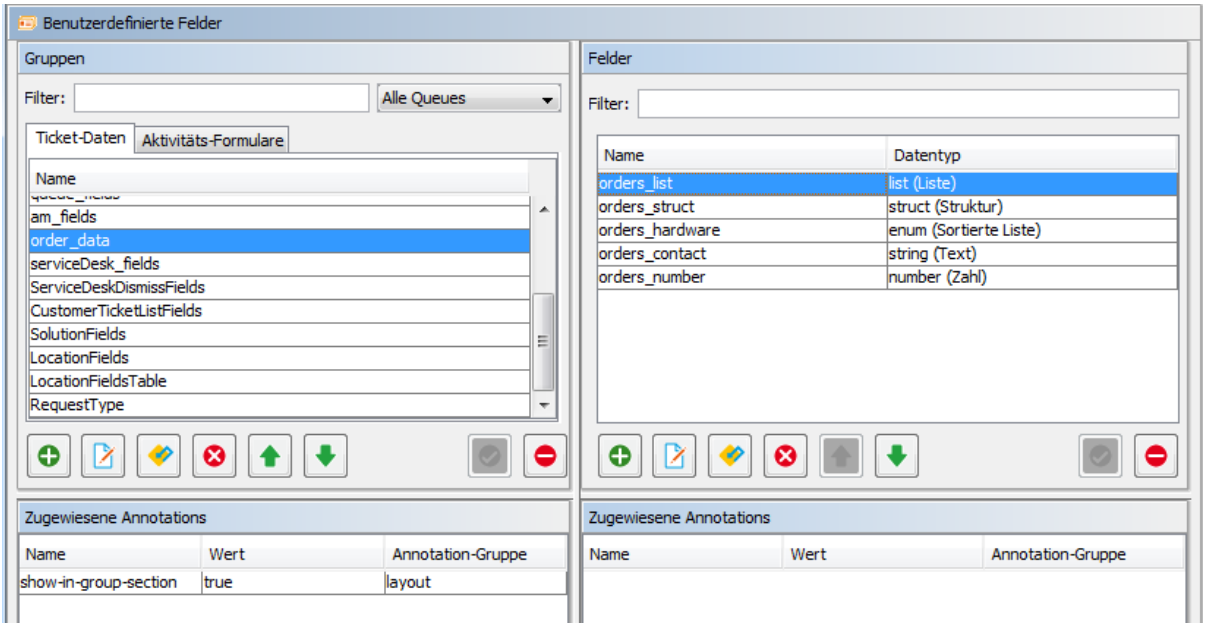


Fig. 7: ConSol*CM Admin-Tool - Benutzerdefinierte Felder für Lists of Structs

Fig. 8: ConSol*CM Web Client - Ticket mit ausgefüllter Tabelle

Abrufen von Daten aus einer List of Structs

```
def structs = ticket.get("order_data.orders_list").each() { str ->
    println("CLASS of LINE is " + str.getClass())
    println("FIELD VALUE HARDWARE is " + str.orders_hardware.getName())
    println("CLASS of FIELD VALUE HARDWARE is " + str.orders_hardware.getName().getClass())
    println("FIELD VALUE CONTACTis " + str.orders_contact)
    println("CLASS of FIELD VALUE CONTACT is " + str.orders_contact.getClass())
    println("FIELD VALUE NUMBER is " + str.orders_number)
    println("CLASS of FIELD VALUE NUMBER is " + str.orders_number.getClass())
}
```

```
2014-03-27 11:39:44,425 INFO  [STDOUT                    ] CLASS of LINE is class com.consol.cmas.common.model.customfield.cfel.Struct
2014-03-27 11:39:44,429 INFO  [STDOUT                    ] FIELD VALUE HARDWARE is large_printers
2014-03-27 11:39:44,429 INFO  [STDOUT                    ] CLASS of FIELD VALUE HARDWARE is class java.lang.String
2014-03-27 11:39:44,430 INFO  [STDOUT                    ] FIELD VALUE CONTACTis Mr. Miller
2014-03-27 11:39:44,431 INFO  [STDOUT                    ] CLASS of FIELD VALUE CONTACT is class java.lang.String
2014-03-27 11:39:44,431 INFO  [STDOUT                    ] FIELD VALUE NUMBER is 2
2014-03-27 11:39:44,454 INFO  [STDOUT                    ] CLASS of FIELD VALUE NUMBER is class java.lang.Long
2014-03-27 11:39:44,455 INFO  [STDOUT                    ] CLASS of LINE is class com.consol.cmas.common.model.customfield.cfel.Struct
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] FIELD VALUE HARDWARE is medium_printers
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] CLASS of FIELD VALUE HARDWARE is class java.lang.String
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] FIELD VALUE CONTACTis Mrs. Summer
2014-03-27 11:39:44,457 INFO  [STDOUT                    ] CLASS of FIELD VALUE CONTACT is class java.lang.String
2014-03-27 11:39:44,458 INFO  [STDOUT                    ] FIELD VALUE NUMBER is 5
2014-03-27 11:39:44,458 INFO  [STDOUT                    ] CLASS of FIELD VALUE NUMBER is class java.lang.Long
```

Fig. 9: Log File - Skript-Output

Setzen von Werten in Benutzerdefinierten Feldern für Ticketdaten

Um Werte in Benutzerdefinierten Feldern für Ticketdaten zu setzen, folgen Sie dem gleichen Prinzip wie beim Abrufen von Daten: Sie benutzen den CF-Gruppennamen und den technischen Namen des CF als Referenz. Natürlich wird zusätzlich der neue Wert benötigt und dieser muss dem korrekten Datentyp angehören.

```
ticket.set("<Group_name>.<CF_name>", <value>)
```

Setzen von Werten in Benutzerdefinierten Feldern mit einfachen Datentypen

Setze CF-Wert für ein Datum-CF

```
ticket.set("fields.reaction_time", new Date());
```

Wenn Sie mit Zahlen- oder Datum-Feldern arbeiten, können Sie mit den CF-Werten auch sehr einfach rechnen, wie das folgende Beispiel zeigt:

Rechnen mit dem Wert eines Datum-CF

```
//add 24 hours (in millis) to current field value
ticket.add("fields.deadline", 24*60*60*1000);
```

Das Setzen eines Werts auf *null* (d.h. das Leeren des Felds) ist das gleiche wie das Entfernen eines Wertes:

Setzen eines CF-Werts auf null

```
ticket.set("fields.numberOfEmployees", null)
```

Oder kürzer:

Setzen eines CF-Werts auf null durch Entfernen des Werts

```
ticket.remove("fields.numberOfEmployees" )
```

Setzen von Enum-Werten

Verwenden Sie die folgende Syntax, um einen *enum*-Wert zu setzen. Natürlich muss der neue Wert in der Sortierten Liste (*enum*) vorhanden sein, die vom CF referenziert wird.

```
ticket.set("Group_name.CF_name",<technical name of value>)
```

Setzen von enum-Werten

```
ticket.set("fields.priority", "URGENT");
```

Setzen von Listen-Werten

Setzen von Werten in Listen mit einfachen Datentypen

Wenn Sie eine Zeile hinzufügen möchten, können Sie einfach die *add*-Methode verwenden:

Eine neue Zeile zu einer Liste von strings hinzufügen

```
ticket.add("fields.tags", "my new String")
```

Wenn Sie sich auf einen bestimmten Wert beziehen möchten, um für diesen einen neuen Wert zu setzen, müssen Sie die Syntax für ein *array* verwenden:

Setzen eines Werts in einer Liste von Strings

```
ticket.set("fields.tags[last]", "consol cm6")
```

Setzen von Werten in Lists of Structs

Bei der Arbeit mit Strukturen (*structs)* müssen Sie immer mit dem *key* des Wertes arbeiten, den Sie hinzufügen oder setzen möchten. Wenn Sie eine neue Zeile *hinzufügen* (*add*) möchten, müssen Sie eine neue Struktur (*struct*) als neue Zeile bauen. Die Methode *set* kann iterativ für jedes neue Feld verwendet werden.

Hinzufügen einer neuen Zeile zu einer List of Structs

```
ticket.add("order_data.orders_list", new Struct().set("tA_Id", id).set("orders_hardware",
mynewhardware_model).set("orders_contact", thenewcontactname).set("orders_number",thenewnumber)
```

Ein- und Ausblenden von Benutzerdefinierten Feldgruppen

Eine Benutzerdefinierte Feldgruppe (CF-Gruppe) kann mittels einer *workflowApi*-Methode eingeblendet (sichtbar gemacht) und ausgeblendet (unsichtbar gemacht) werden. Dies funktioniert sowohl für CF-Gruppen, welche im Abschnitt der Hauptticketdaten angezeigt werden (im Kopfbereich des Tickets), als auch für CF-Gruppen, die in Tabs angezeigt werden (im Kopfbereich des Tickets im Bereich *Gruppen*).

Ein typischer Anwendungsfall ist eine CF-Gruppe, die zuerst unsichtbar ist (CF-Gruppen-Annotation *group-visibility* = *false*) und eingeblendet wird, wenn der Bearbeiter im Prozess mit den Daten arbeiten muss. Beispielsweise wird eine CF-Gruppe, die die Gründe für das Verwerfen einer Anfrage enthält, nur angezeigt, wenn der Bearbeiter die Workflow-Aktivität *Ticket verwerfen...* verwendet hat. Dies verhindert, dass das Ticket mit Informationen überladen wird.

Einblenden einer CF-Gruppe

```
workflowApi.setGroupProperty(„CF_Group_Dismissal",GroupPropertyType.VISIBLE,"true")
```

Um eine CF-Gruppe auszublenden, z.B. wenn ein Ticket qualifiziert wurde und einige der CF-Gruppen im Prozess nicht weiter benötigt weden, verwenden Sie Code entsprechend dem folgenden Beispiel:

Ausblenden einer CF-Gruppe

```
workflowApi.setGroupProperty("CF_Group_HardwareInfo",GroupPropertyType.VISIBLE,"false")
workflowApi.setGroupProperty("CF_Group_SoftwareInfo",GroupPropertyType.VISIBLE,"false")
```

6.3.4 Datenfelder für Kundendaten

Benutzerdefinierte Felder für Kundendaten (CM-Version 6.8 und niedriger)

In CM-Version 6.8 und niedriger werden Kundendaten auf der Seite *Verwaltung von Benutzerdefinierten Feldern* im Admin-Tool definiert, im Tab *Kundendaten*.

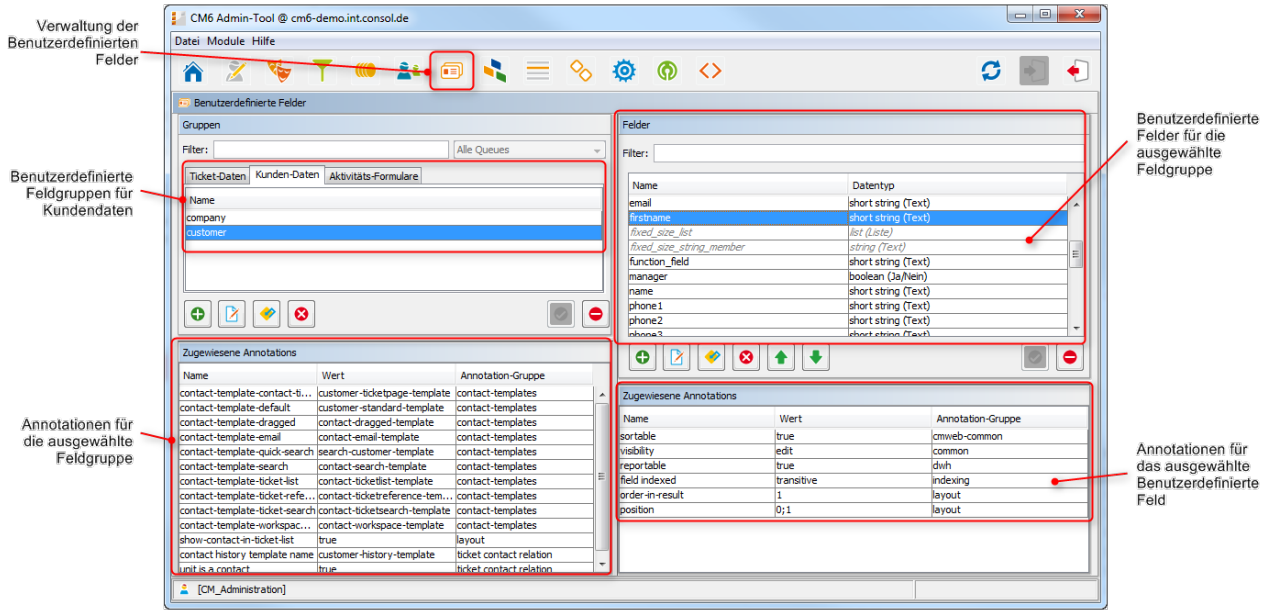


Fig. 10: ConSol*CM Admin-Tool - Verwaltung von Benutzerdefinierten Feldern für Kundendaten (CM-Version 6.8 und niedriger)

Die Kundendaten können eine Stufe (nur Kontakt-Level) oder zwei Stufen (Kontakt-Level und Firmen-Level) umfassen, d.h. Sie arbeiten mit maximal zwei Objekten. Die Namen der Objekte hängen von den Namen ab, welche ihnen im Admin-Tool zugewiesen wurden. Im Beispiel (siehe Bild oben) wurde das Kontakt-(=Kunden-)Objekt *customer* genannt und das Firmen-Objekt *company*.

Abrufen von Werten

Jedes Objekt innerhalb der Kundendaten repräsentiert eine *Unit* (d.h. eine Instanz der Klasse *Unit*). In Skripten muss die Unit (also Kunde oder Firma) abgerufen werden, bevor Sie mit ihr arbeiten können. Wenn das Kundendatenmodell zwei Stufen umfasst (Kontakt und Firma) , sehen Sie ein CF im Kontakt-Objekt, das den Datentyp *contact data reference* besitzt. Dies ist die Verbindung zwischen dem Kontakt- und dem Firmen-Objekt.

```
Unit contact = ticket.getMainContact()
```

```
Unit company = contact.get('<contact data reference_field>')
```

Für alle anderen CFs basiert der Zugriff auf Daten auf dem gleichen Prinzip wie für Ticketdaten.

```
Type t = contact.get('<CF_name>')
```

Zum Beispiel:

Abrufen von Kundendaten aus einem CF

```
def fn = customer.get("firstname")
```

Setzen von Werten für Kundendaten in CM-Version 6.8 und niedriger

```
company.set('<CF_name>', <new value>)
```

Setzen von Werten für eine Firma in einer List of Structs

```
ticket.set("person_data.responsibleConsultants", new Struct[]{
 new Struct().set("lastName", "Miller").set("email", "miller@consol.com"),
 new Struct().set("lastName", "Smith").set("email", "smith@consol.com"),
 new Struct().set("lastName", "Burger").set("email", "burger@consol.com")
 });
```

Datenobjektgruppenfelder für Kundendaten (CM-Version 6.9 und höher)

Ab CM-Version 6.9 und höher sind die Datenobjektgruppen Teil des neuen Kundendatenmodells (*FlexCDM*) und werden im Admin-Tool, auf der Seite *Benutzerattribute*, Tab *Kundendatenmodell*, definiert.

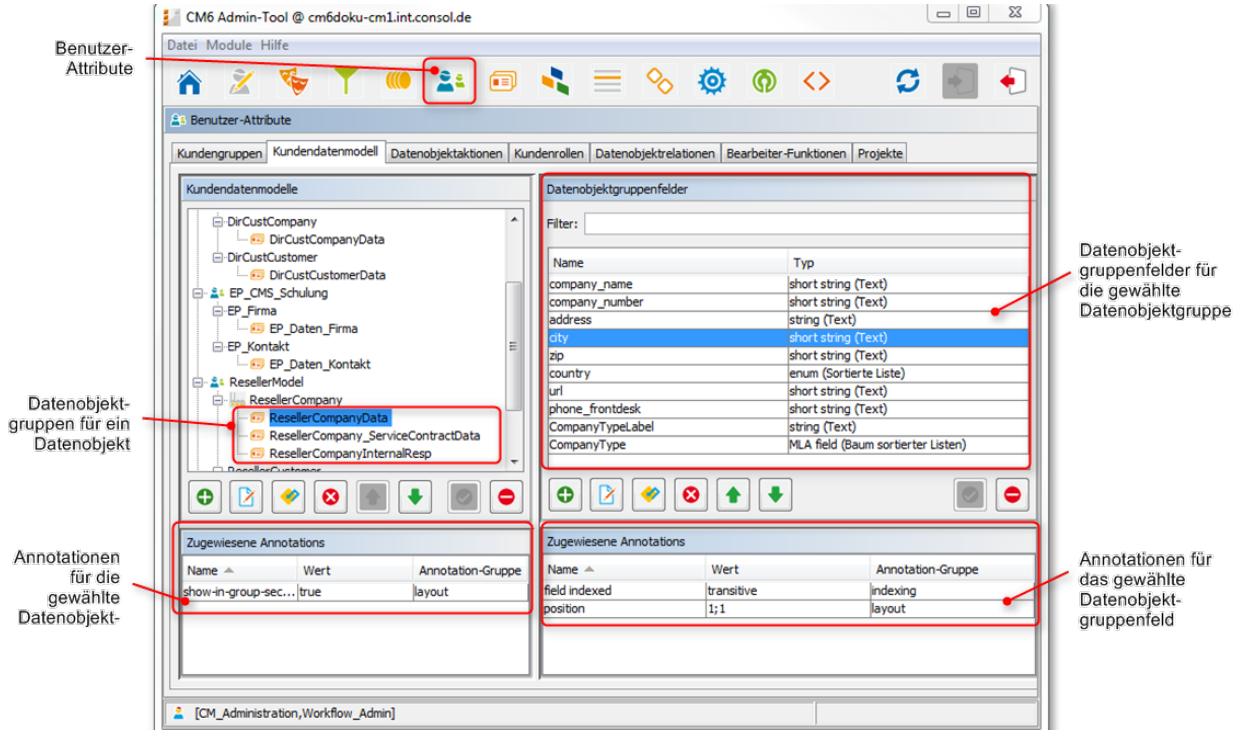


Fig. 11: ConSol*CM Admin-Tool - ConSol*CM Admin-Tool - Verwaltung von Datenobjektgruppenfeldern für Kundendaten (CM Version 6.9 and höher)

Die Felder, die im Kundendatenmodell der vorherigen CM-Versionen Benutzerdefinierte Felder genannt wurden, werden nun **Datenobjektgruppenfelder** genannt. Trotzdem bleibt das Prinzip, das SIe für das Abrufen und das Setzen von Werten der Datenfelder verwenden, prinzipiell das gleiche wie in CM-Version 6.8 und niedriger.

Die wichtigsten Methoden für den Zugriff auf Datenobjektgruppenfelder für Kundendaten

Drei Methoden sind bei der Programmierung von größter Wichtigkeit für den Zugriff auf Datenobjektgruppenfelder (*data object group fields = DOGF*) in CM-Skripten. Sie sind alle Methoden der Klasse *Unit*:

- **Unit.get()**

    - Für das Abrufen von Daten aus einem DOGF.
- **Unit.set()**
    - Für das Setzen von Daten in einem bereits existierenden DOGF.
- **Unit.add()**
    - Für das Rechnen mit einem Wert innerhalb eines DOGF, d.h. um eine bestimmte Zeitspanne zu einem *Datum*-Feld hinzuzufügen.
    - Um eine neue Zeile zu *list*-Feldern hinzuzufügen (einfache Listen und Tabellen).

Eine weitere Methode kann benutzt werden, wenn ein Feld geleert werden soll, d.h. wenn der Wert auf *null* gesetzt werden soll:

- **Unit.remove()**
    - Setzt den Wert eines Feldes auf *null*.

Abrufen von Werten für Kundendaten in CM-Version 6.9 und höher

Da der Name eines *Datenobjektgruppenfelds* in mehr als einer *Datenobjektgruppe* auftreten kann, muss der Name der Datenobjektgruppe mitgeliefert werden, wenn auf Kundendaten zugegriffen wird. Im Kundendatenmodell, das im obigen Bild gezeigt ist, können zum Beispiel die Datenobjektgruppen *ResellerCompanyData* und *DirCustCompanyData* beide das gleiche Datenobjektgruppenfeld mit dem Namen *city* besitzen. Daher ist es wichtig, den Gruppennamen und den Feldnamen zu nennen.

Bitte verwenden Sie die folgende Syntax:

```
unit.get("group1:name")
```

Zum Beispiel:

Abrufen eines Feldwerts für eine Firma

```
def mycity = company.get("ResellerCompanyData:city")
```

Es gibt verschiedene Objekte und Methoden, um mit Daten auf den verschiedenen Stufen von FlexCDM zu arbeiten. Bitte beachten Sie das folgende Beispiel, in dem einige häufige Objekte und Methoden verwendet wurden. Es handelt sich um ein Admin-Tool-Skript, auf das von einer Workflow-Aktivität aus zugegriffen wird. Sein Zweck ist es, einige Daten des Hauptkunden des Tickets anzuzeigen. Das folgende Bild zeigt die Java-Objekte, die in diesem Skript verwendet werden und die ConSol*CM-Objekte im Admin-Tool, welche im Skript referenziert werden.

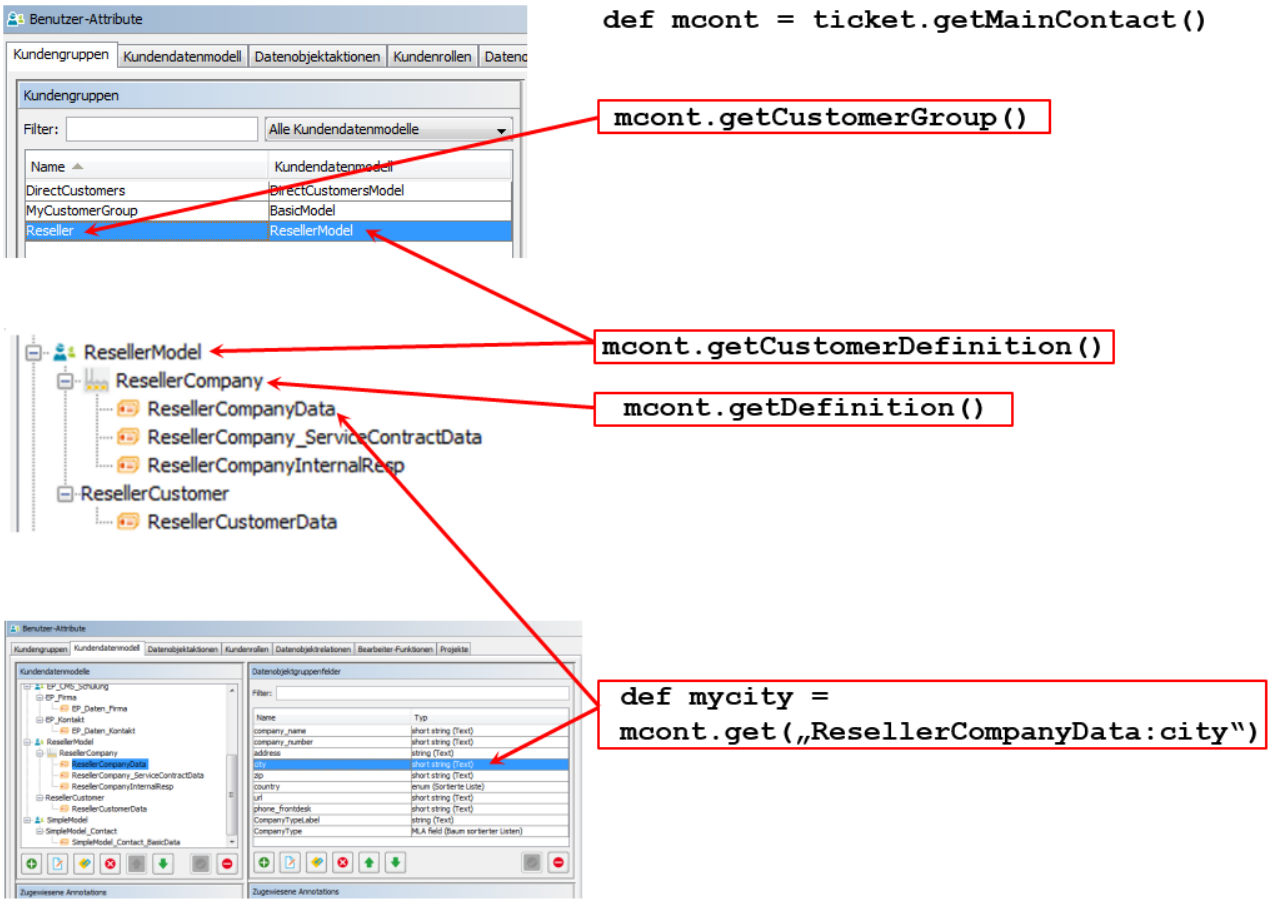


Fig. 12: ConSol*CM Kundenobjekte in Skript und Admin-Tool

Information:

Bitte beachten Sie, dass Sie auch die Kurznotation wie *unit.definition.type* für *Getter*-Methoden *wie unit.getDefinition().getType()* verwenden können.

Admin-Tool-Skript zur Anzeige von Kundendaten

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType
def ticket = workflowApi.getTicket()
def mcont = ticket.getMainContact()
println "CustomerGroup of main contact is now " + mcont.getCustomerGroup().getName()
println "Customer definition of main contact is now " + mcont.getCustomerDefinition().getName()
println "UnitDefinition of main contact is now " + mcont.getDefinition().getName()
def custmod = mcont.getCustomerDefinition().getName()
println "CUSTMOD is now " + custmod
def cityfield
switch (custmod) {
    case "BasicModel" : cityfield = "company:city";
    break;
    case "DirectCustomerModel" : cityfield = "DirCustCompanyData:dir_cust_company_city";
    break;
    case "ResellerModel": cityfield = "ResellerCompanyData:city";
    break;
}
println "CITYFIELD is now " + cityfield
def utype1 = mcont.getDefinition().getType()
def utype2 = mcont.definition.type
println "UTYPE1 is now " + utype1
println "UTYPE2 is now " + utype2
def company = mcont
if (utype2 == UnitDefinitionType.CONTACT) {
    company = mcont.get("company()")
}
def mycity = company.get(cityfield)
println " CITY is now " + mycity
```

Für das folgende Datenset wird der Log-Datei-Output unten gezeigt. Das *Reseller*-Modell aus dem obigen Bild wird verwendet.

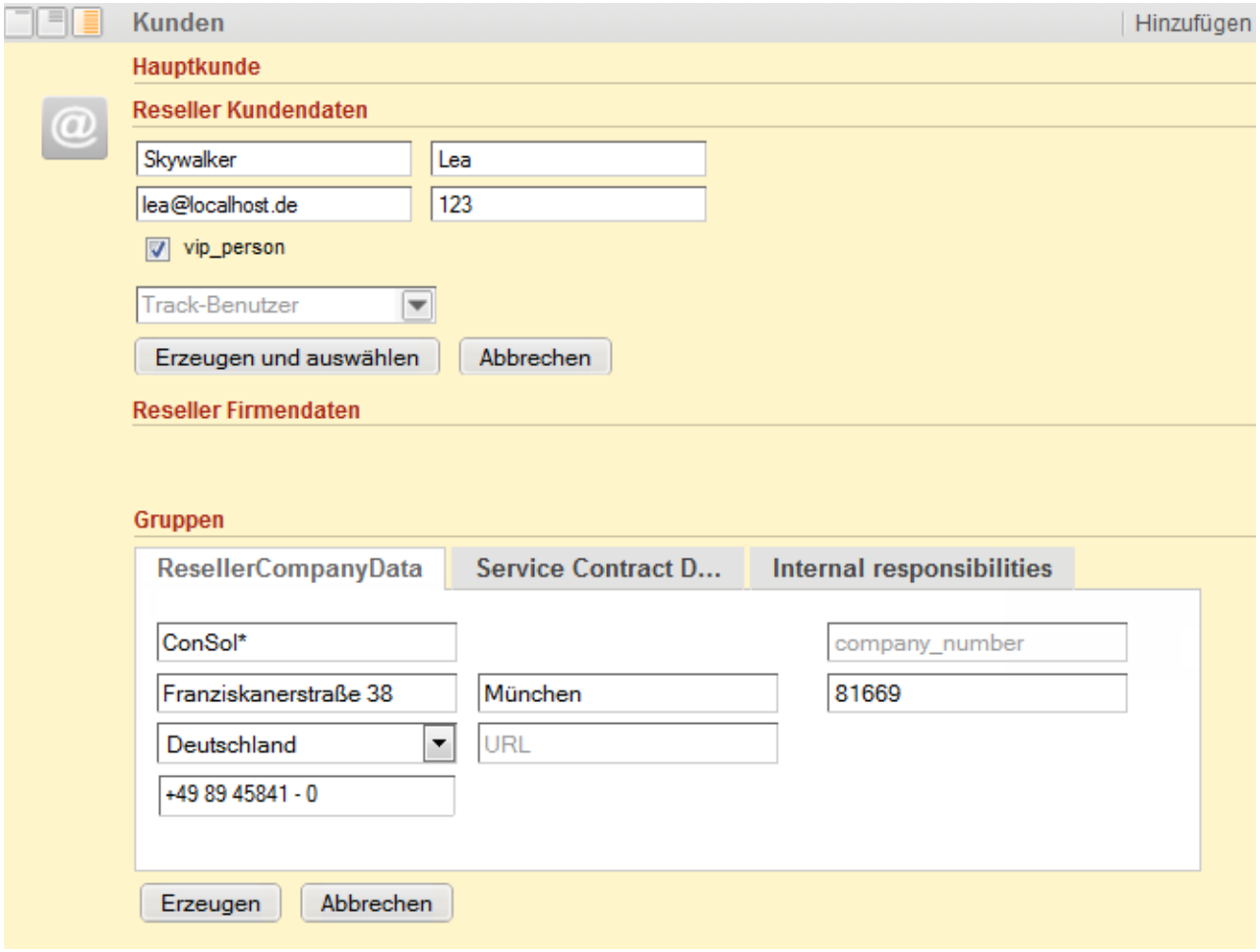


Fig. 13: ConSol*CM Web Client - Kundendatenset

```
2014-03-27 16:09:21,739 INFO  [STDOUT                    ] CustomerGroup of main contact is now Reseller
2014-03-27 16:09:21,739 INFO  [STDOUT                    ] Customer definition of main contact is now ResellerModel
2014-03-27 16:09:21,740 INFO  [STDOUT                    ] UnitDefinition of main contact is now ResellerCustomer
2014-03-27 16:09:21,743 INFO  [STDOUT                    ] CUSTMOD is now ResellerModel
2014-03-27 16:09:21,744 INFO  [STDOUT                    ] CITYFIELD is now ResellerCompanyData:city
2014-03-27 16:09:21,750 INFO  [STDOUT                    ] UTYPE1 is now CONTACT
2014-03-27 16:09:21,751 INFO  [STDOUT                    ] UTYPE2 is now CONTACT
2014-03-27 16:09:21,759 INFO  [STDOUT                    ]  CITY is now München
```

Fig. 14: Log-Datei - Skript-Output

Abrufen eines Wert aus einer List of Structs mittels Index-Notation

```
String firstName = company.get("responsibleConsultants[0].firstName");
```

Setzen von Werten für Kundendaten in CM-Version 6.9 und höher

Setzen von Werten für Datenobjektgruppenfelder mit einfachen Datentypen

Die *set*- und *add*-Methoden funktionieren, wie für die Benutzerdefinierten Felder für Tickets beschrieben. Zum Beispiel:

Setzen und Hinzufügen von Werten für Datenobjektgruppenfelder des Typs integer (Ganzzahl)

```
//Setzen des Zahlen-Felds
company.set("numberOfEmployees", 1);
//1 zum Feldwert hinzufuegen, danach betraegt der Feldwert 2
company.add("numberOfEmployees", 1);
```

Listen

Setzen von Werten in einer List of Structs für Kundendaten

Erstellen einer neuen List of Structs, Version 2

```
  company.set("responsibleConsultants", [
 new Struct().set("lastName", "Miller").set("email", "miller@consol.com"),
 new Struct().set("lastName", "Smith").set("email", "smith@consol.com"),
 new Struct().set("lastName", "Burger").set("email", "burger@consol.com")
]);
```

Hinzufügen einer neuen Zeile in einer List of Structs für Firmendaten

```
company.add("responsibleConsultants", new Struct().set("lastName", " Nowitzki ").set("email", "d
nowitzki@consol.us"));
```

Setzen eines Wers in einer List of Structs mittels Index-Notation

```
company.set("responsibleConsultants[0].firstName", "John");
```

Entfernen einer Struktur (struct = Zeile) aus einer List of Structs (= Tabelle)

```
company.set("responsibleConsultants[last]", null);
```

Convenience-Methoden für den Zugriff auf Kundendaten in CM-Version 6.9 und höher

Convenience-Methoden für den Zugriff auf Kundendaten

```
Unit mainContact = ticket.getMainContact();
// "company"-Extension gibt company fuer den contact zurueck
Unit company = mainContact.get("company()");
// es ist auch moeglich, die company mit der "company"-Extension zu setzen
mainContact.set("company()", company);
// "contacts"-Extension gibt eine Liste von contacts fuer die company zurueck
List contacts = company.get("contacts()");
// "tickets"-Extension gibt eine Liste von Tickets fuer contact oder company zurueck
List tickets = company.get("tickets()");
tickets = mainContact.get("tickets()");
// Extensions koennen miteinander verkettet werden
Integer count = contact.get("company().contacts()[0].tickets()[count]");
// Parenthese kann ausgelassen werden, dies wird aber nicht empfohlen (aufgrund von möglicher
Kollision mit Namen von Gruppe oder Feld)
count = contact.get("company.contacts[0].tickets[count]"); // hier ist "company" nicht
Extension, sondern der Name des Felds
```

6.3.5 Datenfelder für (unsichtbare) Variablen nutzen

Manchmal kann es notwendig sein, mit Variablen zu arbeiten, welche nicht als Werte für in der Web-Client-GUI sichtbare Benutzerdefinierte Felder oder Datenobjektgruppenfelder verwendet werden, sondern nur als Container für interne Programmiervariablen verwendet werden.

Diejenigen, die sich mit der Programmierung von ConSol*CM5-Workflows auskennen, kennen diese Container als *Globale Variablen*. In ConSol*CM6 können Sie das gleiche Ziel erreichen, indem Sie ein normales Benutzerdefiniertes Feld (für Ticketdaten) oder ein Datenobjektgruppenfeld (für Kundendaten) erstellen, das den erforderlichen Datentyp besitzt, und das Feld auf *invisible* setzen. Dies muss über die Annotation *visibility = none* geschehen.Sie können die Variable auch während der Prozessentwicklung sichtbar lassen, um den Wert des Felds kontrollieren zu können. Bevor Sie das System dann an QA und Nutzer übergeben, setzen Sie die Variable auf unsichtbar.

6.4 Senden von E-Mails

- Einleitung zum Senden von E-Mails
- Wichtige Methoden
  - ConSol*CM-Version 6.8 und niedriger
  - ConSol*CM-Version 6.9 und höher
- Beispiele
  - Senden einer automatischen Empfangsbestätigung an den Kunden, wenn dieser ein Ticket eröffnet hat
    - ConSol*CM-Version 6.8 und niedriger
    - ConSol*CM-Version 6.9 und höher
  - Senden einer E-Mail an den Bearbeiter, wenn eine bestimmte Eskalationsstufe erreicht wurde
    - ConSol*CM-Version 6.8 und niedriger
    - ConSol*CM Version 6.9 und höher
      - Senden einer E-Mail an einen Kunden unter Einbeziehung des Queue-spezifischen E-Mail-Skripts
  - Senden einer E-Mail an alle Kontakte eines Tickets
  - Senden einer E-Mail an jeden Kontakt aus der Liste aller Kontakte eines Tickets

6.4.1 Einleitung zum Senden von E-Mails

Die Fähigkeit, E-Mails empfangen und senden zu können, ist eine Kernfunktionalität von ConSol*CM. Im *ConSol*CM Administratorhandbuch* können Sie eine detaillierte Einführung dazu nachlesen.

In diesem Abschnitt wird beschrieben, wie Sie Skripte schreiben, um E-Mails aus einem Workflow heraus zu senden. Dies ist für Anwendungsfälle wie die folgenden sehr nützlich:

- Sie möchten eine automatische Empfangsbestätigung an den Kunden senden, wenn er sein Ticket eröffnet hat.
- Sie möchten den Bearbeiter und seinen Supervisor benachrichtigen, wenn eine bestimmte Eskalationsstufe erreicht ist.
- Sie möchten den Kunden informieren, dass ein Problem gelöst wurde (und wie).

Üblicherweise schreiben Sie den Text einer E-Mail nicht in das Skript, sondern arbeiten mit E-Mail-Templates. Lesen Sie daher bitte zuerst die detaillierte Einführung in den *ConSol*CM Template Designer* im *ConSol*CM Administratorhandbuch*.

6.4.2 Wichtige Methoden

ConSol*CM-Version 6.8 und niedriger

Verwenden Sie *workflowApi.sendEmail().*

ConSol*CM-Version 6.9 und höher

Verwenden Sie ein Objekt der Klasse *Mail*.

Hier können Sie alle benötigten Parameter für eine E-Mail definieren und das das *Mail*-Objekt dahingehend zu konfigurieren, das Queue-spezifische E-Mail-Standardskript zu verwenden. Dies ist ein Skript, das die E-Mail verarbeitet, bevor es das CM-System verlässt. Diese Art von Skript kann einer Queue zugewiesen werden (*E-Mail-Skript*, siehe Abschnitt *Queue-Verwaltung* im *ConSol*CM Administratorhandbuch*). Die Verwendung eines solchen Skripts kann hilfreich sein, wenn Sie zum Beispiel eine *REPLY TO*-Adresse setzen möchten, die von der Standard *REPLY TO*-Adresse abweicht (die in einer System-Property gespeichert ist). Lesen Sie dazu bitte die detaillierte Ausführung im *ConSol*CM Administratorhandbuch*.

6.4.3 Beispiele

Senden einer automatischen Empfangsbestätigung an den Kunden, wenn dieser ein Ticket eröffnet hat

ConSol*CM-Version 6.8 und niedriger

Dieses Skript kann in eine der ersten Aktivitäten des Workflows eingefügt werden.

```
// hole den Hauptkontakt des Tickets
def contact = ticket.getMainContact()
// hole die E-Mail-Adresse = Benutzerdefiniertes Feld des Kontakts
def contact_e = contact.get("email")
// erstelle den E-Mail-Text mittels eines Templates, das im Template Designer gespeichert ist
def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")
// hole die REPLY TO-Adresse, die in einer System-Property gespeichert ist
def replyto = configurationService.getValue("cmweb-server-adapter","mail.reply.to")
// setze den Betreff der E-Mail, die Ticketnummer mit dem korrekten Regulaeren Ausdruck
// muss gesetzt werden, um eintreffende E-Mails für das Ticket korrekt zuordnen zu können
def subj = "Ihr Vorgang wurde registriert als Ticket (" + ticket.getId() + ")"
// versende die E-Mail
workflowApi.sendEmail(contact_e,subj,text,replyto,null)
```

ConSol*CM-Version 6.9 und höher

Dieses Skript kann in eine der ersten Aktivitäten des Workflows eingefügt werden.

```
// erstelle ein neues Mail-Objekt
def mail = new Mail()
// hole den Hauptkontakt des Tickets
def maincontact = ticket.getMainContact()
// hole die E-Mail-Adresse des Hauptkontakts. Das Datenobjektgruppenfeld muss mittels
Datenobjektgruppennamen:Datenobjektgruppenfeldname angesprochen werden
def toaddress = maincontact.get("MyCustomerDataObjectGroup:email")
// setze die E-Mail TO-Adresse in das Mail-Objekt
mail.setTo(toaddress)
// hole die REPLY TO-Adresse, die in einer System-Property gespeichert ist
def replyaddress = configurationService.getValue("cmweb-server-adapter","mail.reply.to")
// setze die E-Mail REPLY TO-Adresse in das Mail-Objekt
mail.setReplyTo(replyaddress)
// erstelle den E-Mail-Text mittels eines Templates, das im Template Designer gespeichert ist
def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")
// setze den E-Mail-Text in das Mail-Objekt
mail.setText(text)
// setze den Betreff der E-Mail, die Ticketnummer mit dem korrekten Regulaeren Ausdruck, muss
gesetzt werden, um eintreffende E-Mails für das Ticket korrekt zuordnen zu können
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"
// setze den Betreff in das Mail-Objekt
mail.setSubject(subject)
// versende die E-Mail
mail.send()
```

Senden einer E-Mail an den Bearbeiter, wenn eine bestimmte Eskalationsstufe erreicht wurde

Dieses Skript kann in eine automatische Aktivität, die mit einem Zeit-Trigger verbunden ist, eingefügt werden. Der Zeit-Trigger misst das Eskalationsintervall. Wenn die Deadline erreicht wurde, feuert der Trigger und das Ticket tritt in die automatische Aktivität ein.

ConSol*CM-Version 6.8 und niedriger

```
// hole den current engineer des Tickets
def eng = ticket.getEngineer()
// hole die E-Mail-Adresse = Standard-Datenfeld von engineer, überprüfe, ob es einen current
engineer gibt, um eine NullPointerException zu vermeiden
def  eng_email = eng?.getEmail()
// erstelle den E-Mail-Text mittels eines Templates, das im Template Designer gespeichert ist
def text = workflowApi.renderTemplate("ESCALATION_Mail")
// hole die REPLY TO-Adresse, die in einer System-Property gespeichert ist
def replyto = configurationService.getValue("cmweb-server-adapter","mail.reply.to")
// setze den Betreff der E-Mail, die Ticketnummer mit dem korrekten Regulaeren Ausdruck, muss
gesetzt werden, um eintreffende E-Mails für das Ticket korrekt zuordnen zu können
def subj = "ESCALATION  Level 3 REACHED! Ticket (" + ticket.getId() + ")"
// versende die E-Mail
workflowApi.sendEmail(eng_email,subj,text,replyto,null)
```

ConSol*CM Version 6.9 und höher

```
// erstelle ein neues Mail-Objekt
def mail = new Mail()
// hole den current engineer des Tickets und setzte ihn als E-Mail-Empfaenger
if (ticket.engineer){
    mail.setTargetEngineer(ticket.engineer)
    // hole die REPLY TO-Adresse, die in einer System-Property gespeichert ist
    def replyaddress = configurationService.getValue("cmweb-server-adapter","mail.reply.to")
    // setze die E-Mail REPLY TO-Adresse in das Mail-Objekt
    mail.setReplyTo(replyaddress)
    // erstelle den E-Mail-Text mittels eines Templates, das im Template Designer gespeichert
ist
    def text = workflowApi.renderTemplate("ESCALATION_Mail")
    // setze den E-Mail-Text in das Mail-Objekt
    mail.setText(text)
    // erstelle den Betreff der E-Mail, die Ticketnummer mit dem korrekten Regulaeren Ausdruck,
muss gesetzt werden, um eintreffende E-Mails für das Ticket korrekt zuordnen zu können
    def ticketname = ticket.getName()
    def subject = "ESCALATION  Level 3 REACHED! Ticket (" + ticket.getId() + ")"
    // setze den Betreff in das Mail-Objekt
    mail.setSubject(subject)
    // versende die E-Mail
    mail.send()
}
```

Senden einer E-Mail an einen Kunden unter Einbeziehung des Queue-spezifischen E-Mail-Skripts

Dies ist das gleiche Skript wie im obigen Beispiel, nur dass das Queue-spezifische E-Mail-Skript verwendet wird. Eine detaillierte Erklärung dieses Skripttyps entnehmen Sie bitte dem *ConSol*CM Administratorhandbuch*, Abschnitt *Admin-Tool-Skripte*.

Dies bewirkt, dass die ausgehende E-Mail das Skript durchläuft, bevor sie das CM-System verlässt. E-Mail-Parameter wie *CC*, *BCC*, oder *REPLY TO* können dabei verändert werden.

```
// erstelle ein neues Mail-Objekt
def mail = new Mail()
// hole den Hauptkontakt des Tickets
def maincontact = ticket.getMainContact()
// hole die E-Mail-Adresse des Hauptkontakts. Das Datenobjektgruppenfeld muss mittels
Datenobjektgruppennamen:Datenobjektgruppenfeldname angesprochen werden
def toaddress = maincontact.get("MyCustomerDataObjectGroup:email")
// setze die E-Mail TO-Adresse in das Mail-Objekt
mail.setTo(toaddress)
// hole die REPLY TO-Adresse, die in einer System-Property gespeichert ist
def replyaddress = configurationService.getValue("cmweb-server-adapter","mail.reply.to")
// setze die E-Mail REPLY TO-Adresse in das Mail-Objekt
mail.setReplyTo(replyaddress)
// erstelle den E-Mail-Text mittels eines Templates, das im Template Designer gespeichert ist
def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")
// setze den E-Mail-Text in das Mail-Objekt
mail.setText(text)
// erstelle den Betreff der E-Mail, die Ticketnummer mit dem korrekten Regulaeren Ausdruck,
muss gesetzt werden, um eintreffende E-Mails für das Ticket korrekt zuordnen zu können
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"
// setze den Betreff in das Mail-Objekt
mail.setSubject(subject)
// die E-Mail soll das E-Mail-Skript, welches für die Queue konfiguriert ist, nutzen
mail.useDefaultScript()
// versende die E-Mail
mail.send()
```

Senden einer E-Mail an alle Kontakte eines Tickets

Dies versendet eine E-Mail, deren Empfänger alle Kontakte (die eine E-Mail-Adresse besitzen) eines Tickets sind. Bitte beachten Sie, dass dies ein einfaches Beispiel, das die Verwendung einer Liste demonstriert, darstellt. Die REPLY-TO-Adresse wird nicht gesetzt, daher würden Antworten auf die E-Mail nicht an das Ticket angehängt werden.

```
def custEmails = workflowApi.getContactList()*.get("email").findAll{it != null}.join(",")
workflowApi.sendEmail(custEmails, "Bestätigung", "Guten Tag, wir haben Ihre Anfrage erhalten!",
null, null)
```

Senden einer E-Mail an jeden Kontakt aus der Liste aller Kontakte eines Tickets

Dies versendet ein jeden einzelnen Kunden (der eine E-Mail-Adresse besitzt) eine eigene E-Mail. Bitte beachten Sie, dass dies ein einfaches Beispiel, das die Verwendung einer Liste demonstriert, darstellt. Die REPLY-TO-Adresse wird nicht gesetzt, daher würden Antworten auf die E-Mail nicht an das Ticket angehängt werden.

```
workflowApi.getContactList().each {
  def custEmail = it.get("email")
  if (custEmail) workflowApi.sendEmail(custEmail, "Bestätigung", "Guten Tag, wir haben Ihre
Anfrage erhalten!", null, null)
}
```

6.5 Arbeiten mit Pfadinformationen

- Einleitung
- Abrufen von Pfadinformationen eines Workflow-Elements
- Beispiele für die Verwendung von Pfadinformationen
  - Beispiel 1: Deaktivieren und/oder Re-Initialisieren eines Zeit-Triggers

6.5.1 Einleitung

Wie in einem Dateisystem eines Computers, kann auch jedes Element eines Workflows über den Pfad dieses Elements angesprochen werden. Dies kann notwendig sein, wenn Sie mit dem Element innerhalb eines Workflow-Skripts arbeiten möchten. Ein Pfad repräsentiert die hierarchische Struktur eines Elements innerhalb des Workflows.

> Die Pfadinformationen beziehen sich auf die technischen Namen der Workflow-Elemente, nicht auf deren internationalisierte Bezeichnung. Da im nachfolgenden Beispiel englische technische Namen vergeben wurden, enthalten die Pfadinformationen englischen Namen und unterscheiden sich von den (deutschen, internationalisierten) Bezeichnungen, die auf der grafischen Oberfläche zu sehen sind.

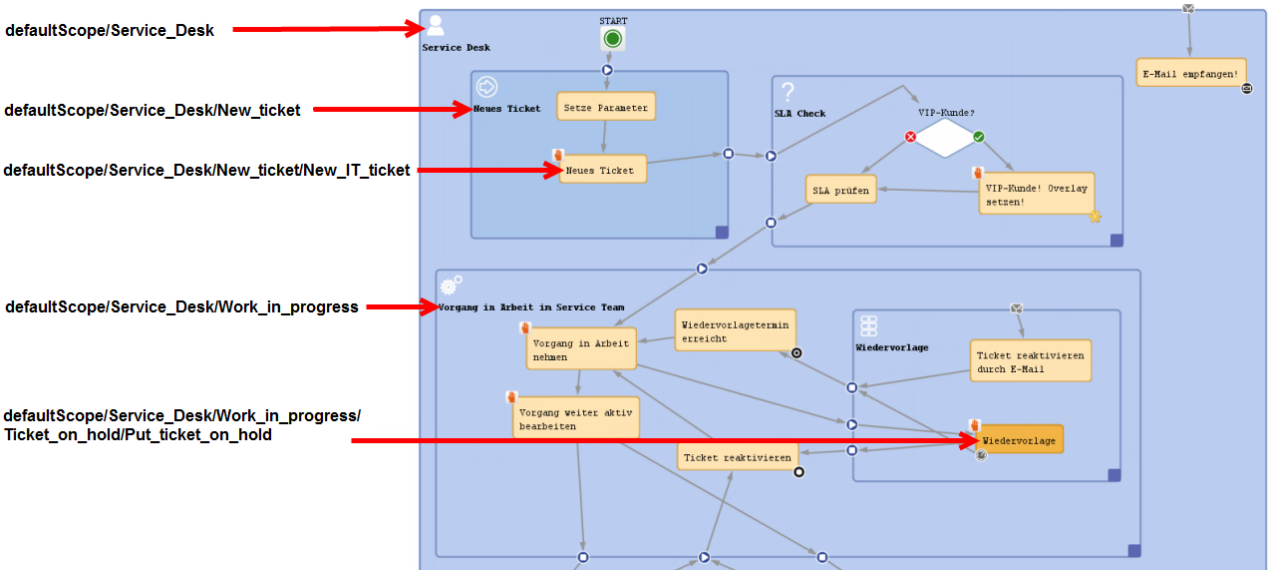


Fig. 1: ConSol*CM Process Designer - Pfadinformationen (Beispiel: Aktivitäten und Scopes)

6.5.2 Abrufen von Pfadinformationen eines Workflow-Elements

Sie können den Pfad eines Elements kopieren, indem Sie mit der rechten Maustaste auf das Element klicken und *Kopiere Pfad des Knoten (bzw. Adornments) in die Zwischenablage* auswählen.

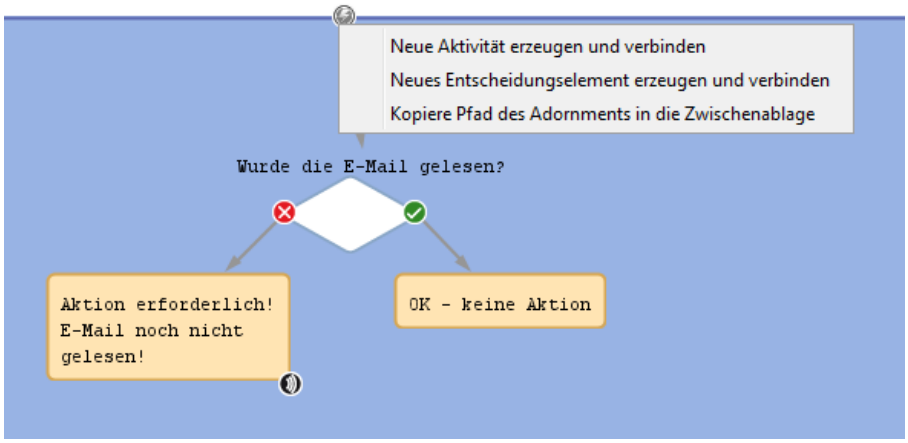


Fig. 2: ConSol*CM Process Designer - Kopieren des Pfades eines Workflow-Elements

6.5.3 Beispiele für die Verwendung von Pfadinformationen

Beispiel 1: Deaktivieren und/oder Re-Initialisieren eines Zeit-Triggers

Ein typischer Fall für die Verwendung von Pfadinformationen ist die Re-Initialisierung eines Zeit-Triggers, wenn Sie z.B. die Zeit messen möchten, nachdem eine E-Mail empfangen wurde und sicherstellen möchten, dass sich innerhalb von maximal 10 Minuten um diese E-Mail gekümmert wird. Dies bedeutet, dass Sie den Zeit-Trigger immer wieder erneut verwenden und ihn nach jeder E-Mail, die vom Ticket empfangen wurde, re-initialisieren.

Wenn ein Ticket erstellt ist, muss der Zeit-Trigger deaktiviert werden. Dafür wird der folgende Code verwendet:

Deaktivierung eines Zeit-Triggers

```
workflowApi.deactivateTimer("defaultScope/Service_Desk/TimeTrigger1")
```

Wenn eine E-Mail empfangen wurde, muss der Trigger re-initialisiert werden. Dafür wird der folgende Code verwendet:

Re-Initialisierung eines Zeit-Triggers

```
workflowApi.reinitializeTrigger("defaultScope/Service_Desk/TimeTrigger1")
```

6.6 Arbeiten mit Kalendern und Zeiten

- Einleitung
- Rechnen mit Daten und Zeiten ohne CM-Arbeitszeitkalender
  - Beispiel: Setzen der Zeit eines Zeit-Triggers mit dynamischer Zeitperiode
- Rechnen mit Daten und Zeiten mit CM-Arbeitszeitkalender
  - Beispiel: Verwendung eines Zeit-Triggers mit einem Arbeitszeitkalender, um die Eskalationszeit zu berechnen (CM 6.9)

6.6.1 Einleitung

Die Berechnung von Daten und Zeiten spielt eine wichtige Rolle in der ConSol*CM-Workflow-Programmierung. Für einen Zeit-Trigger (siehe Abschnitt Zeit-Trigger) kann der exakte Zeitpunkt, an dem der Trigger feuern soll, durch ein Skript bestimmt werden. Dies schafft verschiedene zusätzliche Möglichkeiten der Steuerung von Eskalationszeiten, Erinnerungen für Bearbeiter und andere *aktive* Komponenten eines ConSol*CM-Prozesses. Beispiele für potentielle Kalkulationen mit Daten und/oder Zeiten sind:

- Eskalationsdaten mit Zeit-Triggern
- *Datum*-Felder, wie z.B. eine gewünschte (oder notwendige) Deadline

Wenn Sie ein Datum und/oder eine Zeit berechnen, müssen Sie entscheiden, ob ein Arbeitszeitkalender verwendet werden soll oder nicht. Ein Arbeitszeitkalender definiert Arbeitszeiten (z.B. Servicezeiten) für einen Prozess. Er wird im Admin-Tool definiert und einer oder mehreren Queue(s) zugewiesen.

Zum Beispiel könnte ein Servicedesk-Team Servicezeiten von 8 bis 18 Uhr an 6 Tagen in der Woche haben, während das Administrationsteam auf einer 9-to-5-Basis an 5 Tagen in der Woche arbeitet. Die Verwendung eines Arbeitszeitkalender stellt sicher, dass Eskalationen nicht während der Freizeit auftreten und dass die Nicht-Arbeitsstunden nicht in die Berechnung der verstrichenen Eskalationszeit einbezogen werden. Bitte lesen Sie das *ConSol*CM Administratorhandbuch* für eine detaillierter Einleitung zu den Arbeitszeitkalendern.

Auf der anderen Seite gibt es Fälle, in denen ein Arbeitszeitkalender nicht benötigt wird, sondern die *absolute* Zeit, basierend auf dem normalen Kalender, verwendet werden soll. Zum Beispiel, wenn ein Kunde drei Wochen nach dem Erstkontakt erneut kontaktiert werden soll. Die folgenden Absätze zeigen Ihnen Beispiele für beide Anwendungsfälle.

Wie die Zeit eines Zeit-Triggers mit Arbeitszeitkalender berechnet wird:

1 Tag bedeutet 24 Stunden absoluter Zeit, dies hat nichts mit der Verwendung eines Kalenders zu tun. Der Kalender spielt erst dann eine Rolle, wenn der Zeit-Trigger aktiviert ist, dann werden die 24 Stunden, d.h. 86400000 Millisekunden, als Arbeitszeitkalender-Input genommen (wenn der Kalender aktiviert ist).

Beispiel:
Bei einem Zeit-Trigger mit 1 Tag = 24 Stunden ohne Kalender werden die 24 Stunden wie normale Zeit berechnet, so dass sie Eskalation einen Tag später um die gleiche Zeit feuert.
Im Gegensatz dazu: Wenn wir einen Kalender verwenden (mit z.B. 7 Arbeitsstunden pro Arbeitstag), werden die 24 Stunden dem Kalender entsprechend aufgesplittet, was dazu führt, dass der Trigger mehr als drei Tage später feuert (24 Stunden = 3 x 7 Stunden + 3 Stunden)

6.6.2 Rechnen mit Daten und Zeiten ohne CM-Arbeitszeitkalender

Beispiel: Setzen der Zeit eines Zeit-Triggers mit dynamischer Zeitperiode

Der Zeit-Trigger für eine Eskalation ist abhängig von der Priorität konfiguriert:

Setzen der Zeit für einen Zeit-Trigger

```
// prio ist 'medium'
def escalationTime = configurationService.getValue("custom-mycompany-properties","escalation.
time.medium")
def escalationTimeMillisecs = escalationTime * 60 * 1000L
trigger.setDueTime( escalationTimeMillisecs )
```

6.6.3 Rechnen mit Daten und Zeiten mit CM-Arbeitszeitkalender

Beispiel: Verwendung eines Zeit-Triggers mit einem Arbeitszeitkalender, um die Eskalationszeit zu berechnen (CM 6.9)

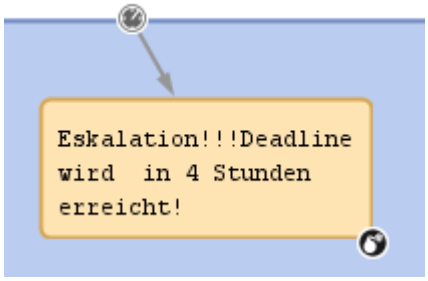


Fig. 1: ConSol*CM Process Designer - Zeit-Trigger für Eskalation 4 Stunden vor der Deadline

Skript für Zeit-Trigger für eine Eskalation 4 Stunden vor der Deadline

```
def deadl = ticket.get("serviceDesk_fields.desiredDeadline")
// 4 Stunden vor der Deadline soll die Eskalation gesetzt werden
// Arbeitszeitkalender soll benutzt werden
// ServiceDeskCalendar ist der Queue ServiceDesk zugewiesen, dies ist hier transparent
def now = new Date()
// time required in millisecds
def four_hours = -4*60*60*1000L
// berechne das Eskalationsdatum
def escalDate = BusinessCalendarUtil.getBusinessTime(deadl, four_hours, ticket.queue.calendar)
// berechne und setze Zeit des Ablaufs
def dueTime = escalDate.time - now.time
trigger.setDueTime(dueTime)
```

6.7 ConSol*CM Process Designer Handbuch - Arbeiten mit Objektrelationen

6.7.1 Arbeiten mit Objektrelationen

In ConSol*CM können Sie mit zwei Typen von Relationen arbeiten:

Relationstyp	Erklärung
Ticketrelationen	Hierarchische oder einstufige Beziehung zwischen zwei Tickets, siehe Abschnitt Arbeiten mit Ticketrelationen
Kundenrelationen	Relationen zwischen Kundendatenobjekten, z.B. Kontakten und Firmen, siehe Abschnitt Arbeiten mit Kundenrelationen (Datenobjektrelationen)

6.7.2 Arbeiten mit Ticketrelationen

- Einleitung
- Einfache Ticketrelation ohne Hierarchie
  - Beispiel: Erstellen einer einfachen Relation zwischen zwei Tickets
- Master-Slave-Relationen
  - Beispiel: Erstellen einer Master-Slave-Relation zwischen zwei Tickets
  - Syntax: Finden aller Slave-Tickets
- Parent-Child-Relationen
  - Beispiel 1: Erstellen eines neuen Child-Tickets als Child des Current Tickets
  - Beispiel 2: Finden des Parent-Tickets eines Tickets
  - Beispiel 3: Finden aller Child-Tickets eines Tickets
  - Beispiel 4: Finde alle Brother-Tickets (andere Child-Tickets) des gleichen Parent-Tickets
- Wichtige Methoden für die Arbeit mit Ticketrelationen

Einleitung

Relationen zwischen Tickets können Ihnen auf sehr effiziente Weise dabei helfen, Ihre Business-Prozesse zu modellieren.

ConSol*CM bietet drei Typen dieser Relationen an:

- **Einfache Ticketrelationen**
  Nicht-hierarchische, einfache Referenz. Jedes Ticket kann eine beliebige Anzahl von Referenzen besitzen.
  Eine einfache Ticketrelation kann durch einen Bearbeiter im Web Client oder durch einen Programmierer mittels der ConSol*CM-API erstellt werden.
  In beiden Fällen kann eine Referenz nur zwischen zwei existierenden Tickets erstellt werden.
- **Master-Slave-Relationen**
  Hierarchisch. Ein Master-Ticket kann mehrere Slave-Tickets besitzen. Ein Slave-Ticket besitzt immer genau ein Master-Ticket.
  Diese Konstruktion kann durch einen Bearbeiter im Web Client oder durch einen Programmierer mittels der ConSol*CM-API erstellt werden.
  Eine Master-Slave-Relation kann nur zwischen zwei existierenden Tickets erstellt werden, d.h. beide Tickets müssen zuerst existieren, erst danach kann eine Master-Slave-Relation zwischen ihnen aufgebaut werden, um sie zu verbinden.
- **Parent-Child-Relations**
  Hierarchisch. Ein Parent-Ticket kann mehrere Child-Tickets besitzen. Ein Child-Ticket besitzt immer genau ein Parent-Ticket.
  Diese Konstruktion kann nur durch einen Programmierer mittels der ConSol*CM-API erstellt und verändert werden.
  Eine Parent-Child-Relation kann zwischen zwei existierenden Tickets erstellt werden. Es kann auch ein neues Ticket während des Prozesses erstellt werden.

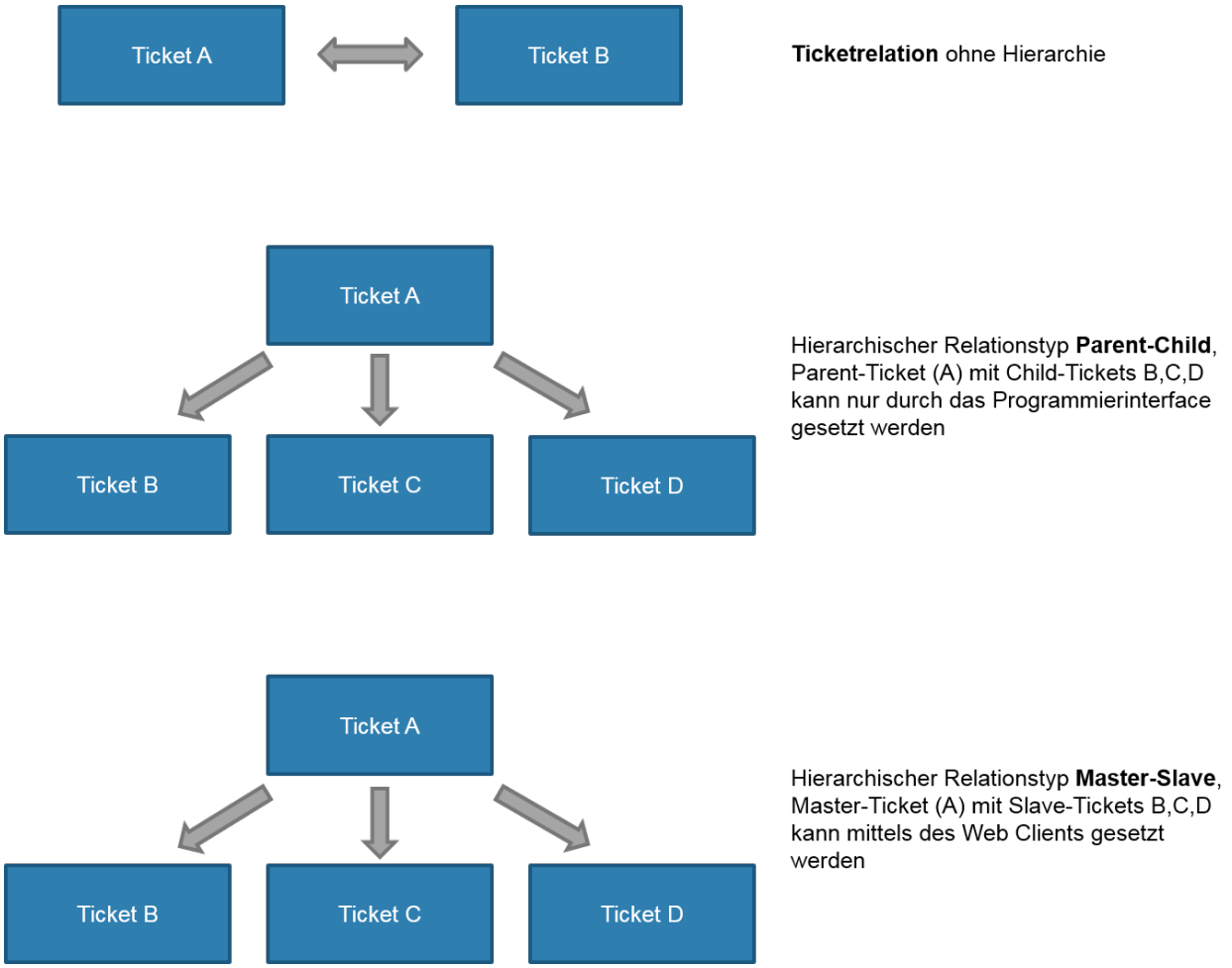


Fig. 1: ConSol*CM Relationstypen

In diesem Abschnitt wird nicht erklärt, wie Ticketrelationen mittels des Web Clients erstellt werden können, sondern es wird gezeigt, wie Relationen mittels der ConSol*CM API erstellt werden können, genauer gesagt durch Workflow-Skripte.

In der ConSol*CM Workflow API wird der Referenztyp durch die Klasse (enum) *com.consol.cmas.common.model.ticket.TicketRelationType* repräsentiert. Diese bietet drei Werte:

- REFERENCE
- MASTER_SLAVE
- PARENT_CHILD

Einfache Ticketrelation ohne Hierarchie

Dieser Relationstyp kann hilfreich sein, wenn Sie Relationen erstellen möchten, die dabei helfen, die Tickets, die in einer Relation zu einem Ticket stehen, einfacher als mittels der Suchfunktion zu finden.

Beispiel-Anwendungsfälle sind:

- Wenn ein neues Ticket erstellt wird, möchten Sie sehen, ob bereits andere offene Tickets von demselben Kunden existieren. Falls ja, erstellen Sie eine Relation zwischen diesen Tickets. Auf diese Weise kann ein Bearbeiter sehr einfach von einem offenen Ticket des Kunden zu einem seiner anderen offenen Tickets springen.
- Wenn ein Ticket für eine bestimmte Hardware-Kategorie erstellt wird, möchten Sie eine Relation zu allen anderen Tickets, die den gleichen Hardware-Typ besitzen, erstellen.

Diese Relationstyp kann entweder im Web Client oder mittels der ConSol*CM API in Scripten erstellt und verändert werden. Daher kann eine Relation vom Typ *REFERENCE* durch ein Workflow-Skript erstellt werden und danach von einem Bearbeiter im Web Client verändert werden, vorausgesetzt, dieser besitzt die dafür benötigten Berechtigungen.

Beispiel: Erstellen einer einfachen Relation zwischen zwei Tickets

Erstellen einer Ticketrelation vom Typ REFERENCE mittels workflowAPI

```
workflowApi.addRelation(TicketRelationType.REFERENCE, "Dies ist eine sehr wichtige Relation",
pSourceTicketId, pTargetTicketId)
```

Master-Slave-Relationen

Dieser Relationstyp kann hilfreich sein, wenn Sie eine Hierarchie zwischen einer bestimmten Anzahl von existierenden Tickets erstellen möchten. Denken Sie daran, dass dieser Relationstyp sowohl über den Web Client als auch über die Programmieroberfläche erstellt werden kann. An dieser Stelle wird jedoch nur die Erstellung mittels Programmierung erklärt.

Beispiel-Anwendungsfälle sind:

- In einer Firma gibt es mehrere Projekte, welche jeweils durch Tickets repräsentiert werden. Wenn eine Entscheidung getroffen wurde, eines dieser Projekte in ein übergeordnetes Programm (ebenfalls durch ein Ticket repräsentiert) zu integrieren, verwendet der Projektmanager die Workflow-Aktivität *In Programm integrieren*. Dort muss das korrekte Programm ausgewählt werden (z.B. mittels eines ACF). Im Skript der Workflow-Aktivität *In Programm integrieren* wird das Programm-Ticket als *Master*-Ticket des aktuellen Projekt-Tickets gesetzt.
- In einem Serviceteam werden Tickets für verschiedene Produkte verwaltet. Für jedes Produkt existiert ein Produkt-Ticket. Wenn ein neues Service-Ticket eröffnet wird, verwendet der Bearbeiter die Aktivität *Produkt auswählen*, in der er das betreffende Produkt aus einem Drop-Down-Menü auswählen kann. Im Workflow-Skript der Aktivität *Produkt auswählen* wird das Service-Ticket automatisch als *Slave* für das Produkt-Ticket gesetzt.

Vorsicht:

Eine Master-Slave-Relation kann sowohl über den Web Client als auch über die ConSol*CM-API in Skripten erstellt und verändert werden kann. Daher kann eine Relation vom Typ *MASTER_SLAVE* mittels eines Workflow-Skripts erstellt werden und danach von einem Bearbeiter im Web Client verändert werden, sofern dieser die erforderlichen Berechtigungen besitzt. Wenn Sie sicherstellen möchten, dass kein Bearbeiter die Ticket-Hierarchie verändern kann, verwenden Sie bitte die *Parent-Child*-Konstruktion.

Beispiel: Erstellen einer Master-Slave-Relation zwischen zwei Tickets

Erstellen einer Ticketrelation vom Typ MASTER_SLAVE mittels workflowAPI

```
//in diesem Skript wird das Projekt-Ticket (=current ticket) als Slave-Ticket zum Programm-
Ticket gesetzt
// welches das Master-Ticket wird
// hole die Programm-Ticket-ID. Die ID des Programm-Tickets ist bereits gespeichert als Custom
Field im Ticket
def progTicketId = ticket.get("ReferencesFields.ProgramTicketId")
// hole die ID des current ticket (welches das Slave-Ticket werden wird)
def mySlaveProjectId = ticket.id
workflowApi.addRelation(TicketRelationType.MASTER_SLAVE, "Slave-Ticket: Dieses Projekt ist Teil
des Programms, welches im Master-Ticket angegeben ist", progTicketId, mySlaveProjectId)
```

Syntax: Finden aller Slave-Tickets

Version A: Finden aller Ziel-Tickets (hier: aller Slave-Tickets)

```
// das Ticket kann gesetzt werden, kann das current ticket oder ein anderes Ticket sein
List<Ticket> mytickets = workflowApi.getTargetTickets(myTicket.getId(), TicketRelationType.
MASTER_SLAVE)
```

Version B: Finden aller Ziel-Tickets (hier: aller Slave-Tickets)

```
// verwendet für current ticket
List<Ticket> mytickets = workflowApi.getTargetTickets(TicketRelationType.MASTER_SLAVE)
```

Parent-Child-Relationen

Dieser Relationstyp kann hilfreich sein, wenn Sie eine Hierarchie zwischen einer bestimmten Anzahl von Tickets erstellen möchten, welche nicht manuell geändert werden soll.

Beispiel-Anwendungsfälle sind:

- Ein Projekt soll durch ein Projektmanagement-Ticket verwaltet werden, welches das Parent-Ticket wird. Alle Aufgaben im Projekt werden durch Child-Tickets repräsentiert. Diese Struktur wird während der Erstellung des Projekt-Tickets automatisch durch ein Workflow-Skript erzeugt.
- Eine Systemmigration wird mittels eines Parent-Tickets geplant. Für jede einzelne Komponente, die migriert werden soll, wird ein Child-Ticket erstellt. Diese Struktur wird während der Erstellung des Projekt-Tickets automatisch durch ein Workflow-Skript erzeugt.

Der Relationstyp *PARENT_CHILD* kann nur mittels der ConSol*CM API erstellt und verändert werden. Daher kann eine Relation dieses Typs nur mit einem Skript erstellt werden und danach nur durch andere Skripte verändert werden.

Beispiel 1: Erstellen eines neuen Child-Tickets als Child des Current Tickets

Erstellen eines Child-Tickets

```
// dieses Skript erstellt ein Ticket für eine Aufgabe, welches ein Child-Ticket
// eines Projekt-Tickets werden wird; das Projekt-Ticket (das aktuelle Ticket) wird das Parent-
Ticket

// erstelle ein neues Ticket, welches das Aufgaben(=Child)-Ticket werden wird
Ticket newTask = new Ticket()
// hole den Betreff des Tickets, welches zum Parent werden wird, d.h. des current ticket
def subj = ticket.subject
// or longer:  def subj = ticket.getSubject()
// setze den Betreff des neuen Aufgaben(=Child)-Ticket
newTask.setSubject("Neue Aufgaben für Projekt " + subj)
// setze das Aufgaben(=Child)-Ticket in die Aufgaben-Queue
def tasksQueue = queueService.getByName("Tasks")
newTask.setQueue(tasksQueue)
// Anfangs wird das neue Aufgaben-Ticket keinen Bearbeiter besitzen
newTask.setEngineer(null)
// definiere den Tickettext, d.h. den ersten Kommentar im neuen Aufgaben-Ticket
def taskTicketText = "Bitte an dieser Aufgabe asap arbeiten"
// der Hauptkontakt für das neue Aufgaben-Ticket soll der gleiche sein, wie der für das Projekt-
Ticket:
def taskContact = workflowApi.getPrimaryContact()
//erstelle PARENT_CHILD-Relation zwischen dem Projekt (Parent) und der Aufgabe (Child)
workflowApi.createChildTicket(newTask, taskTicketText, taskContact)
```

Beispiel 2: Finden des Parent-Tickets eines Tickets

Finden des Parent-Tickets eines Tickets

```
def my_parent = workflowAPI.getParentTicket()
```

Beispiel 3: Finden aller Child-Tickets eines Tickets

Finden aller Child-Tickets eines Tickets

```
// funktioniert nur beim current ticket:
List<Ticket> my_childtickets = workflowApi.getChildTickets()
```

Beispiel 4: Finde alle Brother-Tickets (andere Child-Tickets) des gleichen Parent-Tickets

Finde alle Brother-Tickets eines (Child-)Tickets

```
// funktioniert nur beim current ticket:
List<Ticket> my_brothers = workflowApi.getBrotherTickets()
```

Wichtige Methoden für die Arbeit mit Ticketrelationen

Bitte beachten Sie die folgenden Regeln für die Arbeit mit Ticketrelationen:

- In MASTER_SLAVE-Relationen ist der Master immer die Quelle.
- In PARENT_CHILD-Relationen ist das Parent immer die Quelle.
- In einfachen REFERENCE-Relationen ist die Quelle das Ticket, von welchem aus die Relation erstellt wurde.

Die folgenden Methoden sind Methoden der Klasse **WorkflowContextService**, welche implizit als **workflowApi**-Objekt in Workflow-Skripten verfügbar ist.

Methode	Erklärung
Ticket createChildTicket(Ticket pTicket, String pTicketText, Unit pCustomer)	Erstellt ein neues Child-Ticket. Queue, Priorität und Kategorie müssen korrekt gesetzt werden.
List getChildTickets()	IntSet, das die Ticketobjekte des Child-Tickets des current tickets enthält.
List getBrotherTickets()	IntSet, das die Ticketobjekte der Brother-Tickets des current tickets enthält.
Ticket getParentTicket()	Ticketobjekt des Parent-Tickets oder *null*, wenn das current ticket kein Parent-Ticket besitzt.
List getTargetTickets(TicketRelationType pType)	Hole die Liste von Ticketobjekten, zu denen das current ticket Relationen eines bestimmten Typs besitzt. Für diese Relationen ist das current ticket das Quell-Ticket.
List getTargetTickets(long pTicketId, TicketRelationType pType)	Hole die Liste von Ticketobjekten, zu denen das current ticket Relationen eines bestimmten Typs besitzt. Für diese Relationen ist das Ticket das Quell-Ticket, das für *pTicketId* angegeben wird.
List getSourceTickets(TicketRelationType pType)	

Methode	Erklärung
	Hole die Liste von Ticketobjekten, von denen das current ticket Relationen eines bestimmten Typs besitzt. Für diese Relationen ist das current ticket das Ziel-Ticket.
List getSourceTickets(long pTicketId, TicketRelationType pType)	Hole die Liste von Ticketobjekten, von denen das current ticket Relationen eines bestimmten Typs besitzt. Für diese Relationen ist das Ticket das Ziel-Ticket, das für *pTicketId* angegeben wird.
boolean hasTargetTickets(TicketRelationType pType)	Überprüfe, ob das Ticket Ziel-Tickets besitzt. Überprüfe, ob Relationen bestehen, die dieses Ticket als Quell-Ticket besitzen.
boolean hasTargetTickets(long pTicketId, TicketRelationType pType)	Überprüfe, ob das angegebene Ticket Ziel-Tickets besitzt. Überprüfe, ob Relationen bestehen, die dieses Ticket als Quell-Ticket besitzen.
boolean hasSourceTickets(TicketRelationType pType)	Überprüfe, ob das Ticket Quell-Tickets besitzt. Überprüfe, ob Relationen bestehen, die dieses Ticket als Ziel-Ticket besitzen.
boolean hasSourceTickets(long pTicketId, TicketRelationType pType)	Überprüfe, ob das angegebene Ticket Quell-Tickets besitzt. Überprüfe, ob Relationen bestehen, die dieses Ticket als Ziel-Ticket besitzen.
void changeSourceTickets(TicketRelationType pType, long pTargetTicketId, List<Long> pSourceTicketIds)	Für das Ziel-Ticket (z.B. ein Child-Ticket) werden die Relationen eines bestimmten Typs (z.B. PARENT_CHILD) entfernt. Mit dem gleichen Relationstyp wird eine neue Relation mit den angegebenen Quell-Tickets erzeugt.
void changeTargetTickets(TicketRelationType pType, long pSourceTicketId, List<Long> pTargetTicketsIds)	Für das angegebene Quell-Ticket werden alle Relationen des angegebenen Typs entfernt. Für die Liste der gelieferten Ziel-Tickets werden neue Relationen des angegebenen Typs erzeugt.
void removeRelation(TicketRelationType pType, long pSourceTicketId, long pTargetTicketId)	Entferne die Ticketrelation zwischen zwei Tickets mit dem spezifizierten Typ.
void addRelation(TicketRelationType pType, String pComment, long pSourceTicketId, long pTargetTicketId)	Füge eine Relation des spezifizierten Typs zwischen dem Ticket *sourceTicketId* und *targetTicketId* hinzu.

6.7.3 Arbeiten mit Kundenrelationen (Datenobjektrelationen)

- Einleitung
- Erstellen von Unit-Relationen mittels der CM API
    - Beispiel: Hinzufügen einer Reseller-Endkunde-Relation
- Wichtige Java-Klassen für die Arbeit mit Unit-Relationen

Einleitung

Seit Version 6.9.0 bietet ConSol*CM *Kundenrelationen.* In älteren Versionen ist dieses Feature nicht verfügbar!

Um mit Kundenrelationen arbeiten zu können, müssen Sie tiefgehende Kenntnisse über *FlexCDM*, dem Flexiblen Kundendatenmodell von ConSol*CM, besitzen. Bitte lesen Sie das *ConSol*CM Administratorhandbuch (Version 6.9)* für detaillierte Informationen.

Drei Groovy-Objekte sind essentiell:

Objekt	Groovy-Klasse	Admin-Tool-Bezeichnung	Erklärung
Kunde	Unit	<keine>	Die allgemeine Bezeichnung oder das allgemeine Objekt, dass einen Kunden repräsentiert, d.h. eine Person oder eine Firma, die in der CM-Datenbank liegt
Firma	Unit	Datenobjekt vom Typ *Firma.*	Ein Objekt auf der Stufe *Firma* (d.h. auf der höchsten Stufe im Kundendatenmodell). Dies kann eine reale Firma sein, oder auch eine Maschine oder ein anderes Objekt, welches diese Stufe repräsentiert. Ein Objekt auf der *Firmen-Stufe* kann die Parent-Stufe für eine Objekt auf der Stufe *Kontakt* sein. Aus der logischen

Objekt	Groovy-Klasse	Admin-Tool-Bezeichnung	Erklärung
			Perspektive kann eine Firma mehrere Kontakte besitzen.
Kontakt	Unit	Datenobjekt vom Typ *Kontakt.*	Ein Objekt auf der Stufe *Kontakt* (d.h. auf der niedrigsten Stufe im Kundendatenmodell). Dies kann eine reale Person sein oder ein anderes Objekt, welches diese Stufe repräsentiert. Ein Objekt auf der *Kontakt-Stufe* kann ein für sich allein stehendes Objekt sein (in einem einstufigen Kundendatenmodell) oder einem *Firmen-Stufen*-Objekt untergeordnet sein. Aus der logischen Perspektive kann ein Kontakt zu keiner oder genau einer Firma gehören.

Vorsicht:

Bitte beachten Sie, dass, beginnend mit CM-Version 6.9, der Hauptkunde eines Tickets ein Kontakt oder eine Firma sein kann! Die verwendete Methode ist *ticket.getMainContact()*. Diese gibt ein Objekt der Klasse *Unit* zurück. Dieses Objekt kann ein Kontakt oder eine Firma sein!

Kundenrelationen repräsentieren Relationen zwischen Kunden, d.h. Firmen und Kontakten.

Diese können sein:

- **Gerichtet**
  Unterschiedliche Hierarchiestufen.
- **Referenz**
  Gleiche Stufe, keine Hierarchie.

Eine Relation gehört einem der folgenden Typen an:

- **Firma - Firma**
  z.B. ....*steht in Kooperation mit* ... (Firma X kooperiert mit Firma Y)
  - Die Firmen können derselben oder unterschiedlichen Kundengruppen angehören.
  - Die involvierten Kundengruppen können dasselbe oder unterschiedliche Kundendatenmodelle besitzen.
- **Firma - Kontakt**
  z.B. ... *ist Kunde von* ... (Kontakt X ist Kunde der Firma Y)
  - Die Firma und der Kontakt können derselben oder unterschiedlichen Kundengruppen angehören.
  - Die involvierten Kundengruppen können dasselbe oder unterschiedliche Kundendatenmodelle besitzen.
- **Kontakt - Kontakt**
  z.B. ... *wird betreut von* ... (Kontakt X von Firma X wird von Kontakt Y von Firma Y betreut)
  - Die Firmen und die Kontakte können derselben oder unterschiedlichen Kundengruppen angehören.
  - Die involvierten Kundengruppen können dasselbe oder unterschiedliche Kundendatenmodelle besitzen.

In der ConSol*CM API wird ein Kundenobjekt (d.h. ein Kontakt oder eine Firma) durch ein Objekt der Klasse *Unit* repräsentiert.

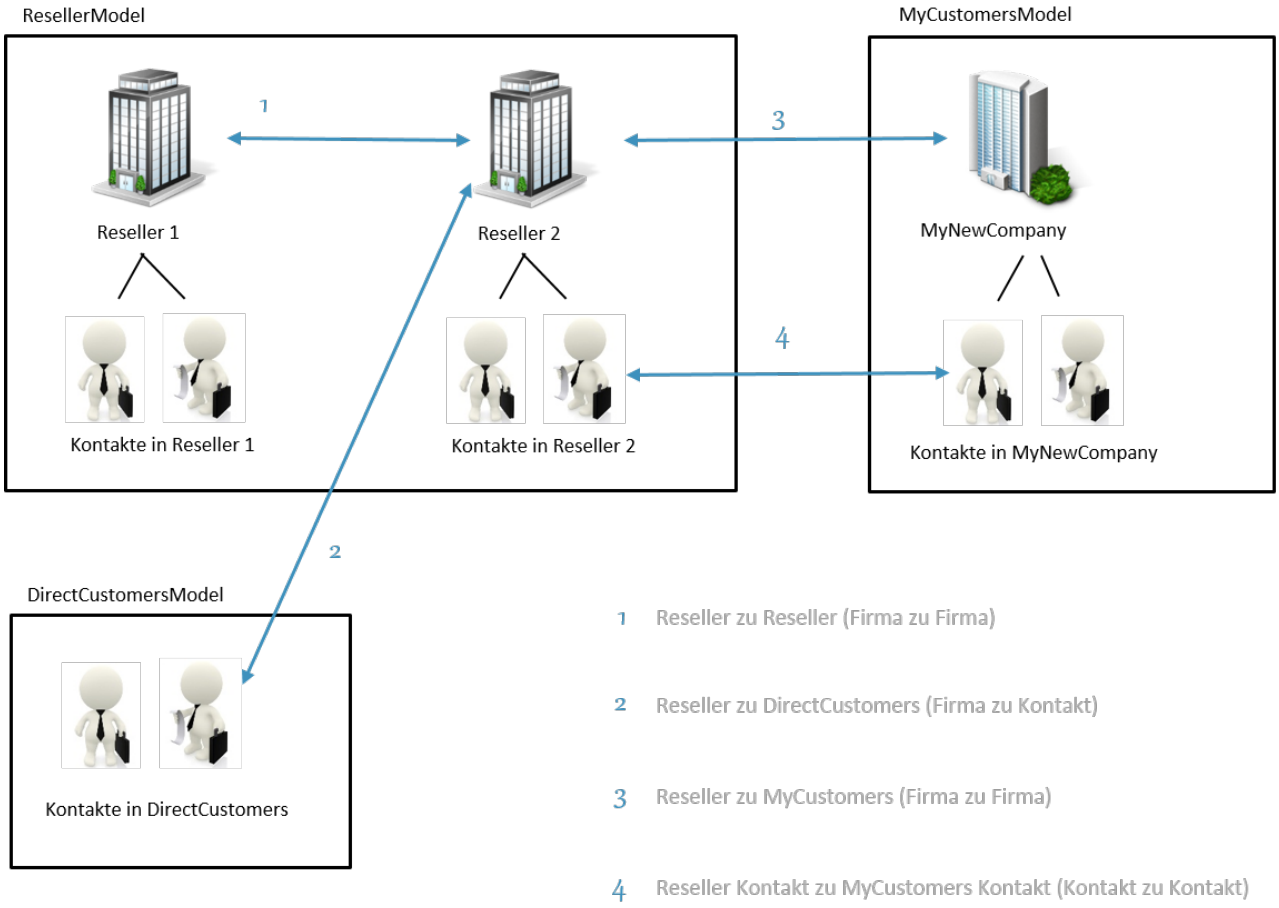


Fig. 1: ConSol*CM Kundenrelationen

Vorsicht:

Wenn Sie mit *Unit*-Relationen in Workflow-Skripten zu arbeiten, stellen Sie vorher sicher, dass Sie alle benötigten Relationen im Admin-Tool erstellt und konfiguriert haben, bevor Sie mit der Programmierung anfangen.

Erstellen von Unit-Relationen mittels der CM API

Vorsicht:

In diesem Handbuch verwenden wir manchmal die neuen Begriffe *Datenobjekt* und *Datenobjektdefinition*, welche Teil des neuen Kundenmodells von ConSol*CM-Version 6.9 und höher sind (*FlexCDM*). Trotzdem lauten die Namen der entsprechenden Java-Klassen *Unit* und *UnitDefinition*. Alle anderen Java-Klassen, welche Kundendatenobjekte betreffen werden ebenfalls weiterhin *Unit...* genannt. Bitte beachten Sie dies, wenn Sie sowohl auf Administrator-Ebene als auch auf Programmierer-Ebene mit einer 6.9.x Version arbeiten. Bitte nehmen Sie die *ConSol*CM Java API* Dokumentation zu Hilfe, um Details nachzuschlagen.

Beispiel: Hinzufügen einer Reseller-Endkunde-Relation

Im folgenden Beispiel wurde im Admin-Tool eine Relation definiert,um eine *Reseller-Endkunde-Relation* abzubilden. Eine Firma der Kundengruppe *Reseller* verkauft Produkte an einen Kunden (eine Person, einen Kontakt) der Kundengruppe *DirectCustomers*.

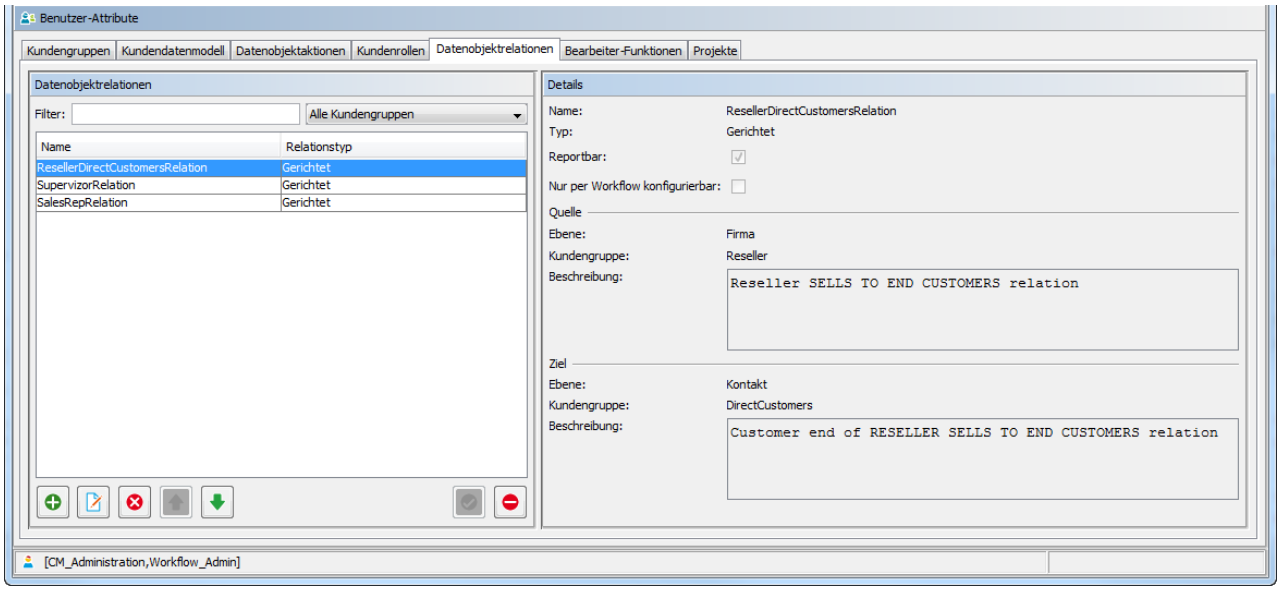


Fig. 2: ConSol*CM Admin-Tool - Definition einer Reseller-Endkunde-Relation

Ein Ticket mit einem Hauptkunden wird erstellt. Dieser Kunde ist ein Mitarbeiter der Reseller-Firma. Der Endkunde, an den die Reseller-Firma Produkte verkauft, wird als zusätzlicher Kunde mit der Rolle *Endkunde* zum Ticket hinzugefügt. Der Bearbeiter, der mit dem Ticket arbeitet, soll in der Lage sein, eine Relation zwischen der Reseller-Firma (Quelle) und der Endkunden-Person (Ziel) mittels einer Workflow-Aktivität zu erstellen.

Fig. 3: ConSol*CM Web Client - Beispielticket mit dem Hauptkunden und einem zusätzlichen Kunden

Im *Service Desk*-Workflow existiert eine Workflow-Aktivität *Reseller-Endkunde-Relation hinzufügen* (siehe nächstes Bild).

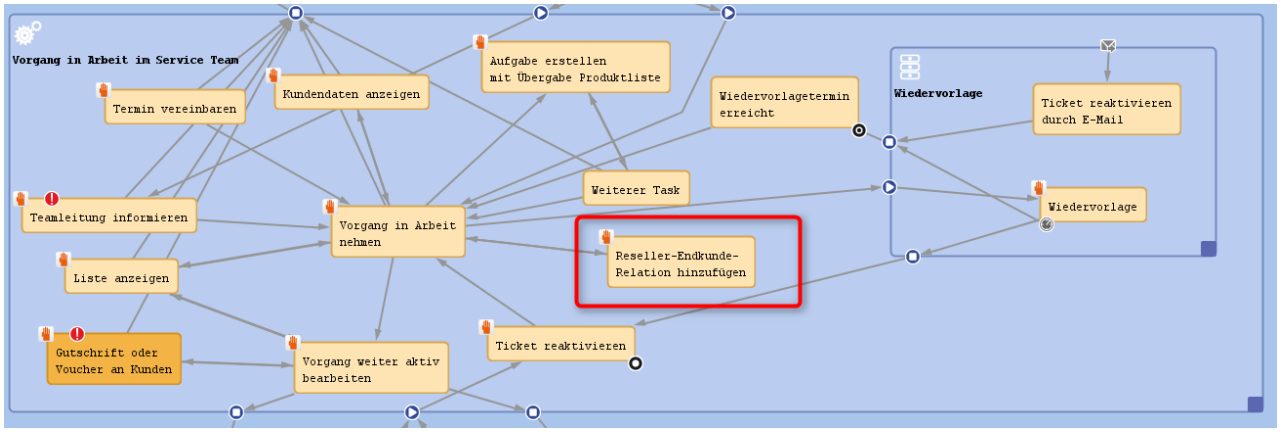


Fig. 4: ConSol*CM Process Designer - Workflow-Aktivität für das Hinzufügen einer Unit-Relation

Das folgende Skript wird in der Workflow-Aktivität *Reseller-Endkunde-Relation hinzufügen* verwendet:

Hinzufügen einer Datenobjektrelation mittels eines Workflow-Skripts

```
// hole die Firma des Hauptkunden des Tickets, dies ist die RESELLER-Firma:
// 1. hole den main contact des Tickets. Hier ist dies eine Person = Kontakt:
def cont = ticket.getMainContact()
// 2. hole die Firma dieses Kontakts, dies ist die Reseller-Firma
def comp = cont.getCompany()
// hole alle zusaetzlichen Kontakte des Tickets mit der Kundenrolle „Endkunde“ (technischer
Name "end customer")
// und starte den Loop für alle diese zusaetzlichen Kunden:
def end_custs = ticket.getContacts("end customer").each() { e_cust  ->
    // erstelle alle Komponenten für eine neue unit-Relation:
    // 1.hole die UnitDefinition nach Name (dies ist der Name, der im Admin-Tool verwendet
wird):
    def unitrel_def = unitRelationDefinitionService.getByName("ResellerDirectCustomersRelation")
    // erstelle ein neues unitRelation-Objekt mit der unit-Definition, -Quelle (die Reseller-
Firma)
    // und -Ziel (die Endkunden-Person)
    def new_rel = new UnitRelation(unitrel_def, comp, e_cust, "Dieser Reseller verkauft an
diesen Endkunden")
    // erstelle die neue unit-Relation im System
    def new_rel2 = unitRelationService.create(new_rel)
}
```

Wenn der Bearbeiter die Workflow-Aktivität ausgeführt hat, wird die Relation von der *Reseller*-Firma zum *Endkunden* erstellt.

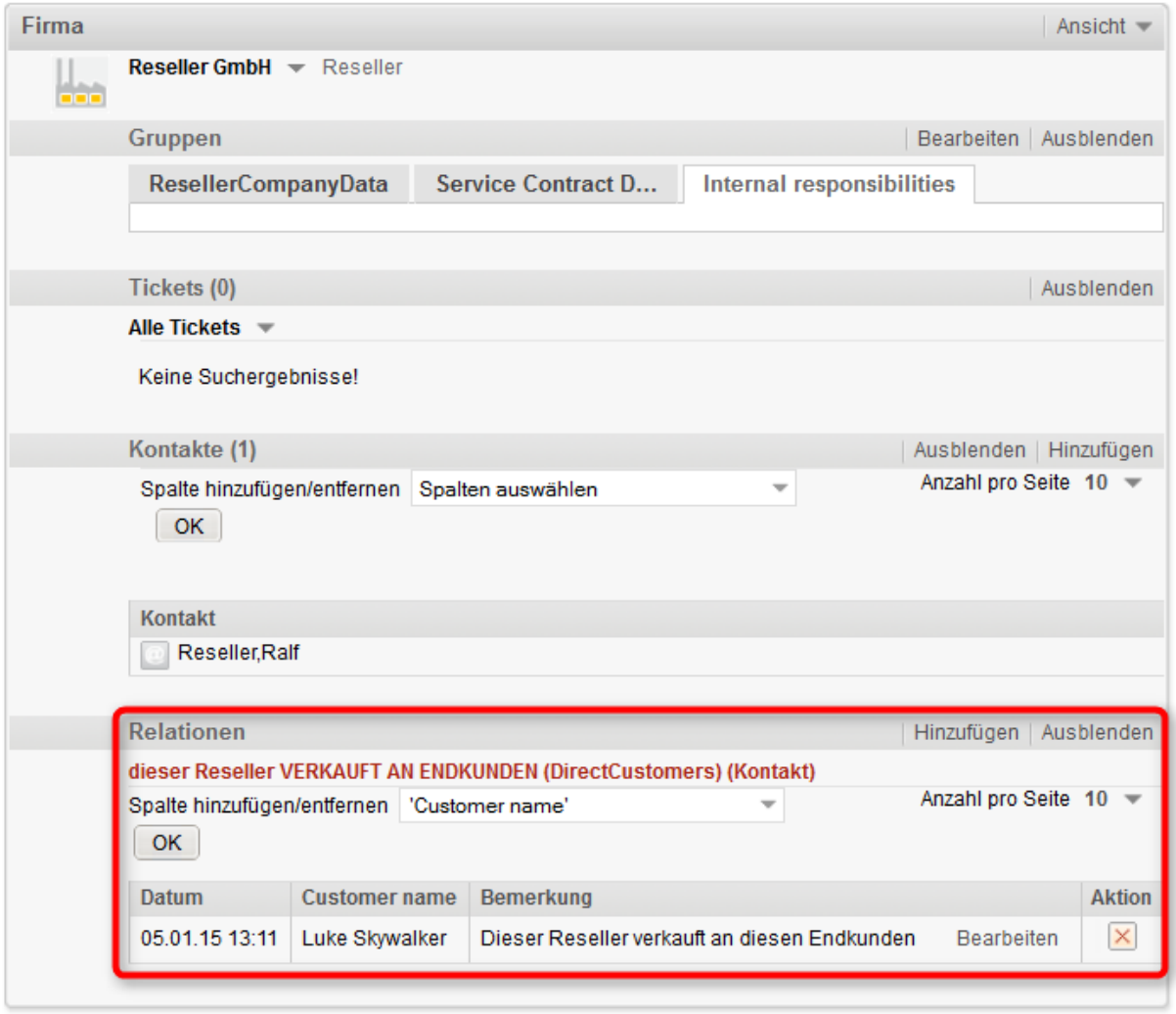


Fig. 5: ConSol*CM/Web Client - Neue Unit-/Kundenrelation (erstellt durch Workflow-Skript)

Wichtige Java-Klassen für die Arbeit mit Unit-Relationen

Java-Klassen	Erklärung
Unit	Ein Datenobjekt (Unit): ein Kontakt oder eine Firma.
UnitRelation	Eine Relation zwischen zwei Datenobjekten (Units). Sichtbar im Web Client, auf der Kundeseite des Kontakts oder der Firma unter *Relationen*.
UnitRelationDefinition	Die Definition einer Unit-Relation, wie sie im Admin-Tool unter *Benutzerattribute* -

Java-Klassen	Erklärung
	Datenobjektrelationen konfiguriert wurde. Eine *Unit Relation* besitzt immer eine bestimmte *UnitRelation Definition*.
UnitRelationDefinitionService	Singleton. Verfügbar als Objekt *unitRelationDefinitionService*. Ein Service, der nützliche Methoden für die Arbeit mit Datenobjekt- (Unit-) Relationen bereitstellt. Siehe *ConSol*CM Java API* Dokumentation für Details..
UnitRelationService	Singleton. Verfügbar als Objekt *unitRelationService* . Ein Service, der nützliche Methoden für die Arbeit mit Datenobjekt- (Unit-) Relationen bereitstellt. Siehe *ConSol*CM Java API* Dokumentation für Details..

6.8 Suche nach Tickets und Kunden mittels der ConSol*CM Workflow API

- Einleitung
- Suche nach Tickets
  - Beispiel 1: Allgemeines Beispiel für die Suche nach Tickets
  - Beispiel 2: Finde alle Tickets mit dem gleichen Service wie das Current Ticket
  - Beispiel 3: Suche nach Tickets gemäß Unit
- Suche nach Units (Kontakten und Firmen)
  - Beispiel 1: Suche nach Kontakten mittels Vorname und Nachname
  - Allgemeine Syntax für die Suche nach Units mittels Enum-Werten
  - Beispiel 2: Suche nach Units mittels Enum-Wert

6.8.1 Einleitung

In ConSol*CM können Sie die Datenbank nach Tickets und Units (Kontakten und Firmen) durchsuchen. Beide Suchmodi basieren auf dem gleichen Prinzip:

1. Ein *criteria*-Objekt wird erstellt, in dem alle Parameter für das Ziel-Objekt gespeichert werden.
   a. **TicketCriteria** für Tickets
   b. **UnitCriteria** für Kontakte und Firmen
2. Dieses *criteria*-Objekt wird an einen Service übergeben, welcher eine Liste mit den Ergebnis-Objekten zurückgibt.
   a. **workflowApi** (WorkflowContextService) für Tickets
   b. **UnitService** für Units

Die Felder, welche als Parameter für die *criteria*-Objekte gesetzt werden, müssen indiziert werden, d.h. die Annotation *field-indexed* muss gesetzt werden.

6.8.2 Suche nach Tickets

Um nach Tickets zu suchen, müssen Sie ein *TicketCriteria*-Objekt erstellen. Die folgenden Felder können gesetzt werden (siehe auch die entsprechenden *Setter*-Methoden im folgenden Bild*)*:

- Datum der Ticketeröffnung
- Bearbeiter
- System-spezifische Benutzerdefinierte Felder
- Ticketprotokoll-Kriterien
- Ticket-IDs
- Änderungsdatum
- Ticket-Name
- Muster für das Ticket-Thema
- Queue-IDs

- IDs der aktuelle Workflow-Scopes
- Aktueller Status (offen/geschlossen)
- Zusätzliche Bearbeiter

```
setCreationDateRange(DateRange pCreationDateRange)
setEngineerCriteria(TicketCriteria.EngineerCriteria pEngineerCriteria)
setFields(Set<AbstractField> pFields)
setHistoryCriteria(TicketCriteria.HistoryCriteria pHistoryCriteria)
setIdRange(org.apache.commons.lang.math.LongRange pIdRange)
setIds(Set<Long> pIds)
setModificationDateRange(DateRange modificationDateRange)
setName(String pName)
setPattern(String pPattern)
setQueueIds(Set<Long> pQueueIds)
setScopeIds(Set<Long> pScopeIds)
setStatus(TicketCriteria.Status pStatus)
setSubject(String pSubject)
setUserCriteria(Set<TicketUserCriteria> pUserCriteria)
```

Fig. 1: Setter-Methoden der Klasse TicketCriteria, CM-Version 6.9.3

Das *TicketCriteria*-Objekt muss an den *WorkflowContextService* übergeben werden, welcher in jedem Skript implizit als Singleton *workflowAPI* verfügbar ist. Bitte sehen Sie sich die folgenden Beispiele an und lesen Sie für Details über Klassen und Methoden die *ConSol*CM Workflow API Java* Dokumentation.

Beispiel 1: Allgemeines Beispiel für die Suche nach Tickets

Suche nach Tickets

```
def ticketCrit = new TicketCriteria()
ticketCrit.subject = "TICKET_SUBJECT"
ticketCrit.setQueueIds([new Long(workflowApi.getQueueByName("QUEUE_NAME").id)] as Set)
ticketCrit.setFields([new StringField(new FieldKey("FIELD_GROUP", "FIELD_NAME"), "SEARCH_VALUE")
] as Set)
def foundTickets = workflowApi.getTicketsByCriteria(ticketCrit)
def firstTicket = foundTickets?.first()
```

Beispiel 2: Finde alle Tickets mit dem gleichen Service wie das Current Ticket

Das folgende Beispiel stammt aus einem Workflow für eine Helpdesk-Umgebung. Wenn ein Ticket erstellt wurden und der Service aus einer Liste ausgewählt wurde, soll der Workflow automatisch überprüfen, ob andere offene Tickets mit dem gleichen Service existieren. Für die Services wird eine Abhängige Sortierte Liste (*dependent enum*) verwendet:

- **1st level**
  Unterschiedliche Kategorien, eine davon ist *HARDWARE*.

- **2nd level**
  Existiert nur, wenn *HARDWARE* im 1st Level ausgewählt wurde. Im 2nd Level werden Hardware-Kategorien aufgelistet.

Finde alle Tickets mit dem gleichen Service wie das current ticket

```
def crit = new TicketCriteria()
crit.setStatus(TicketCriteria.Status.OPEN)
Set<AbstractField> cfs = new HashSet<AbstractField>();
if (serv1.getName().equals("HARDWARE")){
    def serv2 = ticket.get("Service_Fields.Hardware")
    cfs.add(new EnumField(new FieldKey("Service_Fields", "Hardware"), serv2));
} else {
    cfs.add(new EnumField(new FieldKey("Service_Fields", "Service"), serv1));
}
crit.setFields(cfs)
List<Ticket> foundTickets = workflowApi.getTicketsByCriteria(crit);
```

Beispiel 3: Suche nach Tickets gemäß Unit

In diesem Beispiel suchen wir nach dem *Account Management*-Ticket für eine bestimmte Firma.

Search for tickets by unit

```
import com.consol.cmas.common.model.scripting.unit.PostActionType
import com.consol.cmas.common.model.scripting.unit.PostActionParameter
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.ticket.TicketCriteria
import com.consol.cmas.common.model.customfield.ListField
import com.consol.cmas.common.model.customfield.ContactReferenceField
import com.consol.cmas.common.model.customfield.UnitReferenceSearchField
import com.consol.cmas.common.model.customfield.ContactReferenceSearchField
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.ContactTicketRole
import com.consol.cmas.common.model.customfield.StringField
import com.consol.cmas.common.model.scripting.unit.UnitActionScriptResult
//hole AM-Queue für die Suche
def q_id = (workflowApi.getQueueByName("AccountManagement")).id
def q_ids = new HashSet()
q_ids.add(q_id)
//finde AM-Ticket für diese Firma
def crit = new TicketCriteria()
crit.setQueueIds(q_ids)
// Erstelle ListFieldKey
def contactSearchListFieldKey  = new FieldKey("queue_fields","contacts")
// Bereite ListField vor
def contactsListField = new ListField(contactSearchListFieldKey )
// Erstelle Memberfield Key
def contactSearchFieldKey = new FieldKey("queue_fields","contacts_member")
// Erstelle Unit Memberfield mit Unit und Ticket-Main Role
def contactsMember = new ContactReferenceSearchField(contactSearchFieldKey, unit,
ContactTicketRole.MAIN_ROLE)
// Setze Memberfield in Unit List Field
contactsListField.addChild(contactsMember)
// Setze vorbereitete fields in TicketCriteria
crit.setFields([contactsListField] as Set)
// Suche ... und Resultat
def foundTickets = ticketService.getByCriteria(crit)
println "Found tickets: ${foundTickets}"
if ( foundTickets ) {
  def AM_tic = foundTickets.first()
  def AM_tic_id = AM_tic.id
}
```

6.8.3 Suche nach Units (Kontakten und Firmen)

Um nach Units (d.h. nach Kontakten und/oder Firmen) zu suchen, müssen Sie ein *UnitCriteria*-Objekt erstellen. Die folgenden Felder können gesetzt werden (siehe auch die entsprechenden *setter*-Methoden im folgenden Bild):

- Kundengruppe
- System-spezifische Datenobjektgruppenfelder
- Unit-IDs
- Muster für Units
- Telefonnummer (neu in CM-Version 6.9.3, verwendet für CM/Phone)
- TicketCriteria
- UnitDefinition-Name
- Boolean UseInCriterion

Danach verwenden Sie den *unitService*, um das Suchergebnis zu erhalten.

```
setCustomerGroupIds(Set<Long> pCustomerGroupIds)
setFields(Set<AbstractField> pFields)
setGroupNames(Set<String> pGroupNames)
Deprecated.
setIdRange(org.apache.commons.lang.math.LongRange pIdRange)
setIds(Set<Long> pIds)
setPattern(String pPattern)
setPhoneNumber(String pPhoneNumber)
setTicketCriteria(Set<AbstractField> pCallUnitReferences, TicketCriteria pTicketCriteria)
setUnitDefinitionNames(Set<String> pUnitDefinitionNames)
setUseInCriterion(boolean pUseInCriterion)
```

Fig. 2: Setter-Methoden der Klasse UnitCriteria, CM-Version 6.9.3

Beispiel 1: Suche nach Kontakten mittels Vorname und Nachname

Suche nach Kontakten mittels Vorname und Nachname

```
def unitCrit = new UnitCriteria()
unitCrit.setFields([new StringField(new FieldKey("UNIT_GROUP_NAME", "firstname"), "Max"),
                    new StringField(new FieldKey("UNIT_GROUP_NAME", "lastname"), "Mustermann")]
as Set)
def foundContacts = unitService.getByCriteria(unitCrit)
def firstContact = foundContacts?.first()
```

Allgemeine Syntax für die Suche nach Units mittels Enum-Werten

Suche nach Units mittels Enum-Werten (allgemeine Syntax)

```
import com.consol.cmas.common.model.customfield.UnitCriteria
import com.consol.cmas.common.model.customfield.EnumSearchField
import com.consol.cmas.common.model.customfield.meta.FieldKey
def unitCrit = new UnitCriteria()
def companyEnumField = new EnumSearchField(new FieldKey("customer", "company"), [enumService.
getValueByName("ENUM_GROUP_NAME",ENUM_VALUE_NAME)] as Set)
unitCrit.setFields([companyEnumField] as Set)
unitService.getByCriteria(unitCrit).each { foundContact ->
  println "Processing found contact: "+foundContact.get("name")
}
```

Beispiel 2: Suche nach Units mittels Enum-Wert

Suche nach Units mittels Enum-Wert (Beispiel)

```
def unitCrit = new UnitCriteria()
//alle anderem UnitCriteria init operations übersprungen
// dies ist der gewünschte Wert innerhalb der Liste:
def secLvl = ticket.get("transportEntryData.securityLevel")
//ShipperData/securityLevel ist der Pfad des EnumField innerhalb der Liste
def secLvlEnumFieldKey = new FieldKey("ShipperData","securityLevel")
//erstelle ein Template-Feld mit FieldKey und unserem Wert, um danach zu suchen
def secLvlTemplateField = new EnumField(secLvlEnumFieldKey, secLvl)
//ShipperData/securityLevels ist der Pfad der Liste selbst
def secLvlListTemplateFieldKey = new FieldKey("ShipperData","securityLevels")
//initialisiere die Template-Liste mit dem Wert, nach dem gesucht werden soll
def secLvlListTemplateField = new ListField(secLvlListTemplateFieldKey,[secLvlTemplateField])
// setze die Template-Liste in das UnitCriteria-Objekt
unitCrit.setFields([secLvlListTemplateField] as Set)
// Suche ... und Resultat
def shippers = unitService.getByCriteria(unitCrit)
```

6.9 Debug-Informationen

6.9.1 Einleitung

Manchmal ist es notwendig, den Output eines Workflow- oder Admin-Tool-Skripts mittels des Debug-Outputs in Logdateien zu überprüfen. In ConSol*CM wird der Debug-Output normalerweise in die Datei *server.log* geschrieben, welche sich im folgenden Pfad befindet:

- **In JBoss:**
  `<SERVER_HOME>\log\server.log`
- **In Oracle WebLogic:**
  `<DOMAIN_HOME>\cm-logs and <DOMAIN_HOME>\cmrf-logs\server.log`

Die Logging-Konfiguration kann geändert werden, indem Sie die *log4j*-Konfigurationsdatei editieren. Falls Sie einen anderen Log-Pfad als den Standard-Pfad definiert haben, finden Sie die *server.log*-Datei unter diesem Pfad.

Als Alternative zum Log-Output können Sie (z.B. in Test-Systemen) auch Informationen als Text in das Ticket schreiben.

6.9.2 Befehle für den Debug-Output

Debug-Output zur server.log-Datei

Die folgenden Befehle können verwendet werden, um Log-Informationen in die *server.log*-Datei zu schreiben. Dies funktioniert sowohl in Workflow-Skripten als auch in Admin-Tool-Skripten.

- `println 'Dies ist meine Debug-Meldung.'`
- `println("Dies ist meine Debug-Meldung.")`
- `log.info("Dies ist meine Debug-Meldung.")`
- `log.info "Dies ist meine Debug-Meldung."`

Vorsicht:

In einem WebLogic-System muss normalerweise der *log.info-Befehl* verwendet werden. Der Befehl *println* funktioniert wahrscheinlich nicht.

Debug-Output als Texteintrag in einem Ticket

Wenn Sie sich die Informationen in einem Ticket anzeigen lassen möchten (z.B. weil Sie keinen Zugriff auf das Dateisystem haben, in dem die Log-Dateien gespeichert sind), können Sie den Text als normalen Kommentar in ein Ticket schreiben:

Ticket Text hinzufügen

```
workflowApi.addTicketText('Dies ist meine Debug-Meldung', 'Dies ist der Betreff meiner Debug-Meldung', false)
```

Debugging von ConSol*CM Standard-Skripten

In ConSol*CM Standard-Skripten, z.B. *createTicket.groovy*, finden Sie Befehle ähnlich wie dem folgenden:

Debug-Eintrag im ConSol*CM Standard-E-Mail-Skript

```
if (log.isDebugEnabled()) {
        log.debug("Extracted email from from-field is $email")
}
```

Um den Debug-Output zu aktivieren, d.h. CM die Debug-Informationen in die Log-Dateien schreiben zu lassen, müssen Sie das Log-Level des entsprechenden Moduls (hier: E-Mail) auf *DEBUG* setzen. Dies geschieht in der Datei *jboss-log4j.xml*.

In diesem Handbuch wird dieses Thema nicht näher ausgeführt. Wenn Sie mehr über CM-Logging erfahren möchten, lesen Sie bitte das *ConSol*CM Betriebshandbuch*.

7 Best Practices

- Die grundlegende Organisation eines Workflows: Verwendung von Bereichen (Scopes)
  - Variante A: Verwendung eines globalen Scopes
  - Variant B: Verwendung von drei oder mehr Haupt-Scopes
- Die Position des START-Knotens
- Speichern Sie einige Workflow-Skripte im Admin-Tool
  - Wann Admin-Tool-Workflow-Skripte verwendet werden
  - Wie Admin-Tool-Workflow-Skripte verwendet werden
- Bedenken Sie die Verwendung von Trigger-Kombinationen genau
- Stoßen Sie keine unnötigen Ticket-Update-Events an
- Wie Sie den Parameter "Automatische Aktualisierung deaktivieren" verwenden
- Vermeiden Sie selbstauslösende Business-Event-Trigger

7.1 Die grundlegende Organisation eines Workflows: Verwendung von Bereichen (Scopes)

Eine der ersten Punkte, die Sie bedenken müssen, wenn Sie anfangen, ein Konzept für einen Workflow zu erstellen, ist die Anzahl und Organisation der Bereiche (Scopes).

> **Information:**
>
> Natürlich können Sie den Workflow in späteren Schritten immer noch ändern, aber dies kann Auswirkungen auf existierende Tickets, Sichten und Reports haben. Dies ist insbesondere von Bedeutung, wenn der Workflow in einem Produktionssystem verwendet wird.

Bedenken Sie die folgenden Punkte, wenn Sie die grundlegende Struktur eines Workflows erstellen:

- Welcher Trigger soll für das Ticket in welchen Status des Prozesses aktiv sein?
  Soll zum Beispiel ein Zeit-Trigger, der neue Tickets überwacht, auch für Tickets aktiv sein, welche sich bereits in Bearbeitung befinden? Oder soll ein Mail-Trigger aktiv sein, wenn ein Ticket durch einen Bearbeiter beendet wurde?
- Welche Sichten werden benötigt?
  Sichten basieren auf der Position der Tickets in den Scopes, siehe *ConSol*CM Administratorhandbuch,* Abschnitt *Sichtenverwaltung* für Details.

7.1.1 Variante A: Verwendung eines globalen Scopes

Ein globaler Scope ist ein Scope, der alle anderen Scopes des Workflows umfasst. Es ist kann günstig sein, einen solchen globalen Scope zu verwenden, da einige Prozesse während des gesamten Prozesses Reaktionen auf Ereignisse erfordern. Solche Ereignisse werden mittels Trigger implementiert, die an dem globalen Scope hängen. Wenn Sie zum Beispiel für den gesamten Prozess überwachen möchten, ob eine E-Mail eingetroffen ist, verbinden Sie einen Mail-Trigger (siehe Abschnitt Mail-Trigger) mit dem globalen Scope. Alle Unter-Scopes des globalen Scopes übernehmen die Anfälligkeit für diesen Trigger. Wenn die E-Mail nur in einem Unter-Scope überwacht werden soll, verbinden Sie den Mail-Trigger mit diesem Unter-Scope.

Das gleiche Prinzip gilt für alle anderen Arten von Triggern, z.B. Business-Event-Trigger (siehe Abschnitt Business-Event-Trigger) und Zeit-Trigger (siehe Abschnitt Zeit-Trigger).

Der START-Knoten muss immer außerhalb des globalen Scopes positioniert werden!

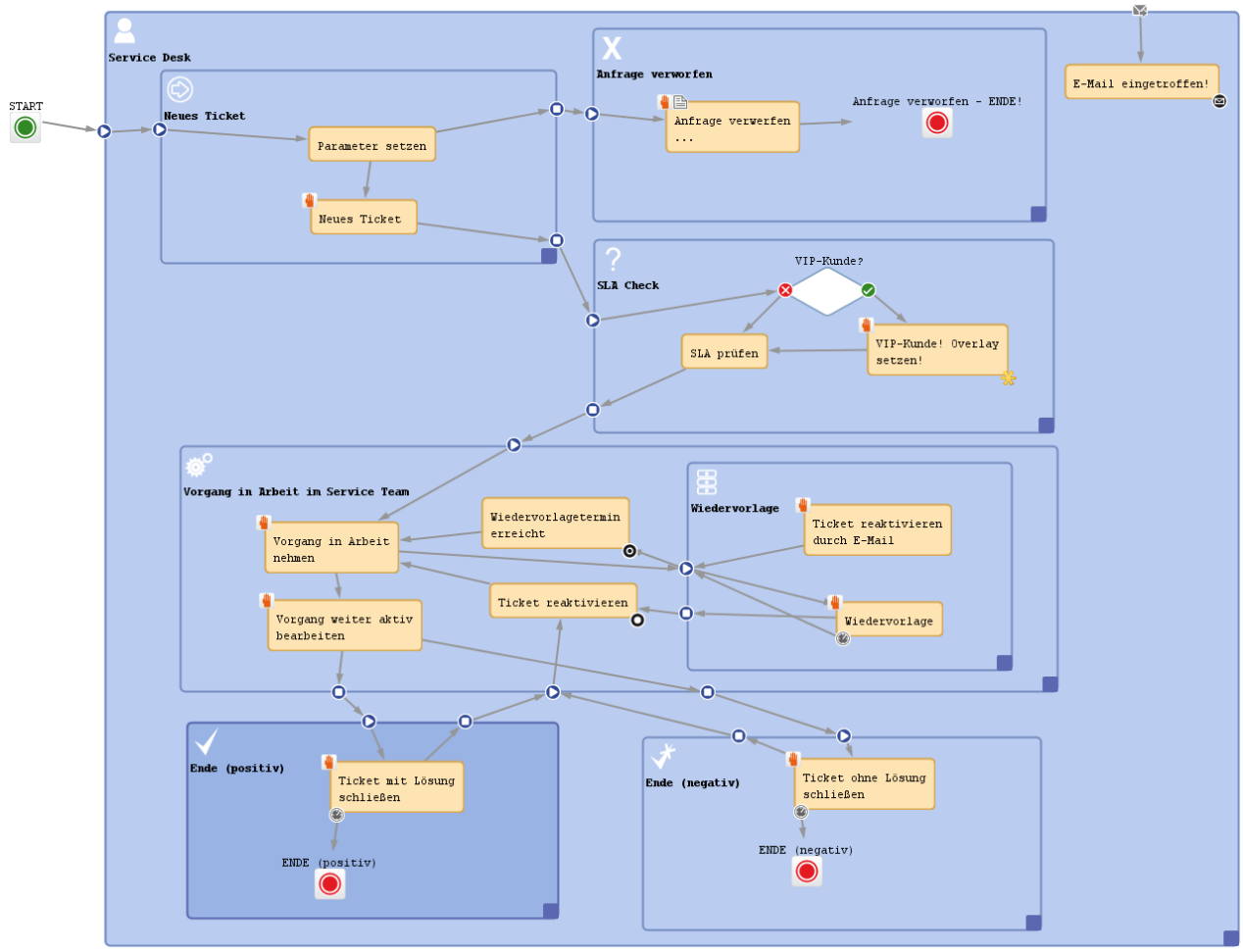


Fig. 1: ConSol*CM Process Designer - Workflow mit globalem Scope

Bitte beachten Sie, dass Sie immer Trigger in inneren Scopes verwenden können, die dann das Ereignis konsumieren (siehe Abschnitt Feuer-Reihenfolge von Business-Event-Triggern als Beispiel für Business-Event-Trigger). Wenn Sie zum Beispiel einen Mail-Trigger für den gesamten Prozess im globalen Scope verwenden möchten, aber eine bestimmte Reaktion des Tickets auf eingehende E-Mails im Scope *Beendet* benötigen, können Sie zusätzlich zum Trigger am globalen Scope einen Mail-Trigger, der mit dem Scope *Beendet* verbunden ist, verwenden.

7.1.2 Variant B: Verwendung von drei oder mehr Haupt-Scopes

Ein alternativer Weg, um einen Workflow zu konstruieren, ist die Verwendung von drei oder mehr Haupt-Scopes, wie zum Beispiel:

- Neue Tickets
- In Bearbeitung (nur hier wird ein Mail-Trigger angefügt)
- Geschlossene Tickets (in einem oder mehreren separaten Scopes)

Das folgende Bild zeigt ein Beispiel für einen Workflow, welcher nach diesem Prinzip erstellt wurde:

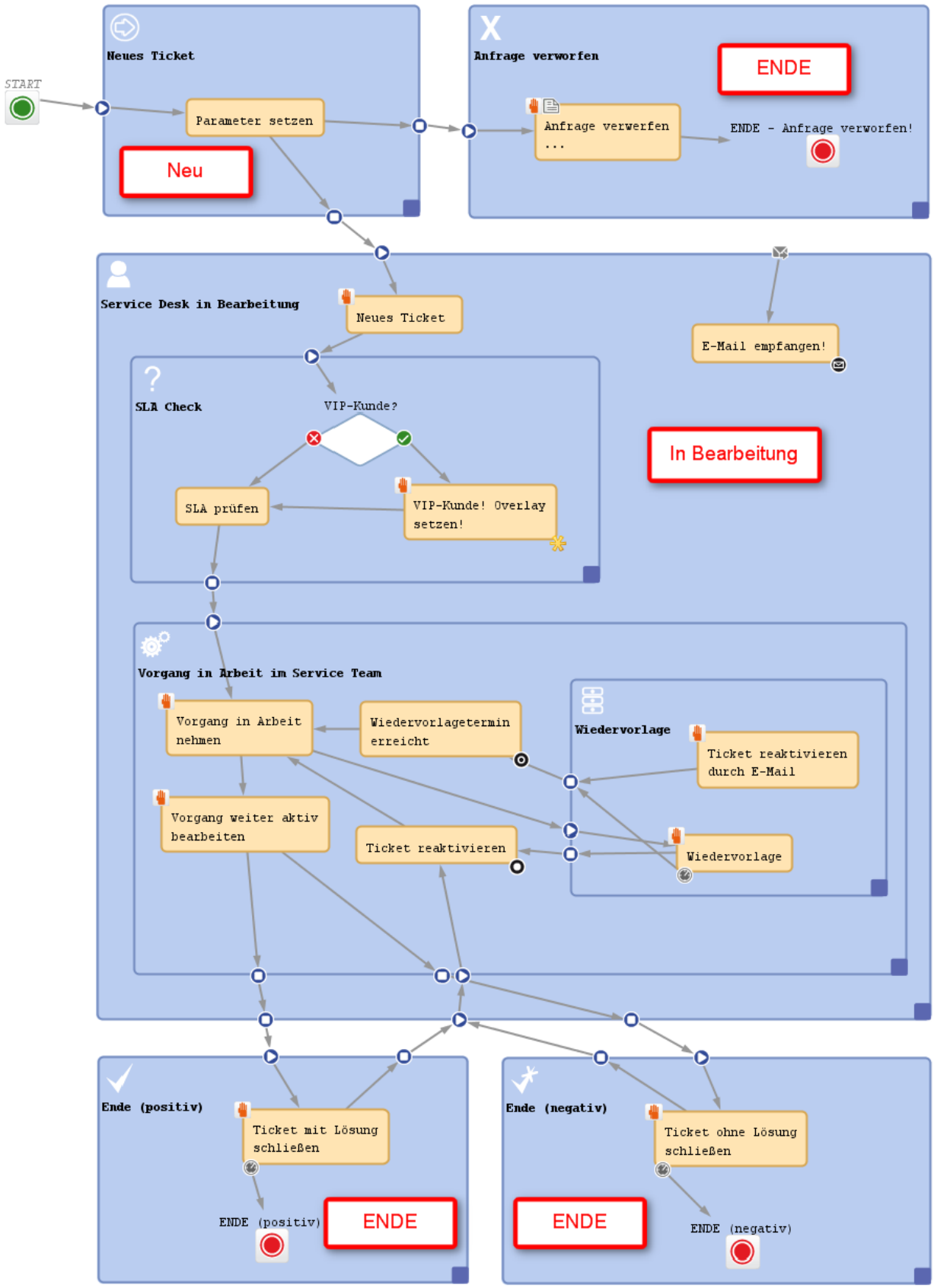


Fig. 2: ConSol*CM Process Designer - Workflow mit drei Arten von Haupt-Scopes

7.2 Die Position des START-Knotens

Die beste Position des START-Knotens hängt von der Verwendung von Triggern im folgenden Scope ab. Wenn im ersten Scope Zeit-Trigger verwendet werden, in den Tickets nach dem Start-Knoten weitergeleitet werden, sollte der Start-Knoten außerhalb des Scopes platziert werden. Falls der Start-Knoten innerhalb des ersten Scopes platziert wird, könnte der Zeit-Trigger möglicherweise nicht korrekt initialisiert werden. Platzieren Sie den Start-Knoten daher in den Standard(*Default*)-Scope (also ausserhalb aller weiteren Scopes).

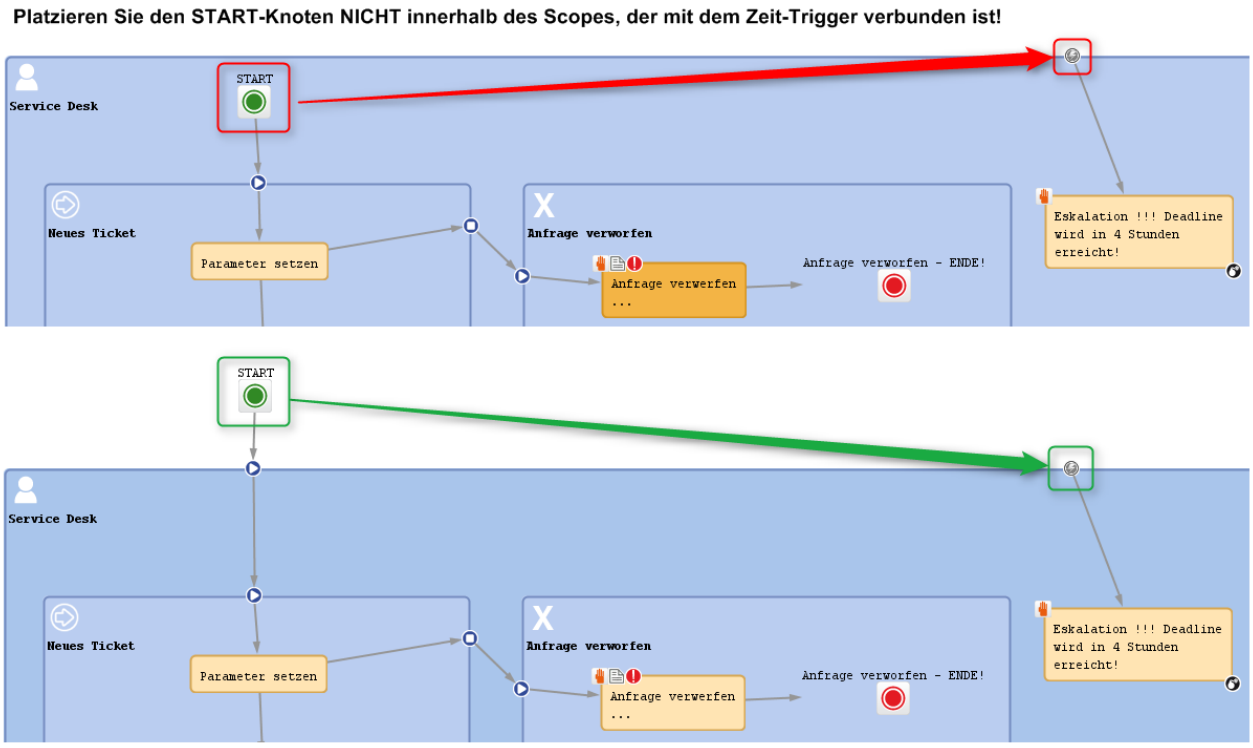


Fig. 3: ConSol*CM Process Designer - Position eines START-Knotens

7.3 Speichern Sie einige Workflow-Skripte im Admin-Tool

Es ist empfehlenswert, Skripte, die immer wieder in Workflow-Aktivitäten und/oder Bedingungsskripten verwendet werden, im Admin-Tool im Bereich *Skripte* zu speichern und vom Workflow-Skript aus aufzurufen.

7.3.1 Wann Admin-Tool-Workflow-Skripte verwendet werden

Wir empfehlen weder, diese Methode immer zu verwenden, noch raten wir von dieser Methode ab. Wir zeigen Ihnen die Vor- und Nachteile dieser Vorgehensweise auf, so dass Sie selbst entscheiden können, an welcher Stelle in Ihrem System Sie sie anwenden möchten.

Die **Vorteile** des Speicherns von Workflow-Skripten im Admin-Tool sind die folgenden:

- Das Skript wird nur einmal gespeichert und muss nur an einer Stelle gepflegt/geändert werden.
- Veränderungen am Skript werden im System sofort ausgeführt, es wird keine Installation (Deployment) benötigt (im Gegensatz dazu ist dies für Workflows notwendig).

Die **Nachteile** des Speicherns von Workflow-Skripten im Admin-Tool sind die folgenden:

- Die Prozesslogik wird an zwei getrennten Stellen gespeichert, d.h. Sie müssen immer sowohl mit dem Process Designer als auch mit dem Admin-Tool arbeiten, um den gesamten Prozess zu erkennen.
- Der Skript-Editor im Admin-Tool ist nicht so komfortabel wie der Workflow-Skript-Editor.
- Die meisten Objekte müssen in Admin-Tool-Skripte importiert werden, da sie dort nicht implizit verfügbar sind.
- Ein Workflow-Export allein reicht nicht aus, um einen Workflow umzuziehen, da Skripte im Admin-Tool nicht in den Export eingeschlossen werden.

7.3.2 Wie Admin-Tool-Workflow-Skripte verwendet werden

Admin-Tool-Skripte, die im Workflow verwendet werden, müssen vom Typ *Workflow* sein. Ein Admin-Tool-Skript wird immer vom Workflow mittels des Interface *ScriptProvider* angesprochen.

Ansprechen eines Admin-Tool-Skripts von einem Workflow aus

```
def scriptProvider = scriptProviderService.createDatabaseProvider("scriptName.groovy")
def r = scriptExecutionService.execute(scriptProvider)
```

Ansprechen eines Admin-Tool-Skripts von einem Workflow aus unter Verwendung von Parameter

```
// Erstelle den scriptProvider fuer das benoetigte Admin-Tool-Skript, hier "scriptName.groovy"
def scriptProvider = scriptProviderService.createDatabaseProvider("scriptName.groovy")
// Definiere eine HashMap mit den key-value pairs welche Sie an das Admin-Tool-Skript übergeben
möchten
def params = [ "templateName": "newCustomer" ]
// Fuehre das Skript aus. Die uebergebenen Parameter sind im Admin-Tool-Skript verfuegbar. In
diesem
// Beispiel, muss die Variable templateName nicht im Admin-Tool-Skript definiert werden
// sondern ist verfuegbar basierend auf der Definition der übergebenen HashMap.
// Die Variable r wird den Rueckgabewert des Skripts enthalten oder Null, wenn es keinen
Rueckgabewert gibt
def r = scriptExecutionService.execute(scriptProvider, params)
```

7.4 Bedenken Sie die Verwendung von Trigger-Kombinationen genau

> **Vorsicht:**
>
> Vermeiden Sie das unnötige Ausführen von Triggern! Es konsumiert Ressourcen und verlangsamt die Performance der Anwendung.

Beispiel 1:
Dieses Beispiel zeigt viele Business-Event-Trigger in einem großen *globalen Scope*.

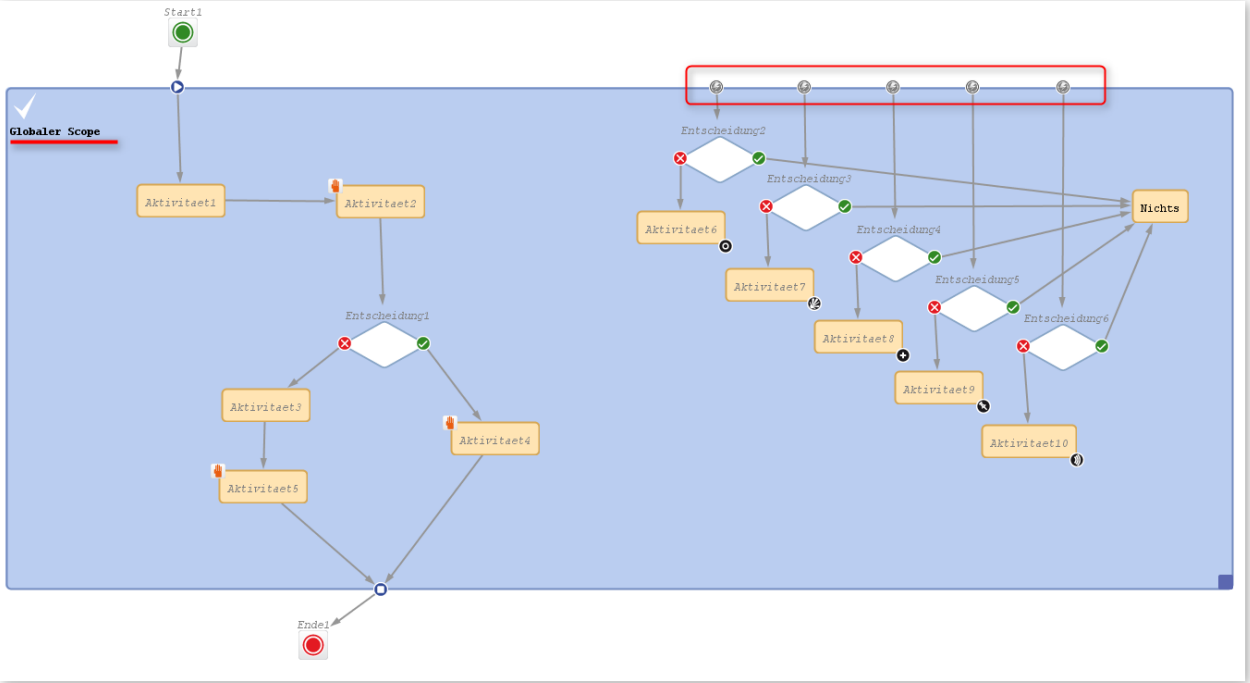


Fig. 4: ConSol*CM Process Designer - Scope mit Triggern

Beispiel 2:

Wenn möglich, verwenden Sie Trigger bitte in dem kleinstmöglichen Scope (in diesem Beispiel wurde der Trigger mit *Entscheidung6* in einen kleineren Scope verschoben).

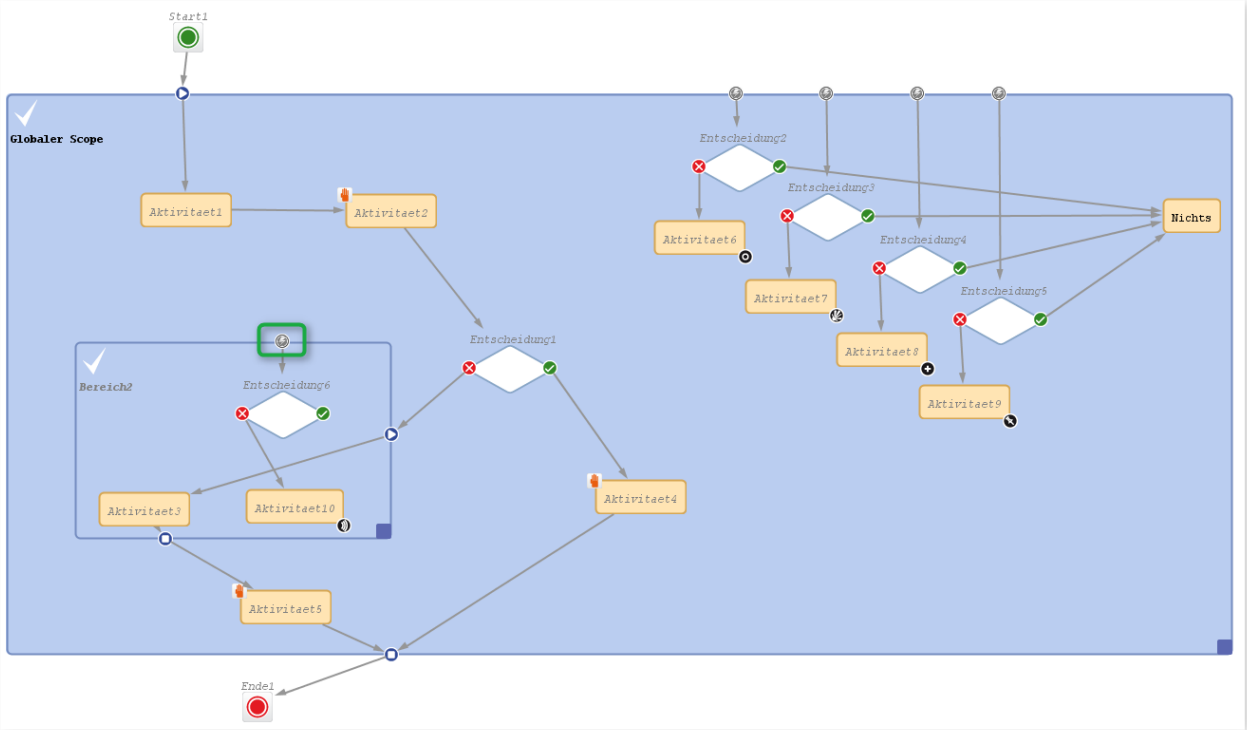


Fig. 5: ConSol*CM Process Designer - Trigger an kleineren Scope verschoben

Beispiel 3:
Wenn es **nicht** möglich ist, Trigger in einen kleineren Scope zu verschieben und Sie nicht möchten, dass alle Trigger während des Ausführens einer Aktivität angesprochen werden, verschieben Sie diese Aktivität in einen außerhalb liegenden Scope ohne Trigger.

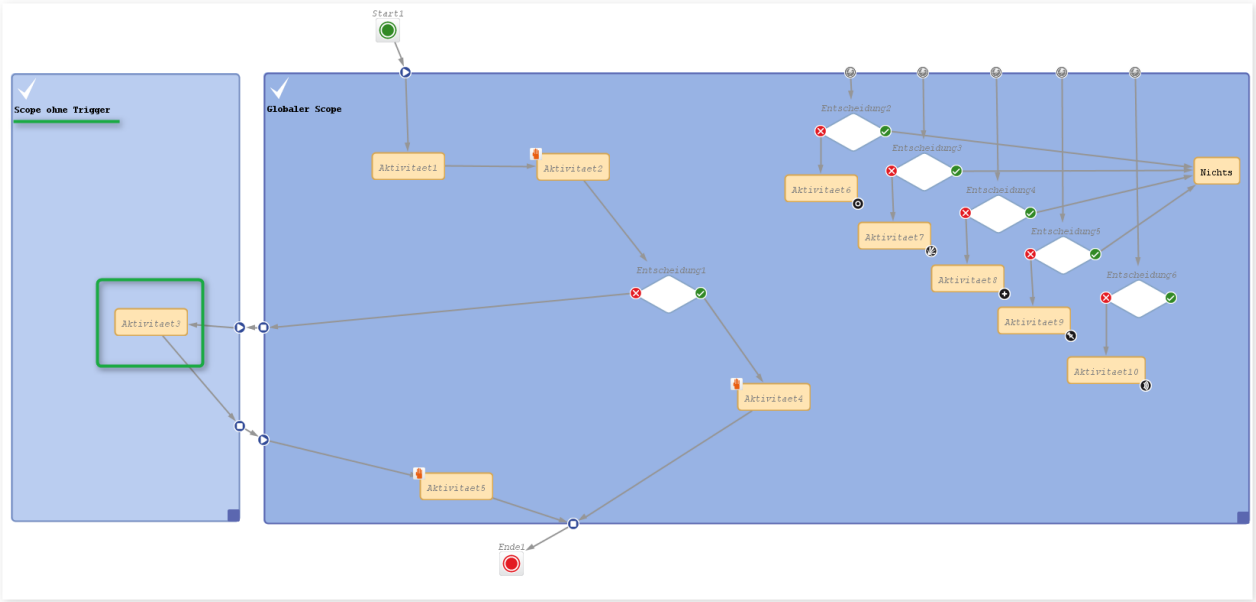


Fig. 6: ConSol*CM Process Designer - Separate Scopes mit und ohne Triggern

In diesem Beispiel wurde die Position von *Aktivität3* optimiert. Sie hätte sonst (im globalen Scope) viele Entscheidungsaufrufe (Entscheidungsknoten im Workflow) getriggert, und diese wären alle *ins Nichts* gelaufen. Das Ausführen von *Aktivität3* außerhalb des globalen Scopes stellt eine gute Workflow-Performance sicher.

7.5 Stoßen Sie keine unnötigen Ticket-Update-Events an

Vorsicht:

Vermeiden Sie unnötige Ticket-Update-Events (Java-Klasse *TicketUpdateEvent*)!

Beispielsweise kann die Zuweisung des current engineer (aktueller Bearbeiter, d.h. der Bearbeiter, der gerade eingeloggt ist und mit dem Web Client arbeitet) zu einem Ticket auf zwei Arten geschehen. Bei der einen Vorgehensweise feuert ein Ticket-Update-Event, bei der anderen nicht. Wenn es für einen Business-Anwendungsfall nicht notwendig ist, einen *TicketUpdateEvent* zu werfen, vermeiden Sie dies, da das unnötige Aufrufen von *TicketUpdateEvent* die Performance verringert.

Code, der TicketUpdateEvent anstößt

```
//diese Methode wirft einen TicketUpdateEvent, nachdem der current engineer einem Ticket
zugewiesen wird
workflowApi.assignEngineer(workflowApi.currentEngineer)
```

Code, der keinen TicketUpdateEvent anstößt

```
//diese Methode wirft KEINEN TicketUpdateEvent!
ticket.setEngineer(workflowApi.currentEngineer)
```

7.6 Wie Sie den Parameter "Automatische Aktualisierung deaktivieren" verwenden

> ⚠ **Vorsicht:**
>
> Verwenden Sie den Flag *Automatische Aktualisierung deaktivieren* für Workflow-Komponenten mit Vorsicht!

Bitte bedenken Sie, dass es die Standardeinstellung ist, d ass ein Ticket-Update-Event nach jeder Ausführung einer Aktivität gefeuert wird . Ein Ticket-Update-Event ist eine Operation, die große Auswirkungen hat und die daher mit Vorsicht verwendet werden muss.

Um Performance-Probleme zu vermeiden, können Sie das Flag *Automatische Aktualisierung deaktivieren* verwenden. Es hängt von der Business-Logik ab, ob es sinnvoll ist, diesen Flag zu verwenden oder nicht.

Wenn es zum Beispiel eine Serie von automatischen Aktivitäten gibt, ist folgendes eine gute Praktik:

- Die **erste** automatische Aktivität hat den *Automatische Aktualisierung deaktivieren* Flag **gesetzt**.
  (Sie wird den Ticket-Update-Service nach Ausführung der Aktivität **nicht** aufrufen.)
- Die **zweite** automatische Aktivität hat den *Automatische Aktualisierung deaktivieren* Flag **gesetzt**.
  (Sie wird den Ticket-Update-Service nach Ausführung der Aktivität **nicht** aufrufen.)
- Die **dritte** automatische Aktivität hat den *Automatische Aktualisierung deaktivieren* Flag **gesetzt**.
  (Sie wird den Ticket-Update-Service nach Ausführung der Aktivität **nicht** aufrufen.)
  ...
- Die **letzte** automatische Aktivität hat den *Automatische Aktualisierung deaktivieren* Flag **nicht gesetzt**.
  (Sie wird *TicketUpdateEvent* **einmal** aufrufen, am **Ende** der Pipeline!)

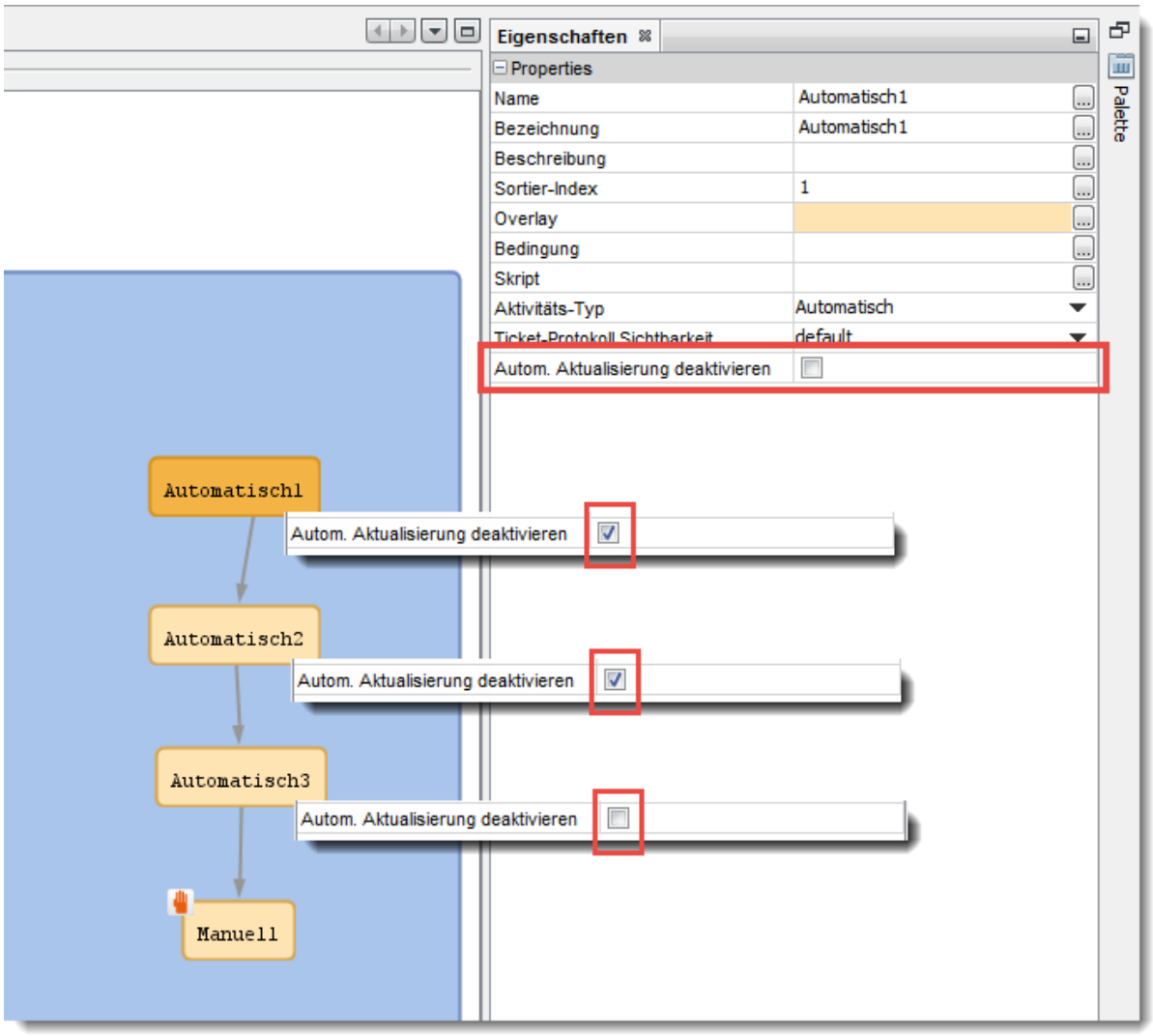


Fig. 7: ConSol*CM Process Designer - Aktivitäten mit Option "Automatische Aktualisierung deaktivieren"

7.7 Vermeiden Sie selbstauslösende Business-Event-Trigger

Wenn Sie einen Business-Event-Trigger verwenden, der von einer automatischen Aktivität gefolgt wird, seien Sie vorsichtig, dass in dieser automatischen Aktivität die Felder oder Objekte, die den Business-Event-Trigger auslösen, ⚠ nicht wieder geändert werden (dies würde den Trigger erneut feuern lassen!)

Wenn der Anwendungsfall es erforderlich macht, dass die Felder, die das Feuern des Triggers ausgelöst haben, erneut geändert werden müssen, dann muss die Logik, bei der die Felder geändert werden, in einer Aktivität außerhalb des Scopes, der den Trigger enthält, platziert werden.

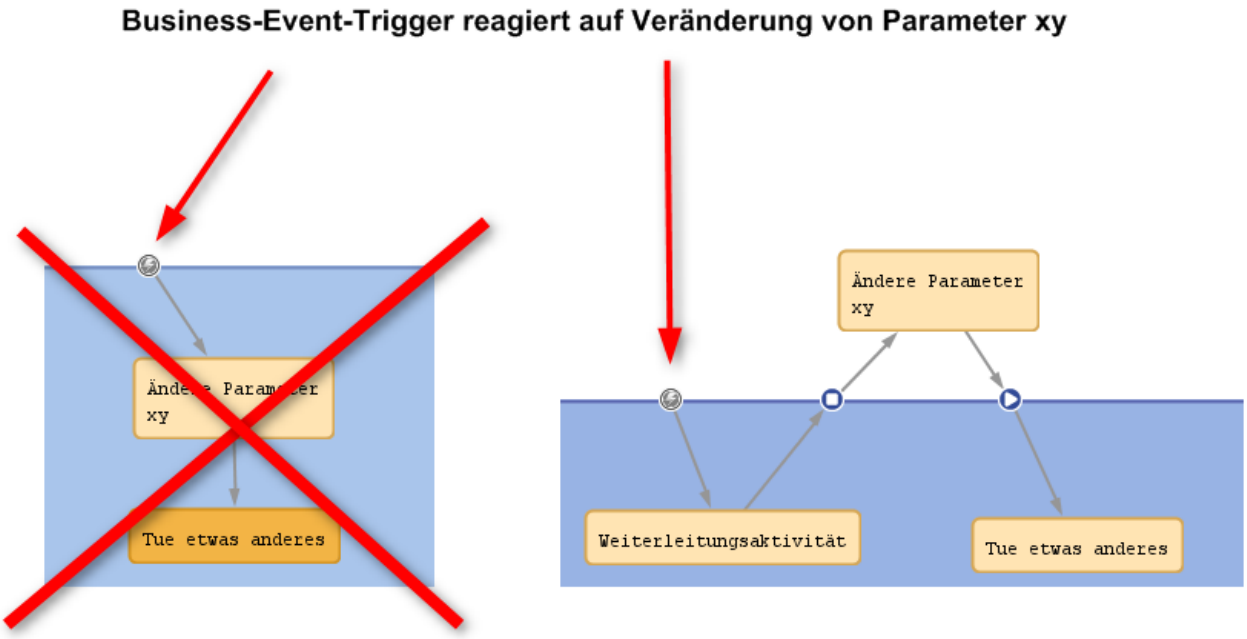


Fig. 8: ConSol*CM Process Designer - Vermeidung von selbstauslösenden Business-Event-Triggern

8 Installieren von Workflows

- Einleitung und Workflow-Lifecycle
- Für die Workflow-Installation benötigte Bearbeiter-Berechtigungen
- Aktionen während der Workflow-Installation

8.1 Einleitung und Workflow-Lifecycle

Während der Erstellung eines Workflows verwenden Sie die folgenden Funktionen, welche den Workflow-Lifecycle widerspiegeln:

- Laden  Sie den Workflow oder erstellen Sie einen neuen Workflow, z.B. Version 1.2.
- Editieren Sie den Workflow.
- Speichern  Sie den Workflow als neue Version. Dabei wird automatisch durch das System eine neue Versionsnummer verwendet, z.B. 2.0.
- Fahren Sie mit dem Editieren des Workflows fort.
- Speichern  Sie den Workflow in der aktuellen Version, z.B. Version 2.0.
- Fahren Sie mit dem Editieren des Workflows fort.
- Installieren  Sie den Workflow. Dies wird den Workflow *speichern* als neue Version und *installieren* , z.B. Version 3.0.
  Ein installierter Workflow besitzt verglichen mit der letzten gespeicherten Version immer eine größere Haupt(versions)nummer.
  Der Workflow, der vorher aktiv/installiert war, ist nun nicht mehr aktiv, stattdessen ist die neue Version des Workflows sofort in Betrieb. Das ConSol*CM-System muss dafür nicht angehalten werden.
  Die neue Version wird in der Workflow-Liste, welche bei den *Laden*- und *Speichern*-Operationen geöffnet wird, mit fetten Buchstaben und mit dem Status *ist aktuell installiert* markiert.
  Nach diesem Schritt wird die nächste gespeicherte Version als *Neue Version* gespeichert.

Vorsicht:

Seien Sie sich der Anzahl der Tickets bewusst, die übertragen werden müssen, wenn Sie einen neuen Workflow installieren! Die Installieren-Operation kann in großen Umgebungen einige Zeit benötigen! Siehe Abschnitt Aktionen während der Workflow-Installation.

8.2 Für die Workflow-Installation benötigte Bearbeiter-Berechtigungen

Ein Bearbeiter, der Workflows installieren soll, muss mindestens eine Rolle mit einer der folgenden Berechtigungen besitzen:

- **Allgemeine Berechtigungen:**
  Administrator
- **Workflow-Berechtigungen:**
  Workflow installieren

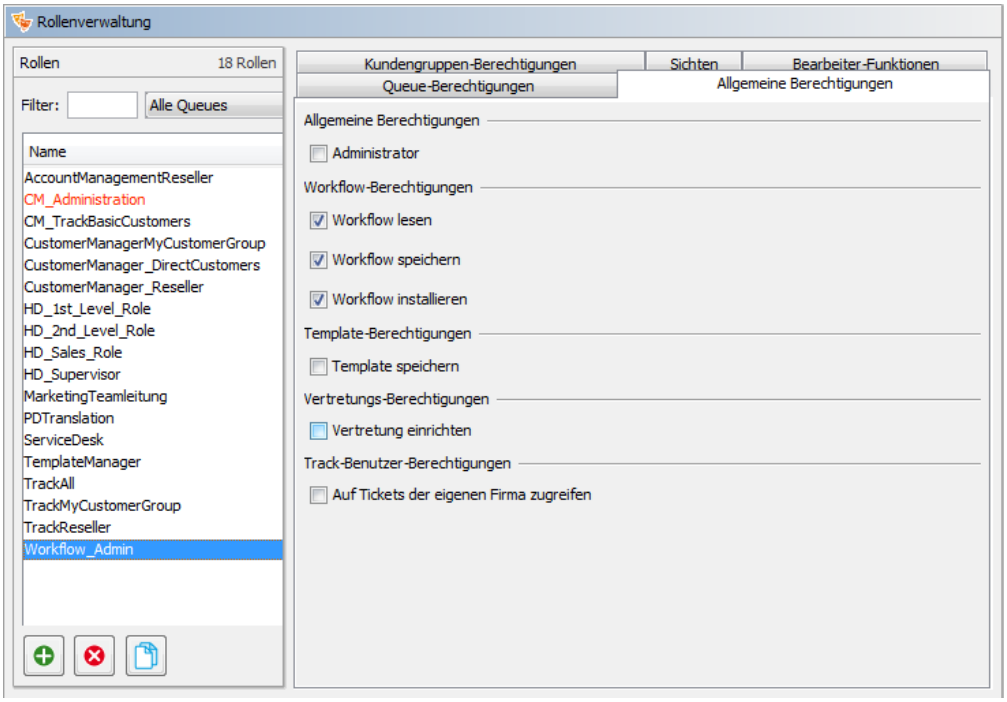


Fig. 1: ConSol*CM Admin-Tool - Bearbeiter-Berechtigungen für die Installation von Workflows

8.3 Aktionen während der Workflow-Installation

Wenn ein Workflow installiert ist, ist er sofort aktiv. Bedenken Sie daher gut, was mit offenen Tickets geschehen wird, die sich in einer Queue befinden, bei der der neue Workflow angewendet wird. Sie werden in einen neuen Workflow übertragen.

Falls Sie einen oder mehrere der folgenden Schritte durchgeführt haben:

- eine oder mehrere Aktivitäten entfernt
- eine oder mehrere automatische Aktivitäten hinzugefügt
- einen oder mehrere Trigger hinzugefügt

werden die folgenden Aktionen initiiert, nachdem Sie den *Installieren*-Button geklickt haben.

Es öffnet sich eine Anzeige, in der Sie eine Entscheidung treffen müssen, die die offenen Tickets in den entsprechenden Queues betrifft, die nicht in ihrer alten Position innerhalb des Prozesses bleiben können, da die Architektur verändert wurde:

1. Alle betroffenen Tickets so nah wie möglich an ihrer vorherigen Position beibehalten ("*Position im Prozess beibehalten*", Standard).
2. Alle betroffenen Tickets am Beginn des Prozesses neu starten lassen ("*Prozess neu starten*").

Falls Sie die erste Option gewählt haben (*Position im Prozess beibehalten*) werden die folgenden Aktionen ausgeführt:

1. Die Übertragung der Tickets beginnt.
2. Der Name der für das Ticket zuletzt ausgeführten Aktivität wird mit den Namen der Aktivitäten in der aktuellen (neuen) Workflow-Definition verglichen. Wenn die Aktivität des Tickets nicht länger in der Workflow-Definition existiert, muss eine neue Ziel-Aktivität für das Ticket gefunden werden.
3. Das *Protokoll* des Tickets wird geladen. Die Übertragungs-Engine iteriert über alle Aktivitäten, die vom Beginn der Prozess-Instanz an ausgeführt wurden und versucht eine geeignete Aktivität zu finden, d.h. eine Aktivität, die
    a. immer noch in der Workflow-Definition vorhanden ist,
    b. kein Ziel-Element für einen Trigger ist,
    c. keine Sackgassen-Aktivität darstellt.

Jedes Ticket, das seine Position nicht beibehalten kann, wird an die geeignete Position, die diesen Kriterien entspricht, verschoben. In allen diesen Fällen werden die Tickets innerhalb des Workflows rückwärts bewegt, niemals vorwärts.

Für eine *Zusammenfassung* aller Ticket-Transfers, klicken Sie im Hauptmenü auf *Anzeige* und wählen Sie *Zeige die Übertragungshistorie der Tickets an*:

- **Workflow-Name**
  Name des Workflows.
- **Version**
  Version des alten Workflows.

- **Startzeit**
  Start des Transfers. Ist die Startzeit der *Installieren*-Operation.
- **Endezeit**
  End-Zeit des Transfers. Nach dieser Zeit ist der neue Workflow vollständig in Betrieb.
- **Übertragene Tickets**
  Anzahl der Tickets, die übertragen wurden, d.h. welche vom System während der Workflow-Installation angefasst werden mussten. Sollte mit der Summe aller offenen Tickets in allen Queues, die diesen Workflow verwenden, identisch sein.
- **Details**
  Zusätzliche Informationen, die die Installation mit dem Ticket-Transfer betreffen.

In der unteren rechten Ecke der Process Designer GUI wird der allgemeine Status des Ticket-Transfers angezeigt.

9 Appendix A - Liste der Annotationen

- Alphabetische Liste der Feld-Annotationen (bis Version 6.9.3)
- Alphabetische Liste der Annotationen für Benutzerdefinierte Feldgruppen (Version 6.8 und niedriger)
- Alphabetische Liste der Annotationen für Benutzerdefinierte Feldgruppen (Version 6.9 und höher)

9.1 Alphabetische Liste der Feld-Annotationen (bis Version 6.9.3)

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
A	accuracy	validation	Definiert für Datumsfelder, wie detailliert das Datum angezeigt wird.	date (Standard)	Zeigt das Datum ohne Uhrzeit an.
				date-time	Zeigt das Datum mit Uhrzeit an.
				only-time	Zeigt nur die Uhrzeit und kein Datum an.
B	boolean-type	component-type	Definiert, wie ein boolean Feld angezeigt wird.	check box (Standard)	Checkbox, die aktiviert werden kann (standardmäßig auf *false* gesetzt, d.h. die Checkbox ist anfänglich immer deaktiviert).
				radio	2 Radiobuttons (ja/nein) zur Auswahl (nur eine kann jeweils angewählt sein).
				select	Drop-Down-Felder mit 2 Werten (ja /nein).
C	colspan	layout	Definiert, wie viele Spalten für dieses Feld	<number>	Anzahl der Spalten.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
			im Layout reserviert sind.		
	contact search result column	search-result	Definiert, ob das Feld standardmäßig in den Suchergebnissen angezeigt werden soll.	true	Entfernen Sie die Annotation, wenn das Feld nicht standardmäßig in den Suchergebnissen angezeigt werden soll.
	contains contacts	ticket contact relation	Nur für die Definition von Listenfeldern (enums). Gibt an, dass das definierte Feld Kontaktinformationen beinhalten kann.	true / false	Benötigt, um zu unterscheiden, ob die Liste innerhalb eines Kontakts (true) oder eines Tickets (false) angezeigt werden soll.
D	dialable	phone commander	Definiert ein Feld mit einer Telefonnummer.	true	Entfernen Sie diese Annotation, wenn das Feld keine wählbare Telefonnummer enthalten soll. **Version 6.9 und höher:** Nur mit CM /Phone verwendet. Markiert eine Telefonnummer als automatisch wählbar für ausgehenden

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					Anrufe für das CTI-System.
E	email	validation	Kontrolliert bei E-Mail-Adressen, ob das Format korrekt ist. d.h. ob <name>@<domain> eingegeben wurde.	true	Kann bei Benutzerdefinierten Feldern vom Typ *string* verwendet werden. Entfernen Sie die Annotation, wenn das Format nicht kontrolliert werden soll.
	enum field with ticket color	ticket display	Definiert die Hintergrundfarbe des Ticket-Icons im Ticket und in der Ticketliste.	true / false	Das Feld muss in der Verwaltung der Sortierten Listen existieren, wo Listen, Werte und Farben definiert werden.
	enum-in-search-type	component-type	Definiert, ob ein Benutzerdefiniertes Feld vom Typ *enum (Sortierte Liste)*, das für die Detailsuche verwendet wird, eine Suche über mehrere Werte erlaubt.	single (Standard) / multiple	Wenn der Wert *multiple* gesetzt ist, erlaubt dieses Feld in der Detailsuche die Suche über mehrere (über eine OR-Verknüpfung verbundene) Werte für ein Suchkriterium.
	enum-type	component-type	Layout-Definition der Anzeige einer Liste.	select (Standard)	Drop-Down-Liste für die Auswahl.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
				radio	Liste von Radiobuttons für die Auswahl (nur eine Option kann jeweils ausgewählt sein).
				autocomplete	Drop-Down-Liste für die Auswahl, hierbei ist das Feld ein Eingabefeld, das zur Filterung der Liste benutzt wird.
F	field-group	layout	Ermöglicht die Gruppierung von Feldern im Modus *Ansicht*. Die Annotation wird im Modus *Bearbeiten* ignoriert.	<string>	Um die Felder zu gruppieren, muss für jedes Feld der Gruppe der gleiche *String*-Wert eingetragen werden. Zwei oder mehr Benutzerdefinierte Felder werden verbunden, wenn sie für diese Annotation den gleichen Wert besitzen. Die Gruppe der gekoppelten Benutzerdefinierten Felder wird nur angezeigt,

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					wenn in allen von ihnen Werte gesetzt sind.
	field indexed	indexing	Gib t an, dass das Feld indiziert wird, d. h., das Feld wird in die Suche einbezogen und der Inhalt des Feldes taucht in Suchergebniss en auf.	transitive	Alle Daten werden angezeigt (Tickets und Kunden).
				unit	Für Kundendaten. Nur die Unit (Kontakt oder Firma) und ggf. die übergeordnete U nit (Firma) wi rd im Suchergebnis angezeigt, Tickets werden nicht angezeigt.
				local	Für Kundendaten. Nur die Unit wird als Suchergebnis ausgegeben, es werden keine Firma und keine Tickets angezeigt.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
				not indexed	Feld wird nicht indiziert.
	fieldsize	layout	Größe, mit der das Feld im Ticket-Layout angezeigt wird.	<rows>;<cols>	Für Benutzerdefinierte Felder vom Typ *string* mit der Annotation *text-type* und dem Wert *textarea*. <rows> definiert die Anzahl der angezeigten Zeilen und <cols> definiert die Anzahl der Zeichen, die pro Zeile angezeigt werden. Nur für Layout-Zwecke.
				<number>	Für Benutzerdefinierte Felder vom Typ *enum (Sortierte Liste)*. Definiert, wie viele Werte in der Listenbox direkt sichtbar sind. Nur für Layout-Zwecke.
	format	validation	Kontrolliert Datumsfelder auf das korrekte Format.	<date format>	Das Datumsformat basiert auf dem *SimpleDateFormat*, z.B. tt.mm.jjjj.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					Denken Sie daran, die geeignete *colspan* einzustellen, wenn Sie Stunden /Minuten hinzufügen möchten. Siehe http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html als Referenz für Datumsformate.
G	groupable	cmweb-common	Ermöglicht, die Tickets in der Ticketliste nach den Werten dieses Feldes zu gruppieren.	true	Nur für Benutzerdefinierte Felder vom Typ *enum (Sortierte Liste)*. Entfernen Sie die Annotation, um die Möglichkeit, die Ticketliste nach diesen Werten gruppieren zu können, zu deaktivieren.
L	label-group	layout	Zeigt im Modus Ansicht eine Feldgruppe zusammen mit deren Label an. Im Bearbeiten-Modus wird	<string>	Zeigt eine Gruppe Benutzerdefinierter Felder zusammen mit deren beschreibenden Label an. Die Annotation

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
			diese Annotation ignoriert.		wird im Anzeige-Modus verwendet und im Bearbeiten-Modus ignoriert. Die Gruppe kann genau ein Label besitzen (dieses ist ein Benutzerdefiniertes Feld vom Typ *String* mit der zusätzlich hinzugefügten Annotation *text-type* mit dem Wert *label*). Das Label wird angezeigt, wenn bei mindestens einem Benutzerdefinierten Feld aus seiner Gruppe ein Wert gesetzt ist. Alle Felder mit dem gleichen Label-Wert werden gruppiert und unter diesem Label angezeigt. Die Annotation *label-group* muss dem Label Beschriftung

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					ebenfalls zugewiesen werden.
	label-in-view	layout	Zeigt den Wert des Benutzerdefinierten Felds als Beschriftung im Anzeige-Modus an. Im Bearbeiten-Modus wird diese Annotation ignoriert.	true	Entfernen Sie diese Annotation, wenn die Beschriftung im Anzeige-Modus nicht sichtbar sein soll.
	ldapid ***nur Version 6.9 und höher***	contact authentication	Verwendet in einer Datenobjektgruppe vom Typ *customer (Kunde)* für das Datenobjektgruppenfeld, welches die LDAP-ID für die CM/Track-Authentifzierung enthält.		Zeigt an, dass dieses Feld als LDAP-ID für den Authentifizierungsprozess verwendet wird. Datentyp *String* ist erforderlich. Da die Definition auf dem Level Kundengruppe getroffen wird, kann die LDAP-Authentifizierung im Mixed-Mode laufen, d. h. dass LDAP für manche Kundengruppen verwendet wird und normale Authentifizierun

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					g für andere Kundengruppen.
	leave-trailing-zeros	common	Für die Anzeige von Festkommazahlen.	true / false	Verbleibende Nullen der Nachkommastellen werden nicht abgeschnitten, wenn der Wert auf *true* gesetzt ist.
	list-type	component-type	Deaktiviert die *Hinzufügen* und /oder *Löschen* Optionen bei Benutzerdefinierten Feldern vom Typ *list* oder *struct*.	fixed-size	Es ist nicht möglich, Felder und Zeilen hinzuzufügen oder zu löschen.
				non-shrinkable	Es ist nicht möglich, Felder und Zeilen zu löschen.
				non-growable	Es ist nicht möglich, Felder und Zeilen hinzuzufügen.
M	matches	validation	Kontrolliert, ob die Eingabe bei einem Benutzerdefinierten Feld vom Typ *string* mit dem angegebenen RegEx übereinstimmt.	<string>	Wird für Benutzerdefinierte Felder vom Typ *string* benutzt.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
	maxLength	validation	Definiert die maximale Länge einer Eingabe für Benutzerdefinierte Felder vom Typ *string*.	<number>	Wird für Benutzerdefinierte Felder vom Typ *string* benutzt.
	maxValue	validation	Definiert den maximalen Wert für für Benutzerdefinierte Felder vom Typ *number*.	<number>	Wird für Benutzerdefinierte Felder vom Typ *number* benutzt., d.h. *number* und *fixed-point number*.
	minLength	validation	Definiert die minimale Länge einer Eingabe für Benutzerdefinierte Felder vom Typ *string*.	<number>	Wird für Benutzerdefinierte Felder vom Typ *string* benutzt.
	minValue	validation	Definiert den minimalen Wert für für Benutzerdefinierte Felder vom Typ *number*.	<number>	Wird für Benutzerdefinierte Felder vom Typ *number* benutzt., d.h. *number* und *fixed-point number*.
N	no-history-field	performance	Gibt an, dass ein einzelnes Benutzerdefinierte Feld nicht protokolliert wird. Überschreibt die Gruppen-Annotation *no-history*.	true / false	Annotation ist aktiv, wenn der Wert auf *true* gesetzt ist. Für Felder, die gespeichert, aber nicht im Protokoll sichtbar sein sollen, wird die Annotation *visi*

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					bility configuration benutzt.
O	order-in-result	layout	Zeigt Feld als Spalte in den Suchergebnissen an der angegebenen Position an.	<number>	Die Spalten werden in aufsteigender Reihenfolge sortiert.
P	password	contact authentication	Zeigt an, dass dieses Feld beim Authentifizierungsprozess als Passwort-Feld verwendet wird.	<string>	Für CM/Track.
	position	layout	Definiert die Position eines Feldes innerhalb eines Grid-Layouts.	<number>; <number>	Die Werte definieren *Zeile* und *Spalte (Zeile;Spalte)*, Nummerierung startet mit 0;0. Wenn keine Werte gesetzt werden, nimmt das Benutzerdefinierte Feld die nächste freie Zelle des Grid-Layouts.
			Definiert die Position eines Feldes innerhalb einer struct (Unterkomponente einer list of structs)	0;<number>	Die Werte definieren *Zeile* und *Spalte* (*Zeile*;*Spalte*), nur der Wert für *Spalte* wird genutzt, der Wert für *Zeile* wird ignoriert.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
R	readonly	common	Zeigt an, dass das Benutzerdefinierte Feld nicht editiert werden kann.	true / false	Feld ist schreibgeschützt, wenn der Wert auf *true* gesetzt ist. Wird kein Wert gesetzt oder wird irgendein anderer Wert außer *false* gesetzt, wird der Wert ebenfalls als *true* behandelt.
	reportable	dwh	Gibt an, dass das Feld in Reports verwendet wird und es in das DWH übermittelt werden soll.	true / false	Feld kann in Reports verwendet werden, wenn der Wert auf *true* gesetzt ist.
	required	validation	Gibt an, dass dies ein notwendiges Feld ist.	true / false	Feld ist notwendig, wenn der Wert auf *true* gesetzt ist.
	rowspan	layout	Gibt an, wie viele Zeilen das Feld innerhalb des Layouts belegt.	<number>	Anzahl der Zeilen.
S	sortable	cmweb-common	Ermöglicht, die Tickets in der Ticketliste nach den Werten dieses Feldes zu sortieren.	true	Nur für Benutzerdefinierte Felder vom Typ *enum (Sortierte Liste)*. Entfernen Sie die Annotation, wenn Sie die

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					Möglichkeit, die Ticketliste nach den Werten dieses Feldes sortieren zu können, deaktivieren möchten. Funktioniert nur, wenn das Benutzerdefinierten Feld indiziert ist.
T	text-type	component-type	Definiert die möglichen Typen eines Feldes vom Typ *string*.	text (Standard)	Einzeiliges Eingabefeld.
				textarea	Mehrzeiliges Eingabefeld.
				password	Eingabefeld für Passwörter. Passwörter werden im Modus *Ansicht* als ******* dargestellt.
				label	Eingabe wird als Label angezeigt, d.h. das Feld wird nur angezeigt, es ist keine Dateneingabe möglich.
				url	Eingabe wird im Anzeige-Modus als Link dargestellt. Die Zeichenfolge

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					muss dafür diesem URL-Muster entsprechen. "^((?:mailto\:|(?:(?:ht|f)tps?)\://)1\S+)(?: (?:|)?(.*))?$" Beispiel: "http://consol.de ConSol*"
	ticket-list-colspan	layout	Definiert, wie viele Spalten dieses Feld in der in der Ticketliste-Box belegt.	<number>	Anzahl der Spalten.
	ticket-list-position	layout	Definiert die Position des Felds in der Ticketliste-Box.	<number>;<number>	Werte definieren *Zeile* und *Spalte* (*Zeile*;*Spalte*), Nummerierung beginnt bei 0; 0.
	ticket-list-rowspan	layout	Definiert, wie viele Zeilen dieses Feld in der in der Ticketliste-Box belegt.	<number>	Anzahl der Zeilen.
U	username	contact authentification	Zeigt an, dass dieses Feld als ein Login-Name im Authentifizierungsprozess verwendet wird.	true / false	Für CM/Track.
V	visibility	common		edit	

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
			Definiert, wann ein Feld sichtbar ist.		Feld ist im Modus *Bearbeiten* sichtbar.
				view	Feld ist im Modus *Anzeige* sichtbar.
				none	Feld ist nicht sichtbar.
					Wenn irgendein anderer Wert oder kein Wert gesetzt wird, ist das Feld immer sichtbar.
	visibility configuration	visibility	Gibt die Sichtbarkeit des Felds im Protokoll an.	on every level	Feld wird in jedem Sichtbarkeitslevel des Protokolls angezeigt.
				2nd level and 3rd level	Feld wird im zweiten und dritten Sichtbarkeitslevel des Protokolls angezeigt (*2nd level* und *3rd level*).
				only 3rd level	Feld wird nur im dritten Sichtbarkeitslevel des Protokolls angezeigt (*3rd level*).

9.2 Alphabetische Liste der Annotationen für Benutzerdefinierte Feldgruppen (Version 6.8 und niedriger)

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
C	contact history template name	ticket contact relation	Gibt an, wie Kontaktdaten im Ticketprotokoll dargestellt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben.
	contact-template-contact-ticket-page	contact-templates	Gibt an, in welchem Format Kurzinformationen über einen Kunden in einem Ticket und auf einer Kundenseite dargestellt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn diese Annotation nicht konfiguriert wird, wird das für die Annotation *contact-template-default* angegebene Template für das Format der Darstellung verwendet.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
	contact-template-default	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden dargestellt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn diese Annotation nicht konfiguriert wird, wird die (veraltete) Annotation *unit search template name* für das Format der Darstellung verwendet.
	contact-template-dragged	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden dargestellt werden, während sie mit gedrückter linker Maustaste gezogen werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wen n diese Annotation nicht konfiguriert wird, wird das für die Annotation *cont act-template-default* angegebene

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					Template für das Format der Darstellung verwendet.
	contact-template-email	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden beim Autovervollständigen von E-Mail-Empfängern dargestellt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn diese Annotation nicht konfiguriert wird, wird das für die Annotation *contact-template-default* angegebene Template für das Format der Darstellung verwendet.
	contact-template-quick-search	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden in der Ergebnisliste der Schnellsuche angezeigt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn diese Annotation nicht

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					konfiguriert wird, wird das für die Annotation *contact-template-default* angegebene Template für das Format der Darstellung verwendet.
	contact-template-search	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden in der Ergebnisliste der Kontaktsuche angezeigt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn di ese Annotation nicht konfiguriert wird, wird das für die Annotation *contact-template-default* angegebene Template für das Format der Darstellung verwendet.
	contact-template-ticket-list	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden in der	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
			Ticketliste angezeigt werden.		wird der Name dieses Templates angegeben. Wenn diese Annotation nicht k onfiguri ert wird, wird das für die Annotation *cont act-template-default* angegebene Template für das Format der Darstellung verwendet.
	contact-template-ticket-reference	contact-templates	Gibt an, in welchem Format Kurzinformatio n über einen Kunden im Bereich *Relatio nen* angezeigt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn diese Anno tation nicht konfiguriert wird, wird das für die Annotation *cont act-template-default* angegebene Template für das Format der Darstellung verwendet.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
	contact-template-ticket-search	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden in den Suchergebnissen einer Ticket-Suche angezeigt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn diese Annotation nicht konfiguriert wird, wird das für die Annotation *contact-template-default* angegebene Template für das Format der Darstellung verwendet.
	contact-template-workspace-favourite	contact-templates	Gibt an, in welchem Format Kurzinformation über einen Kunden im *Workspace* und den *Favoriten* angezeigt werden.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben. Wenn diese Annotation nicht konfiguriert wird, wird das für die Annotation *contact-template-*

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					default angegebene Template für das Format der Darstellung verwendet.
G	group-visibility	common	Definiert die standardmäßige Sichtbarkeit einer Feldgruppe.	true / false	Diese Annotationen kann durch die Annotationen der einzelnen Felder in der Gruppe überschrieben werden.
N	no-history	performance	Gibt an, dass alle Benutzerdefinierten Feldern, die zu dieser Gruppe gehören, nicht protokolliert werden.	true / false	Annotation ist aktiv, wenn der Wert auf *true* gesetzt ist. Für einzelne Felder können Sie die Annotation *no-history-field* verwenden. Mögliche Werte sind *true*, wenn diese Annotation aktiviert sein soll, oder *false*, das den gleichen Effekt wie das Entfernen der Annotation besitzt. Verwenden Sie diese Annotation, wenn Sie das

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					Protokollieren für alle/viele der Felder dieser Gruppe verhindern möchten. Wenn Sie das Protokollieren nur für ein /einige Feld(er) verhindern möchten, verwenden Sie die *no-history-field* (Annotation auf Feld-Ebene, nicht auf Feldgruppen-Ebene) für die betreffenden Felder.
O	open-at-create	layout	Lässt Feldgruppen während der Ticketerstellung sichtbar sein, auch wenn sie initial unsichtbar sind (aufgrund Annotation group-visibility).	true	Entfernen Sie diese Annotation, wenn die Gruppe nicht sichtbar sein soll.
R	reportable group	dwh	Gibt an, dass alle Benutzerdefinierten Felder, die zu dieser Gruppe gehören, in Reports	true / false	Es muss ein Wert gesetzt werden. Die Annotation ist aktiv, wenn der Wert auf *true* gesetzt ist.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
			verwendet werden und deswegen an das CMRF übermittelt werden sollen.		
S	show-contact-in-ticket-list	layout	Gibt an, dass die Feldgruppe (Kontakt) in der Ticketliste angezeigt werden soll.	true	Diese Annotation kann nur Gruppen zugewiesen werden, die die Annotation *unit is a contact* besitzen. Entfernen Sie die Annotation, wenn die Feldgruppe (Kontakt) nicht in der Ticketliste angezeigt werden soll.
	show-in-group-section	layout	Definiert, dass die Feldgruppe im Bereich *Gruppen* angezeigt wird.	true	Ohne diese Annotation wird die Gruppe im Kopfbereich des Tickets angezeigt.
U	unit is a contact	ticket contact relation	Gibt an, dass die Feldgruppe Kundendaten enthält.	true / false	Gruppe wird bei *Kunden* angezeigt, wenn Annotation auf *true* gesetzt ist und bei *Ticket*, wenn die Annotation auf *false* gesetzt ist.

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
	unit search template name ***veraltet***	indexing	Gibt an, welches Template die Darstellung von Kurzinformation über einen Kunden in den gefundenen Kontakten definiert.	<template name>	Das Format der Darstellung wird über ein Template definiert. Hier wird der Name dieses Templates angegeben.

9.3 Alphabetische Liste der Annotationen für Benutzerdefinierte Feldgruppen (Version 6.9 und höher)

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
A	auto-open-group	layout	Die Gruppe wird initial als geöffnet dargestellt. Es kann mehr als ein Wert als Komma-separierte Liste eingegeben werden (kann für die *customer*-Annotation verwendet werden.	ticket:create	Gruppe ist initial geöffnet, wenn ein neues Ticket erstellt wird.
				customer:create	Gruppe ist initial geöffnet, wenn ein neuer Kunde erstellt wird.
				customer:view	Gruppe ist initial geöffnet, wenn die Kundeseite (Kontakt oder Firma) geöffnet wird.
G	group-visibility	common	Definiert die Standard-Sichtbarkeit einer Benutzerdefinierten Feldgruppe.	true / false	Die Annotation kann auf Feld-Ebene (d.h. einer Annotation für ein einzelnes Benutzerdefinie

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
					rtes Feld) überschrieben werden.
N	no-history	performance	Zeigt an, das alle Benutzerdefinierten Felder, die zu dieser Gruppe gehören, nicht protokolliert werden.	true / false	Mögliche Werte sind *true*, wenn diese Annotation aktiviert sein soll,oder *false*, das den gleichen Effekt wie das Entfernen der Annotation besitzt. Verwenden Sie diese Annotation, wenn Sie das Protokollieren für alle/viele der Felder dieser Gruppe verhindern möchten. Wenn Sie das Protokollieren nur für ein /einige Feld(er) verhindern möchten, verwenden Sie die *no-history-field* (Annotation auf Feld-Ebene, nicht auf Feldgruppen-Ebene) für die betreffenden Felder.
R		dwh		true / false	

	Name	Annotationstyp	Beschreibung	Werte	Kommentar
	reportable group		Gibt an, dass alle Benutzerdefinierten Felder, die zu dieser Gruppe gehören, in Reports verwendet werden und deswegen an das CMRF übermittelt werden sollen.		Es muss ein Wert gesetzt werden. Annotation ist aktiv, wenn der Wert auf *true* gesetzt ist.
S	show-contact-in-ticket-list		Obsolet! Verwenden Sie die Seitenanpassung! accordionTicketList.mainCustomerDescriptionVisible={true, false}	obsolet	
	show-in-group-section	layout	Definiert, dass eine Benutzerdefinierte Feldgruppe im Bereich Gruppen (als Tab) angezeigt wird.	true / false	Ohne diese Annotation wird die Gruppe ohne Tabs in den Ticketdaten oder Bereich Kontakt angezeigt.
U	unit is a contact ***veraltet***	ticket contact relation		true/false	Entfernt ab Version 6.9.0.

10 Appendix B - Glossar

	Begriff	Erklärung
A	ACIM	Aktivitäts-Item, Eintrag im Ticketbereich *Protokoll* eines Tickets (z.B. Kommentar, E-Mail, Attachment, Zeitbuchungseintrag).
	AD	Microsoft Active Directory - ein LDAP-basierter Verzeichnisservice für Microsoft Windows Domäne Netzwerke.
	Admin-Tool	Grafische Applikation, um ein ConSol*CM-System zu konfigurieren und zu verwalten. Verwendet Java Web Start.
B	Bearbeiter	Benutzer, der einen Login für den Web Client besitzt und der die Aufgaben, die das Ticket betreffen, ausführt.
	BI	Business Intelligence: Methoden, Technologien und Architekturen, um Daten in nützliche Informationen für Business-Zwecke zu verwandeln.
C	CMDB	ConSol*CM Datenbank, die Arbeitsdatenbank des CM-Systems.
	CMRF	ConSol*CM Reporting Framework: Eine Java EE Applikation, die Daten zwischen der CM-Datenbank und dem DWH synchronisiert.
	CM/Office	Ein Modul von ConSol*CM, das es Bearbeitern mittels des Web Clients ermöglicht, mit MS-Word Dokumenten zu arbeiten, die mit

	Begriff	Erklärung
		ConSol*CM-Ticket- oder Kundendaten vorausgefüllt sind.
	CM/Track	ConSol*CM-Webportal: Ermöglicht Kunden Zugriff zum ConSol*CM-System.
	CTI	Computer-Telefon-Integration (Computer telephony integration), eine Beschreibung für jede Technologie, die es erlaubt, Interaktionen zwischen einem Telefon und einem Computer zu integrieren und koordinieren.
D	Datenobjekt	Ein Kunde, Kontakt oder eine Firma. Ehemals *Unit*.
	Datenobjektgruppe	Eine Gruppe von Feldern, in denen Daten für Kunden (Kontakt oder Firma) gespeichert werden können. Ähnlich den Benutzerdefinierten Feldgruppen für Ticketdaten.
	Datenobjektgruppenfeld	Ein Feld, in dem Daten für Kunden (Kontakt oder Firma) gespeichert werden kann. Ähnlich den Benutzerdefinierten Feldern für Ticketdaten.
	DWH	Data Warehouse: CM-Datenbank, die für das Reporting und die Datenanalyse verwendet wird.
E	ESB	Enterprise Service Bus: Enterprise Service Bus, eine Software-Architektur, die für die Kommunikation zwischen gemeinsam interagierenden Software-Applikationen in einer Serviceorientierten Architektur (SOA) verwendet wird.
	ERP-System	

	Begriff	Erklärung
		Enterprise Resource Planning: Häufig verwendet für diesen Typ von Enterprise Management Software.
	ETL	Extract Transform Load: Extrahiert Daten aus einer Quelle (dies kann eine Datenbank oder eine andere Quelle sein), wandelt diese um und lädt sie in eine andere Datenbank, z.B. ein Data Warehouse.
F	FlexCDM	Das *Flexible Kundendatenmodell,* das Kundendatenmodell, welches in ConSol*CM-Version 6.9 eingeführt wurde. Für jede Kundengruppe kann ein eigenes Kundendatenmodell definiert werden.
	Firma	Ein Datenobjekt des Typs Firma. Häufig ist dies eine reale Firma oder Institution, es kann aber auch etwas anderes sein, wie eine Maschine oder ein Schiff.
G	GUI	Graphical User Interface (Grafische Benutzeroberfläche)
H	Hauptkunde	Der Kunde, der der Hauptkunde eines Tickets ist. Beginnend mit ConSol*CM-Version 6.9 kann dies entweder ein Kontakt oder eine Firma sein.
I	IMAP	Internet Message Access Protocol: Internet-Standardprotokoll für den Zugriff auf E-Mails auf einem Remote-E-Mail-Server. Kann als einfaches (plain) IMAP oder sicheres (secure) IMAP (IMAPs) verwendet werden. Im

	Begriff	Erklärung
		letzteren Fall wird ein gültiges Zertifikat benötigt.
J	Java EE	Java Enterprise Edition
	JMS	Java Message Service: Java EE Komponente, um Nachrichten zwischen JMS-Clients zu versenden.
K	Kontakt	Der Kunde, der eine Frage oder eine Service-Anfrage hat.
	KPI	Key Performance Indicator: Parameter für die Performance-Messung für Firmen, Projekte etc.
	Kunde	Allgemeiner Ausdruck für Kundenobjekte in ConSol*CM. Ein Kunde kann ein Kontakt oder eine Firma sein. Technisch gesehen ist ein Kunde ein Datenobjekt. Die entsprechende Java-Klasse ist Unit.
L	LDAP	Lightweight Directory Access Protocol: Applikationsprotokoll, für den Zugriff auf und die Pflege von Datenverzeichnisinformationen über ein IP-Netzwerk.
M	Mule	Ein Open Source Java-basierter Enterprise Service Bus (ESB).
P	PCDS	Page Customization Definition Section (Bereich der Seitenanpassung)
	Pentaho	Pentaho™ ist eine Business Intelligence (BI) Programmsammlung, die als Open-Source- und als Enterprise-Version verfügbar ist.
	POP	Post Office Protocol: Standard-Internetprotokoll für

	Begriff	Erklärung
		den Fernzugriff auf einen E-Mail-Server mittels TCP/IP. Kann als einfaches POP oder als verschlüsseltes POP (POPs) benutzt werden. Im letzteren Fall werden die geeigneten Zertifikate benötigt.
	Portal	CM/Track: Ermöglicht Kunden Zugriff zum ConSol*CM-System.
	Postfach	Zielort, an den E-Mails zugestellt werden. Postfächer werden auf einem Mail-Server verwaltet. ConSol*CM kann E-Mails von einem oder mehreren getrennten Postfächern abrufen.
	Process Designer	ConSol*CM-Komponente für das Design, die Erstellung und das Installieren von Workflows. Verwendet Java Web Start.
Q	Queue	Enthält Tickets aus gleichen Bereichen und stellt sicher, dass alle Tickets dieses Bereiches auf die gleiche Weise behandelt werden. Eine Queue besitzt immer einen Workflow. Zugriffsrechte und andere Parameter werden immer auf der Basis von Queues definiert.
R	RDBMS	Relational Database Management System: z.B. Oracle ® , MS SQL Server ® , MySQL.
	REST	Representational State Transfer: Methode, um Daten mittels eines Netzwerks zu übermitteln, basierend auf HTTP.
	Rolle	Definiert die Zugriffsrechte und Sichten eines Bearbeiters.

	Begriff	Erklärung
S	Sicht	Eine Auswahl von Tickets basierend auf Scopes (Bereichen) aus einem oder verschiedenen Workflows. Sichten werden einer Rolle zugewiesen und sich in der Ticketliste im Web Client sichtbar.
	Skript	Programme, die für eine spezielle Laufzeit-Umgebung geschrieben werden, die die Ausführung von Aufgaben interpretieren und automatisieren können. Im ConSol*CM werden Skripte im Admin-Tool gespeichert und als Skripte für Aktivitäten in Workflows gespeichert.
	SMTP	Simple Message Transfer Protocol: Standard-Protokoll für den Versand von E-Mails.
T	TAPI	Telephony Application Programming Interface: Microsoft Windows API, welche Computer-Telefonie-Integration bereitstellt und es Microsoft-Windows-PCs ermöglicht, Telefon-Services zu verwenden.
	Template	Vorformatiertes Beispiel betreffend Layout, Text und/oder Daten, z.B. für E-Mails oder CM /Office.
	Ticket	Vorfall, Service-Fall oder andere Anfrage eines internen oder externen Kunden. Ein Ticket ist das Objekt, das den Prozess durchläuft (definiert durch den Workflow).
W	Workflow	

	Begriff	Erklärung
		Modelliert Schritt für Schritt einen Prozess, der mittels ConSol*CM verwaltet werden soll.
Z	Zugriffsrechte	Berechtigungen eines Bearbeiters, um ein Ticket im Web Client zu sehen oder zu verändern. Zugriffsrechte werden immer an eine Gruppe vergeben, niemals an einzelne Bearbeiter.
	Zusätzlicher Kunde	Kunde (Kontakt oder Firma) neben dem Hauptkunden, z.B. ein Mitarbeiter der Firma. Zusätzlichen Kunden können Rollen zugewiesen werden.

11 Appendix C - System-Properties

Die Liste enthält eine Erklärung aller im ConSol*CM-System vorhandenen System-Properties. Sie können die System-Properties im Admin-Tool im Bereich *Allgemeine Konfiguration* , *Erweitert* definieren.

- Appendix C - System-Properties
  - System-Properties sortiert nach Modul
  - System-Properties sortiert nach Name der System-Property

11.1 System-Properties sortiert nach Modul

Modul	System-Property	Erklärung
cmas-app-admin-tool	admin.tool.session.check.interval	*Beschreibung:* Intervall, in dem inaktive (beendete) Sitzungen im Admin-Tool überprüft werden (in Sekunden) *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 30 *Seit:* 6.7.5
cmas-app-admin-tool	autocomplete.enabled ***nur Version 6.9 und höher***	*Beschreibung:* Wenn diese Property fehlt oder der Wert *false* ist, wird der Tab *Adresse Autocomplete* im Admin-Tool ausgeblendet. *Typ:* Boolean *Neustart erforderlich:* Nein *System:* Ja *Optional:* Ja *Beispielwert:* true *Seit:* 6.9.2.0
cmas-core-cache	cache-cluster-name	*Beschreibung:* Cache-Cluster-Name des JBoss *Typ:* String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 635a6de1-629a-4129-8299-2d98633310f0 *Seit:* 6.4.0
cmas-core-cache	eviction.event.queue.size	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 200000 *Seit:* 6.4.0
cmas-core-cache	eviction.max.nodes	

Modul	System-Property	Erklärung
		Beschreibung: *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 100000 *Seit:* 6.4.0
cmas-core-cache	eviction.wakeup.interval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 3000 *Seit:* 6.4.0
cmas-core-index-common	big.task.minimum.size	*Beschreibung:* Gibt an, wie viele Teile ein Task mindestens haben soll, damit er vom Indexer mit Priorität *low* behandelt wird. *Typ:* Integer *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Beispielwert:* 15 (Standard) *Seit:* 6.8.3
cmas-core-index-common	disable.admin.task.auto.commit	*Beschreibung:* Alle Tasks, die für ein Index-Update erstellt werden, werden automatisch direkt nach ihrer Erstellung ausgeführt *Typ:* Boolean *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Beispielwert:* false *Seit:* 6.6.1
cmas-core-index-common	index.attachment	*Beschreibung:* Beschreibt, ob der Inhalt von Attachments indziert wird. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja

Modul	System-Property	Erklärung
		Optional: Nein *Beispielwert:* true *Seit:* 6.4.3
cmas-core-index-common	index.history	*Beschreibung:* Beschreibt, ob die Unit und das Ticket-Protokoll indiziert werden *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.1.0
cmas-core-index-common	index.status	*Beschreibung:* Status des Indexers, mögliche Werte sind RED, YELLOW, GREEN, werden im Admin-Tool angezeigt *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* GREEN *Seit:* 6.6.1
cmas-core-index-common	index.task.worker.threads	*Beschreibung:* Beschreibt, wie viele Threads benutzt werden, u m Batch-Index-Aufgaben au szuf ühren (Synchronisierung, Administrations- und Reparatur-Aufgaben). *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1 (Standard) (Wir empfehlen, einen Wert zu benutzen, der nicht größer als zwei ist) *Seit:* 6.6.14, 6.7.3
cmas-core-index-common	index.version.current	*Beschreibung:* Enthält Informationen über die derzeitige (möglicherweise veraltete) Index-Version.

Modul	System-Property	Erklärung
		Typ: Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1 (Standard) *Seit:* 6.7.0
cmas-core-index-common	index.version.newest	*Beschreibung:* Enthält Informationen, welche Index-Version als die neueste betrachtet wird. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1 (Standard) *Seit:* 6.7.0
cmas-core-index-common	indexed.assets.per.thread.in.memory	*Beschreibung:* Beschreibt, wie vi ele Assets während des Indizierens pro Thread auf einmal in den S peicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 200 (Standard) *Seit:* 6.8.0
cmas-core-index-common	indexed.engineers.per.thread.in.memory	*Beschreibung:* Beschreibt, wie viele Bearbeiter während des Indizieren s pro Thread auf einma l in den Speicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 300 (Standard) *Seit:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.tickets.per.thread.in.memory	*Beschreibung:* Beschreibt, wie viele Tickets während des Indizierens pro Thread auf einm

Modul	System-Property	Erklärung
		al in den Speicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 100 (Standard) *Seit:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.units.per.thread.in.memory	*Beschreibung:* Beschreibt, wie viele Units während des Indizierens pro Thread auf ein m al in den Speicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 200 (Standard) *Seit:* 6.6.14, 6.7.3
cmas-core-index-common	synchronize.master.address	*Beschreibung: Wert der Java System Property -Dcmas.http.host.port, die angibt, unter welcher URL der Index-Master erreichbar ist. Standard ist Null. Seit CM Version 6.6.17 ist dieser Wert beim Set-Up konfigurierbar, um den initialen Index-Master-Server zu bestimmen. Bitte beachten Sie, dass das Verändern dieses Wertes nur erlaubt ist, wenn alle Cluster-Nodes zum Empfang von Index-Veränderungen gestoppt sind.* *Typ: Integer* *Neustart erforderlich: Nein* *System: Ja* *Optional: Ja* *Beispielwert: 127.0.0.1:80* *Seit: 6.6.0*
cmas-core-index-common	synchronize.master.security.token	*Beschreibung:* Das Passwort für den URL-Zugriff auf den Index-Snapshot, z.B. für die Index-

Modul	System-Property	Erklärung
		Synchronisation oder für Backups. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* token *Seit:* 6.6.0
cmas-core-index-common	synchronize.master.security.user	*Beschreibung:* Der Benutzername für den URL-Zugriff auf den Index-Snapshot, z.B. für die Index-Synchronisation oder für Backups. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* user *Seit:* 6.6.0
cmas-core-index-common	synchronize.master.timeout.minutes	*Beschreibung:* Beschreibt, wie oft die Index-Synchronisation ausgehend vom akiuellen Master-Server fehlschlagen darf , bis ein neuer Master für die Index-Reparatur ausgewählt wird. Standard ist 5. Seit CM Version 6.6.17 ist dieser Wert im Setup konfigurierbar, wobei 0 bedeutet, dass der Master-Server nie geändert wird (Failover-Mechanismus deaktiviert). *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Seit:* 6.6.0
cmas-core-index-common	synchronize.megabits.per.second	*Beschreibung:* Beschreibt, wie viel Bandbreite der Master-Server verbrauchen darf, um

Modul	System-Property	Erklärung
		Index-Veränderungen an die Slave-Server zu übermitteln. Standard ist 85. Bitte benutzen Sie nicht die gesamte verfügbare Bandbreite, um die Index-Veränderungen zwischen den Hosts zu übermitteln, da dies dafür sorgen kann, dass die Nodes des Clusters nicht mehr synchron sind. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 85 *Seit:* 6.6.0
cmas-core-index-common	synchronize.sleep.millis	*Beschreibung:* Beschreibt, wie oft jeder Slave-Server den Master-Server auf Veränderungen des Indexes abfragt. Standard ist 1000. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1000 *Seit:* 6.6.0
cmas-core-security	admin.email	*Beschreibung:* Die E-Mail-Adresse des ConSol*CM Administrators. Anfänglich wird hier der Wert genommen, den Sie beim System-Setup eingegeben haben. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0
cmas-core-security	admin.login	*Beschreibung:* Der (Login-) Name des ConSol*CM Administrators . Anfänglich wird

Modul	System-Property	Erklärung
		hier der Wert genommen, den Sie beim System-Setup eingegeben haben. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* admin *Seit:* 6.0
cmas-core-security	authentication.method	*Beschreibung:* Methode der Engineer (Bearbeiter)-Authentifizierung für den Web Client (interne CM-Datenbank oder LDAP-Authentifizierung). Erlaubte Werte sind *LDAP* oder *DATABASE*. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* DATABASE *Seit:* 6.0
cmas-core-security	contact.authentication.method ***nur Version 6.9 und höher***	*Beschreibung:* Definiert die Kontakt-Authentifizierungsmethode für CM/Track, mögliche Werte sind *DATABASE* oder *LDAP* oder *LDAP,DATABASE* oder *DATABASE,LDAP.* *Typ:* String *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Seit:* 6.9.3.0
cmas-core-security	contact.inherit.permissions.only.to.own.customer.group ***nur Version 6.9 und höher***	*Beschreibung:* Zeigt an, ob der authentifizierte Kontakt in CM/Track alle Kundengruppen-Berechtigungen vom repräsentierenden Bearbeiter erbt (*false*), oder nur die Berechtigungen der eignen Kundengruppe erbt (*true*).

Modul	System-Property	Erklärung
		Typ: Boolean *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Seit:* 6.9.2.3
cmas-core-security	kerberos.v5.enabled	*Beschreibung:* Kennzeichnung, welche anzeigt, ob SSO mit Kerberos aktiviert ist. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false (Standard, wenn Kerberos während des System Setups nicht aktiviert wurde) *Seit:* 6.2.0
cmas-core-security	kerberos.v5.username.regex	*Beschreibung:* Regulärer Ausdruck, der für die Zuweisung des Kerberos Principals zum CM-Bearbeiter-Login zuständig ist. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* (.*)@.* *Seit:* 6.2.0
cmas-core-security	ldap.authentication	*Beschreibung:* Authentifizierungsmethode für den Web Client, die bei der LDAP-Authentifizierung benutzt wird. *Typ:* String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* simple *Seit:* 6.0
cmas-core-security	ldap.basedn	*Beschrei bung:* Base DN fü r die Suche von LDAP-Benutzer-Accounts (LDAP

Modul	System-Property	Erklärung
		Authentifizierung im Web Client), wenn LDAP-Authentifizierung verwendet wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* OU=accounts, DC=consol,DC=de *Seit:* 6.0
cmas-core-security	ldap.contact.name.basedn ***nur Version 6.9 und höher***	*Beschreibung:* BaseDN für die Suche n ach Kontakt-DN mitt els LDAP-ID (z.B. ou=accounts, dc=consol,dc=de) (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.password ***nur Version 6.9 und höher***	*Beschreibung:* Passwort für die Suche nach Kontakt-DN mittels L DAP-ID. Wenn nicht gesetzt, wird ein anonymer Account verwendet. (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.providerurl ***nur Version 6.9 und höher***	*Beschreibung:* Adresse des LDAP-Servers (ldap[s]://host: port). (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.searchattr	

Modul	System-Property	Erklärung
	nur Version 6.9 und höher	*Beschreibung:* Attribut für die Suc he nach Kontakt-DN mittels LDAP-ID (z.B. uid). (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.userdn ***nur Version 6.9 und höher***	*Beschreibun g :* Benutzer-DN für die Suche nach Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymer Account verwendet. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Since:* 6.9.3.0
cmas-core-security	ldap.initialcontextfactory	*Beschreibung:* Name der Klas se für initial context factory der LDAP-Impl ementierung, wenn LDAP-Authentifizierung verwendet wird. Ist üblicherweise *com.sun.jndi.ldap. LdapCtxFactory* (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* com.sun.jndi.ldap. LdapCtxFactory *Seit:* 6.0
cmas-core-security	ldap.password	*Beschreibung:* Passwort für die Verbindung zum LDAP, um Benutzer zu suchen (wenn LDAP-Authentifizierung verwendet wird). Wird nur benötigt, wenn die Suche nicht anonym durchgeführt werden

Modul	System-Property	Erklärung
		kann. (LDAP Authentifizierung im Web Client) *Typ:* Password *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.1.2
cmas-core-security	ldap.providerurl	*Beschreibung:* LDAP-Provider (wenn LDAP-Authentifizierung im Web Client verwendet wird). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* ldap://ldap.consol.de:389 *Seit:* 6.0
cmas-core-security	ldap.searchattr	*Beschreibung:* Such-Attribute für die Suche nach LDAP-Einträgen, die mit dem CM6-Login verbunden sind. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* uid *Seit:* 6.0
cmas-core-security	ldap.userdn	*Beschreibung:* LDAP-Benutzer für die Verbindung zum LDAP, um Benutzer zu suchen (wenn LDAP-Authentifizierung verwendet wird). Wird nur benötigt, wenn die Suche nicht anonym durchgeführt werden kann. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.1.2

Modul	System-Property	Erklärung
cmas-core-server	attachment.allowed.types	*Beschreibung:* Komma-separierte Liste der erlaubten Dateinamen-Erweiterungen (wenn keine Werte definiert werden, sind alle Dateinamen-Erweiterungen erlaubt). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* txt,zip,doc *Seit:* 6.5.0
cmas-core-server	attachment.max.size	*Beschreibung:* Maximal Größe von Attachments in MB. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 100 *Seit:* 6.4.0
cmas-core-server	config.data.version	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 11 *Seit:* 6.0
cmas-core-server	defaultCommentClassName	*Beschreibung:* Standard-Textklasse für Kommentare. *Typ:* String *Neustart erforderlich*: Nein *System*: Nein *Optional*: Ja *Beispielwert:* *Seit:* 6.3.0
cmas-core-server	defaultIncommingMailClassName	*Beschreibung:* Standard-Textklasse für eingehende E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Nein

Modul	System-Property	Erklärung
		Optional: Ja *Beispielwert:* *Seit:* 6.3.0
cmas-core-server	defaultOutgoingMailClassName	*Beschreibung:* Standard-Textklasse für ausgehende E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Nein *Optional*: Ja *Beispielwert:* *Seit:* 6.3.0
cmas-core-server	fetchSize.strategy	*Beschreibung:* Auswahl der Strategie, für JDBC Ergebnis-Sets *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy *Seit:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizeFixedStrategy.value	*Beschreibung:* Gibt den Wert für Abhol-Größen an, wenn die ausgewählte Strategie für die Abholung der Größe FetchSizeFixedStrategy ist. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 150 *Seit:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizePageBasedStrategy. limit	*Beschreibung:* Gibt den Wert für Abhol-Größen an, wenn die ausgewählte Strategie für die Abholung der Größe FetchSizePageBasedStrategy ist.

Modul	System-Property	Erklärung
		Typ: Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10000 *Seit:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizeThresholdStrategy. value	*Beschreibung:* Gibt den Grenzwert für Abhol-Größen an , wenn die ausgewählte Strategie für die Abholung der Größe Fetch SizeThresholdStrategy ist. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 150,300,600,1000 *Seit:* 6.8.4.1
cmas-core-server	last.config.change	*Beschreibung:* Zufällige UUID, die während der letzten Veränderung der Konfiguration der Benutzerdefinierten Felder (via Admin-Tool) generiert wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 2573c7b7-2bf5-47ff-b5a2-bad31951a266 *Seit:* 6.1.0, 6.2.1
cmas-core-server	ldap.certificate.basedn	*Beschreibu ng:* Base DN für den Ort der Zertifikate im LDAP-Verzeichnisbaum. Wen n nichts angegeben wird, wird ldap. basedn verwendet. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* OU=accounts, DC=consol,DC=de *Seit:* 6.8.4

Modul	System-Property	Erklärung
cmas-core-server	ldap.certificate.content.attribute	*Beschreibung:* LDAP-Attribut-Name der angibt, wo Zertifikatsdaten im LDAP-Verzeichnisbaum gespeichert werden. Standardwert ist: usercertificate (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* usercertificate *Seit:* 6.8.4
cmas-core-server	ldap.certificate.password	*Beschreibung:* Passwort des LDAP-Zertifikate-Managers. Wenn nichts gesetzt wird, wird ldap.password genommen. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.8.4
cmas-core-server	ldap.certificate.providerurl	*Beschreibung:* URL des LDAP-Zertifikate-Providers. Wenn nichts gesetzt wird, wird ldap.providerurl genommen. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* ldap://ldap.consol.de:389 *Seit:* 6.8.4
cmas-core-server	ldap.certificate.searchattr	*Beschreibung:* LDAP-Attribut-Name, der für die Suche nach Zertifikaten im LDAP-Verzeichnisbaum verwendet wird. Standardwert ist: mail. (LDAP Authentifizierung im Web

Modul	System-Property	Erklärung
		Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* mail *Seit:* 6.8.4
cmas-core-server	ldap.certificate.userdn	*Beschreibung:* DN des LDAP-Zertifikate-Managers. Wenn nichts gesetzt wird, wird ldap.userdn genommen. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.8.4
cmas-core-server	mail.notification.engineerChange	*Beschreibung:* Gibt an, ob eine Benachrichtigungsmail verschickt wird, wenn der Bearbeiter eines Tickets wechselt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.1.0
cmas-core-server	mail.notification.sender	*Beschreibung: From*-Adresse der Benachrichtigungsmails, die verschickt werden, wenn der Bearbeiter eines Tickets wechselt. Wenn kein Wert gesetzt wird, wird hierfür *cmas-core-security admin.email* benutzt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja

Modul	System-Property	Erklärung
		Beispielwert: cm6notification@cm6installation *Seit:* 6.6.3
cmas-core-server	mail.smtp.email	*Beschreibung:* SMTP-Mail-URL für ausgehende EMails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* smtp://mail.consol.de:25 *Seit:* 6.0
cmas-core-server	mail.smtp.envelopesender	*Beschreibung:* E-Mail-Adresse, die als Absender im SMTP-Envelope benutzt wird. Wenn nichts eingetragen wird, wird die FROM-Adresse der E-Mail benutzt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* mysender@mydomain.com *Seit:* 6.5.7
cmas-core-server	max.licences.perUser	*Beschreibung:* Setzt die maximale Anzahl von Lizenzen, die ein einzelner Benutzer benutzen kann (z.B. durch Einloggen von einem anderen Browser aus). Standardmäßig ist dieser Wert nicht beschränkt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10 *Seit:* 6.8.4.5
cmas-core-server	monitoring.engineer.login ***nur Version 6.9 und höher***	*Beschreibung:* Login des Monitoring-Bearbeiters *Typ:* String

Modul	System-Property	Erklärung
		Neustart erforderlich: Nein *System*: Ja *Optional*: Ja *Beispielwert:*bartek *Seit:* 6.9.3.0
cmas-core-server	monitoring.unit.login ***nur Version 6.9 und höher***	*Beschreibung:* Login des Monitoring-Benutzers *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:*bartek *Seit:* 6.9.3.0
cmas-core-server	server.session.archive.reaper.interval	*Beschreibung:* Reaper-Intervall (in Sekunden) von archivierten Server-Sessions. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 60 *Seit:* 6.7.1
cmas-core-server	server.session.archive.timeout	*Beschreibung:* Timeout der Gültigkeit der Server-Session-Archive (in Tagen). Nach diesem Zeitraum werden die Informationen zur Session aus der Datenbank entfernt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 31 *Seit:* 6.7.1
cmas-core-server	server.session.reaper.interval	*Beschreibung:* Interval (in Sekunden) , in dem der sog. *Reaper* inaktive (= beendete) Server-Sessions löscht (aus der Datenbank entfernt) *Typ:* Integer *Neustart erforderlich:* Nur Server

Modul	System-Property	Erklärung
		Session Service *System*: Ja *Optional*: Nein *Beispielwert:* 60 *Seit:* 6.6.1, 6.7.1
cmas-core-server	server.session.timeout	*Beschreibung:* Server-Session-Timeout (in Sekunden) für verbundene Clients. Jeder Client kann dieses Timeout mit benutzerdefinierten Werten mittels seiner ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8 please use PORTER), ETL, REST), die an den Namen der System-Property angehängt wird, überschreiben z. B. server.session.timeout.ADMIN_TOOL *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1800 *Seit:* 6.6.1, 6.7.1
cmas-core-server	tickets.delete.size	*Beschreibung:* Definiert die Anzahl der Tickets, die pro Transaktion gelöscht werden. Standardmäßig ist dieser Wert 10. *Typ:* Integer *Neustart erforderlich:* Only Session Service *System*: Ja *Optional*: Nein *Beispielwert:* 10 *Seit:* 6.8.1
cmas-core-server	ticket.delete.timeout	*Beschreibung:* Übermittlungs-Timeout (in Sekunden) für einen Unit Replacement Aktionsschritt *Typ:* Integer *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Nein *Beispielwert:* 120 *Seit:* 6.8.2
cmas-core-server	unit.replace.batchSize	*Beschreibung:* Beschreibt die Anzahl der Objekte, die bei einer Unit-Replace-Aktion (Übertrag von Tickets von einem Kontakt auf einen anderen) verarbeitet werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Seit:* 6.8.2
cmas-core-server	unit.replace.timeout	*Beschreibung:* Übermittlungs-Timeout (in Sekunden) für einen Unit-Replacement-Aktionsschritt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 120 *Seit:* 6.8.2
cmas-core-server	unused.content.remover.cluster.node.id ***nur Version 6.9 und höher***	*Beschreibung:* Knoten, der die Cluster Node ID angibt, auf der der Content Remover läuft, welcher ungenutzte Ticket-Attachments und Unit-Inhaltseinträge entfernt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 1 (in der Annahme, dass der Cluster-Node mit -Dcmas.clusternode.id=1 parameter gesetzt ist) *Seit:* 6.9.0.0
cmas-core-server	unused.content.remover.enabled	

Modul	System-Property	Erklärung
	nur Version 6.9 und höher	*Beschreibung:* Gibt an, ob das Entfernen ungenutzter Ticket-Attachments und Unit-Inhaltseinträge durchgeführt werden soll. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.9.0.0
cmas-core-server	unused.content.remover.polling.minutes ***nur Version 6.9 und höher***	*Beschreibung:* Gibt an, wie oft überprüft werden soll, ob ungenutzte Ticket-Attachments und Unit-Inhaltseinträge zum Entfernen vorhanden sind. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 15 *Seit:* 6.9.0.0
cmas-core-server	unused.content.remover.ttl.minutes ***nur Version 6.9 und höher***	*Description:* Minimum der Dauer, nach der interval ungenutzte Ticket-Attachments und Unit-Inhaltseinträge entfernt werden können. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1440 *Seit:* 6.9.0.0
cmas-core-shared	cluster.mode	*Beschreibung:* Kennzeichnet, ob CMAS in einem Cluster läuft. *Typ:* Boolean *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.1.0

Modul	System-Property	Erklärung
cmas-core-shared	data.directory	*Beschreibung:* Verzeichnis für CMAS-Daten (z.B. Index). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* C:\Users\user\cmas *Seit:* 6.0
cmas-dwh-server	autocommit.cf.changes	*Beschreibung:* *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.7.0
cmas-dwh-server	batch-commit-interval	*Beschreibung:* Anzahl von Objekten in einer JMS-Nachricht. Höhere Werten bedeuten eine bessere Übertragungsperformance und größeren Speicherverbrauch. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 100 *Seit:* 6.0.0
cmas-dwh-server	dwh.mode	*Beschreibung:* Aktueller Modus der DWH-Datenübermittlung. Mögliche Werte sind OFF, ADMIN, LIVE *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* OFF *Seit:* 6.0.1
cmas-dwh-server	ignore-queues	*Beschreibung:* Durch eine Komma-separierte Liste von Queue-Namen wird hier festgelegt, dass Tickets dieser

Modul	System-Property	Erklärung
		Queues nicht ins DWH übermittelt werden. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* QueueName1, QueueName2,QueueName3 *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	is.cmrf.alive	*Beschreibung:* Als Startpunkt sollte die Zeit genommen werden, bei der zuletzt eine Meldung an CMRF gesendet wu rde . Wen n nach diesem Wert (in Sekunden) keine Antwort vom CMRF empfangen wird, wird ein DWH-Operation-Status mit der Fehlermeldung, dass CMRF nicht erreichbar ist, erstellt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1200 *Seit:* 6.7.0
cmas-dwh-server	java.naming.factory.initial	*Beschreibung:* Factory Java Klasse für DWH context factory. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* org.jnp.interfaces. NamingContextFactory *Seit:* 6.0.1
cmas-dwh-server	java.naming.factory.url.pkgs	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* org.jboss.naming:

Modul	System-Property	Erklärung
		org.jnp.interfaces *Seit:* 6.0.1
cmas-dwh-server	java.naming.provider.url	*Beschreibung:* URL des Naming Providers *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* localhost *Seit:* 6.0.1
cmas-dwh-server	notification.error.description	*Beschreibung:* Text für die E-Mails für Fehlermeldungen des DWHs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Es ist ein Fehler aufgetreten *Seit:* 6.0.1
cmas-dwh-server	notification.error.from	*Beschreibung: From*-Adresse für E-Mails mit Fehlermeldungen des DWHs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1
cmas-dwh-server	notification.error.subject	*Beschreibung:* Betreff für E-Mails mit Fehlermeldungen des DWHs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Es ist ein Fehler aufgetreten *Seit:* 6.0.1
cmas-dwh-server	notification.error.to	*Beschreibung: To*-Adresse für E-Mails mit Fehlermeldungen des DWHs.

Modul	System-Property	Erklärung
		Typ: String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.description	*Beschreibung:* Text für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Übermittlung erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.from	*Beschreibung: From*-Adresse für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.subject	*Beschreibung:* Betreff für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Übermittlung erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.to	*Beschreibung: To*-Adresse für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde.

Modul	System-Property	Erklärung
		Typ: String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully. description	*Beschreibung:* Text für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Übermittlung nicht erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.from	*Beschreibung: From*-Adresse für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.subject	*Beschreibung:* Betreff für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Übermittlung nicht erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.to	*Beschreibung: To*-Adresse für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde.

Modul	System-Property	Erklärung
		Typ: String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0.1
cmas-dwh-server	notification.host	*Beschreibung:* E-Mail (SMTP) Server-Hostname für das Senden von DWH-E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* mail.consol.de *Seit:* 6.1.0
cmas-dwh-server	notification.password	*Beschreibung:* Passwort für das Senden von DWH-E-Mails. (optional) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.1.0
cmas-dwh-server	notification.port	*Beschreibung:* SMTP-Port für das Senden von DWH-E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 25 *Seit:* 6.1.0
cmas-dwh-server	notification.protocol	*Beschreibung:* Das Protokoll, welches für das Senden von E-Mails aus dem DWH heraus verwendet wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* pop3\
cmas-dwh-server	notification.username	

Modul	System-Property	Erklärung
		Beschreibung: (SMTP) Benutzername für das Senden von DWH-E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* maz *Seit:* 6.1.0
cmas-dwh-server	skip-ticket	*Beschreibung:* Tickets werden während Transfer/Update nicht übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	skip-ticket-history	*Beschreibung:* Ticket-Protokoll wird während Transfer/Update nicht übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	skip-unit	*Beschreibung:* Units werden während Transfer/Update nicht übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	skip-unit-history	*Beschreibung:* Unit-Protokoll wird während Transfer/Update nicht

Modul	System-Property	Erklärung
		übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	split.history	*Beschreibung:* Änder t das S QL Statement dahingehend, dass Ticketprotokolle während der DWH-Übermittlung nicht für alle Tickets auf einmal abgeholt werden, sondern ein Ticket pro SQL Statement. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* false *Seit:* 6.8.0
cmas-dwh-server	unit.transfer.order	*Beschreibung:* Legt fest, in welche Reihenfolge Benutzerdefinierte Felder zum DWH übertragen werden. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert: company;customer* *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-esb-core	esb.directory	*Beschreibung:* Verzeichnis, das vom ESB (Mule) verwendet wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* C: \Users\user\cmas\mule *Seit:* 6.0

Modul	System-Property	Erklärung
cmas-esb-mail	mail.attachments.validation.info.sender	*Beschreibung:* Setzt den FROM-Header bei *error notification* Mails, die Attachments betreffen. Standardmäßig wird die E-Mail-Adresse verwendet, die bei der Systeminstallation als Administrator-E--Mail-Adresse angegeben wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* admin@consolcm.com *Seit:* 6.7.5
cmas-esb-mail	mail.attachments.validation.info.subject	*Beschreibung:* Setzt den Betreff bei *error notification* Mails, die Attachments betreffen. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden! *Seit:* 6.7.5
cmas-esb-mail	mail.callname.pattern	*Beschreibung:* Regulärer Ausdruck für den Betreff von eingehenden E-Mails. Verfügbar als TICKET_NAME_PATTERN_FORMAT in Skripten für eingehende E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* .*?Ticket\s+\((\S+)\).* *Seit:* 6.0
cmas-esb-mail	mail.cluster.node.id	

Modul	System-Property	Erklärung
		Beschreibung: Nur der Node, dessen mail.cluster.node.id gleich cmas.clusternode.id ist, startet den Mule ESB Mailservice. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* unspecified *Seit:* 6.6.5
cmas-esb-mail	mail.db.archive	*Beschreibung:* Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* false (Standard) *Seit:* 6.8.5.5
cmas-esb-mail	mail.delete.read	*Beschreibung:* Legt fest, ob CM die per IMAP(S) abgeholten E-Mails löscht. Wenn der Wert auf true gesetzt wird, werden die E-Mails nach der Abholung gelöscht. Standardmäßig werden die per IMAP(S) abgeholten E-Mails nicht gelöscht. Hinweis:E-Mails, die per POP3(S) abgeholt werden, werden immer gelöscht. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.7.3
cmas-esb-mail	mail.encryption	*Beschreibung:* Wenn dieser Wert auf *true* gesetzt ist, ist im Ticket E-Mail Editor die Checkbox zur Verschlüsselung

Modul	System-Property	Erklärung
		der E-Mail standardmäßig aktiviert. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true (Standard = false) *Seit:* 6.8.4.0
cmas-esb-mail	mail.incoming.uri	*Beschreibung:* URL für eingehende E-Mails *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* pop3://cm-incoming-user:password@localhost:10110 *Seit:* 6.0 Dieser Wert sollte mittels der System-Properties nicht verändert werden. Die Posteingänge sollten im Admin-Tool in der Registerkarte *E-Mail* konfiguriert werden (siehe *ConSol*CM Administratorhandbuch* Abschnitt *Registerkarte E-Mail)* . Wenn Sie diese Registerkarte für die Konfiguration benutzen, können Sie alle Einträge konfigurieren, d.h. jedes Postfach, das hinzugefügt wird. CM baut hier während der Einrichtung eines Postfachs eine Test-

Modul	System-Property	Erklärung
		Verbindung auf. Auf diese Weise ist es nicht möglich, falsche Werte einzugeben.
cmas-esb-mail	mail.max.restarts	*Beschreibung:* Maximale Anzahl der Neustarts des Mailservices, bevor aufgegeben wird. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 3 *Seit:* 6.0
cmas-esb-mail	mail.mime.strict	*Beschreibung:* Wenn dieser Wert auf *false* gesetzt wird, werden E-Mail-Adressen nicht auf strikte MIME-Übereinstimmung geparsed. Standard ist *true*, was bedeutet, dass auf strikte MIME-Übereinstimmung geprüft wird. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.17, 6.7.3
cmas-esb-mail	mail.mule.service	*Beschreibung:* FROM-Adresse für E-Mails, die vom Mule-Service aus gesendet werden. *Typ:* EMail *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0
cmas-esb-mail	mail.polling.interval	*Beschreibung:* Abrufintervall für E-Mails in Millisekunden. *Typ:* Integer *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Nein *Beispielwert:* 60000 *Seit:* 6.0
cmas-esb-mail	mail.process.error	*Beschreibung:* TO-Adresse für E-Mails mit Fehlermeldungen von Mule. Standardmäßig wird die E-Mail-Adresse verwendet, die bei der Systeminstallation als Administrator-E--Mail-Adresse angegeben wurde. *Typ:* EMail *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0
cmas-esb-mail	mail.process.retry.attempts	*Beschreibung:* Anzahl der Neuversuche, wenn E-Mails verarbeitet werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 3 *Seit:* 6.0.2
cmas-esb-mail	mail.process.timeout	*Beschreibung:* Timeout für die E-Mail-Verarbeitung in Sekunden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 60 *Seit:* 6.1.3
cmas-esb-mail	mail.redelivery.retry.count	*Beschreibung:* Gibt die Anzahl der Neuversuche an, eine E-Mail aus dem CM-System erneut zuzustellen. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja

Modul	System-Property	Erklärung
		Optional: Nein *Beispielwert:* 3 *Seit:* 6.1.0
cmas-setup-hibernate	hibernate.dialect	*Beschreibung:* Der Hibernate-Dialekt. Normalerweise wird dieser Wert während des initialen Setups gesetzt (abhängig vom Datenbank-System). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* org.hibernate.dialect.MySQL5InnoDBDialect *Seit:* 6.0
cmas-setup-manager	initialized	*Beschreibung:* Kennzeichnet, ob CMAS initialisiert ist. Wenn dieser Wert fehlt oder nicht auf *true* gesetzt ist, wird das Setup ausgeführt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.0 Seien Sie mit der Verwendung dieser System-Property sehr vorsichtig! Wenn Sie den Wert auf false setzen, wird der ConSol*CM-Sevrer beim nächsten Systemstart das System-Setup ausführen, d.h. alle Daten des bestehenden Systems werden verloren gehen,

Modul	System-Property	Erklärung
		inklusive der System-Properties!
cmas-setup-scene	scene	*Beschreibung:* Szenario-Datei, die während des Setups importiert wurde (kann leer gelassen werden). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/ *Seit:* 6.0
cmas-workflow-engine	jobExecutor.adminMail	*Beschreibung:* E-Mail-Adresse, an die Benachrichtigungs-E-Mails, die Probleme der Job Execution betreffen (wenn die Anzahl der Neuversuche überschritten wurde), geschickt werden. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* admin@consol.de *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.idleInterval.seconds	*Beschreibung:* Legt fest, wie oft de r Job Executer Thread nach n euen Jobs zum Ausführen sucht. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 5 (Standard) *Seit:* 6.8.0

Modul	System-Property	Erklärung
cmas-workflow-engine	jobExecutor.jobMaxRetries	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 5 (default) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor. jobMaxRetriesReachedSubject	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* Job max retries reached. Job was removed!!! (default) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.lockTimeout. seconds	*Beschreibung:* Legt fest, wie lange ein Job vo m Job Executor gelockt (als "in der Ausführung" markiert) werden kann. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 360 (Standard) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.lockingLimit	*Beschreibung:* Anzahl der gleichzeitig gelockten (als "in der Ausführung" markierten) Jobs des Job Executor Thread *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10 (Standard) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.mailFrom	*Beschreibung:* E-Mail-Adresse, die als FROM-Header für Admin-Benachrichtigungen eingesetzt wird. *Typ:* String

Modul	System-Property	Erklärung
		Neustart erforderlich: Nein *System*: Ja *Optional*: Ja *Beispielwert:* jobexecutor@consol.de *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.maxInactivityInterval.minutes	*Beschreibung:* Länge der erlaubten Inaktivität des Job Executors in Minuten (z.B. wenn er durch eine Langzeit-Ausführung gesperrt wird). Nach dieser Zeit werden die Executer-Threads neu gestartet. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja. Standardwert ist auf 30 Minuten gesetzt *Beispielwert:* 15 (Standard) *Seit:* 6.9.2.0
cmas-workflow-engine	jobExecutor.threads	*Beschreibung:* Maximale Anzahl der Job-Execution-Threads. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 1 (Standard) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.timerRetryInterval.seconds	*Beschreibung:* Legt fest, wie lange der Job Executer Thread nach einem Fehler bei der Job-Ausführung wartet. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10 (Standard) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.txTimeout.seconds	*Beschreibung:* Übermittlungs-Time-Out für die Job Execution. *Typ:* Integer

Modul	System-Property	Erklärung
		Neustart erforderlich: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 60 (Standard) *Seit:* 6.8.0
cmweb-server-adapter	checkUserOnlineIntervalInSeconds	*Beschreibung:* Das Intervall (in Sekunden), in dem geprüft wird, welche Benutzer online sind (Standard 180 Sekunden = 3 Minuten). *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 180 *Seit:* 6.0
cmweb-server-adapter	cmoffice.enabled	*Beschreibung:* Kennzeichnet, ob CM/Office aktiviert ist. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.4.0
cmweb-server-adapter	commentRequiredForTicketCreation	*Beschreibung:* Kennzeichnet, ob ein Kommentar für die Erstellung eines Tickets notwendig ist. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true (Standard) *Seit:* 6.2.0
cmweb-server-adapter	customizationVersion	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* cd58453e-f3cc-4538-8030-d15e8796a4a7 *Seit:* 6.5.0

Modul	System-Property	Erklärung
cmweb-server-adapter	data.optimization	*Beschreibung:* Definiert die Optimierung der Response-Daten. Bis zu der Version, die in diesem *ConSol*CM Administrator handbuch* behandelt wird, werden die folgenden Werte unterstützt (um mehr als einen Wert zu setzen, trennen Sie die Werte durch '|'): MINIFICATION und COMPRESSION. MINIFICATION minimiert HTML-Daten, indem z.B. Kommentare und Leerräume entfernt werden. COMPRESSION wendet gzip-Komprimierung auf HTTP-Response. (Hinweis: Wenn das System im Cluster-Modus läuft und Sie parallel verschiede Konfigurationen testen möchten, können Sie für jeden Cluster-Node verschiedene Werte setzen, indem Sie die System-Property nach dem Muster data.optimization.*nodeId* spezifizieren, um die Standard System-Property zu überschreiben. *Typ:* String *Neustart erforderlich:* COMPRESSION kann ohne Neustart an- und ausgeschaltet werden, MINIFICATION erfordert einen Neustart. *System*: Ja *Optional*: Ja *Beispielwert:* MINIFICATION|COMPRESSION
cmweb-server-adapter	defaultContentEntryClassName	*Beschreibung:* Standard-Textklasse für neue ACIMs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein

Modul	System-Property	Erklärung
		Beispielwert: default_class *Seit:* 6.3.0
cmweb-server-adapter	defaultNumberOfCustomFieldsColumns	*Beschreibung:* Standard-Anzahl von Spalten für Benutzerdefinierte Felder. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 3 *Seit:* 6.2.0
cmweb-server-adapter	favoritesSizeLimit	*Beschreibung:* Maximale Anzahl von Favoriten in der Favoriten-Liste. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 10 *Seit:* 6.0
cmweb-server-adapter	globalSearchResultSizeLimit	*Beschreibung:* Maximale Anzahl von Suchergebnissen in der Schnellsuche ("Quick and Easy Search"). *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 10 *Seit:* 6.0
cmweb-server-adapter	helpFilePath	*Beschreibung:* URL für die Online-Hilfe. Wenn der Wert nicht leer gelassen wird, wird der *Hilfe* Button im Web Client angezeigt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja

Modul	System-Property	Erklärung
		Beispielwert: http://www.consol.de *Seit:* 6.2.1
cmweb-server-adapter	hideTicketSubject	*Beschreibung:* Wenn der Wert auf *true* gesetzt ist, ist das Thema des Tickets unsichtbar. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.2.1
cmweb-server-adapter	mail.from	*Beschreibung:* Wenn diese E-Mail-Adresse gesetzt wird, wird diese E-Mail-Adresse anstelle der E-Mail-Adresse des Bearbeiters in E-Mail-Konversationen verwendet. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.1.2
cmweb-server-adapter	mail.reply.to	*Beschreibung:* Wenn dieser Wert gesetzt wird, zeigt der Web Client die gesetzte E-Mail-Adresse im Ticket-E-Mail-Editor im REPLY-TO-Feld an, und die E-Mail wird entsprechend verschickt. Im E-Mail-Client wird somit diese REPLY-TO Adresse ebenfalls angeboten. Typ: String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1 Bitte lesen Sie dazu auch im *ConSol*CM Administratorhandbuch*

Modul	System-Property	Erklärung
		den Abschnitt *Queue-Verwaltung*. Wenn Sie die *Reply-to*-Adresse in einem Skript für ausgehende E-Mails festlegen, darf die System-Property *mail.reply.to* nicht gesetzt werden, da sie den im Skript konfigurierten Wert überschreiben würde! Das bedeutet, dass wenn Sie ein Skript für ausgehende E-Mails für **eine** Queue verwenden, Sie für **alle** Queues ein Skript für ausgehende E-Mails definieren müssen, da die System-Property *mail.reply.to* nicht länger verwendet werden kann.
cmweb-server-adapter	mailTemplateAboveQuotedText	*Beschreibung:* Gibt das Verhalten der Templates im Ticket E-Mail Editor an, wenn eine andere E-Mail zitiert wird, d. h. auf diese geantwortet oder diese weitergeleitet wird. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.2.4
cmweb-server-adapter	maxSizePerPagemapInMegaBytes	*Beschreibung:* Maximale Größe (in MB) für jede Wicket pagemap *Typ:* Integer *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Nein *Beispielwert:* 15 *Seit:* 6.3.5
cmweb-server-adapter	pagemapLockDurationInSeconds	*Beschreibung:* Anzahl der Sekunden, die vergehen müssen, bevor eine Pagemap als z u lange gelockt angesehen wird. *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* 60 *Seit:* 6.7.3
cmweb-server-adapter	postActivityExecutionScriptName	*Beschreibung:* Definiert den Namen des Skripts, das nach jeder Workflow-Aktivität ausgeführt wird (siehe *ConSol*CM Administratorhandbuch*, Abschnitt *Standard-Skript für Workflow-Aktivitäten*). Wenn kein Skript ausgeführt werden soll, lassen Sie diesen Wert leer. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* postActivityExecutionHandler *Seit:* 6.2.0
cmweb-server-adapter	queuesExcludedFromGS	*Beschreibung:* Komma-separierte Liste von Queue-Namen, die von der Schnellsuche ausgeschlossen werden sollen. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0

Modul	System-Property	Erklärung
cmweb-server-adapter	rememberMeLifetimeInMinutes	*Beschreibung:* Lebensdauer für *remember me* in Minuten. *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 1440 *Seit:* 6.0
cmweb-server-adapter	request.scope.transaction	*Beschreibung:* Mit *true* oder *false* kann hier das Verhalten bzgl. Datenbank-Transaktionen bei HTTP-Requests eingestellt werden. Steht der Wert auf *true* (Standard), werden alle Service-Methoden, die ausgeführt weden müssen, in einer Datenbanktransaktion abgewickelt. Steht der Wert auf *false*, wird für jede Service-Methode eine separate Transaktion verwendet. *Typ:* Boolean *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* true *Seit:* 6.8.1
cmweb-server-adapter	searchPageSize	*Beschreibung:* Standardgröße der Seiten für Suchergebnisse. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 20 *Seit:* 6.0
cmweb-server-adapter	searchPageSizeOptions	*Beschreibung:* Optionen für Seitengröße für Suchergebnisse. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:*

Modul	System-Property	Erklärung
		10|20|30|40|50|75|100 *Seit:* 6.0
cmweb-server-adapter	serverPoolingInterval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Seit:* 6.1.0
cmweb-server-adapter	supportEmail	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0
cmweb-server-adapter	themeOverlay	*Beschreibung:* Name des verwendeten Themen-Overlays. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* kyoEUR *Seit:* 6.0
cmweb-server-adapter	ticketListRefreshIntervalInSeconds	*Beschreibung:* Aktualisierungsintervall für die Ticketliste (in Sekunden) *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 180 *Seit:* 6.0
cmweb-server-adapter	ticketListSizeLimit	*Beschreibung:* Maximale Anzahl von Tickets in der Ticketliste. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 100 *Seit:* 6.0

Modul	System-Property	Erklärung
cmweb-server-adapter	unitIndexSearchResultSizeLimit	*Description:* Maximum number of units in unit search result (e.g. when searching for contact) *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.0
cmweb-server-adapter	urlLogoutPath	*Beschreibung:* URL die verwendet wird, wenn sich der Bearbeiter ausloggt (wenn keine Werte gesetzt werden, wird nach dem Logout die Login-Seite angezeigt). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* http://intranet.consol.de *Seit:* 6.3.1
cmweb-server-adapter	webSessionTimeoutInMinutes	*Beschreibung:* Session-Timeout in Minuten *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 180 *Entfernt seit:* 6.7.1 *Ersetzt durch:* server.session.timeout
cmweb-server-adapter	wicketAjaxRequestHeaderFilterEnabled	*Beschreibung:* Dies aktiviert Filter für Wicket AJAX-Anfragen. *Typ:* Boolean *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* false *Seit:* 6.8.1
cmas-workflow-jbpm	fetchLock.interval	

Modul	System-Property	Erklärung
		Beschreibung: *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5000 *Entfernt seit:* 6.8.0
cmas-workflow-jbpm	fetchLock.timeout	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 15000 *Entfernt seit:* 6.8.0
cmas-workflow-jbpm	jobExecutor.idleInterval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 45000 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.idleInterval.seconds
cmas-workflow-jbpm	jobExecutor.jobExecuteRetryNumber	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.jobMaxRetries
cmas-workflow-jbpm	jobExecutor.timerRetryInterval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 10000 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.timerRetryInterval.seconds

Modul	System-Property	Erklärung
cmas-workflow-jbpm	mail.sender.address	*Beschreibung: From*-Adresse für E-Mails, die von der Workflow-Engine heraus versendet werden. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.mailFrom
cmas-workflow-jbpm	outdated.lock.age	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 60000 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.lockTimeout.seconds
cmas-workflow-jbpm	refreshTimeInCaseOfConcurrentRememberMeRequests	*Beschreibung:* Legt die Aktualisierungszeit (in Sekunden) fest, nach der die Seite im Falle von gleichzeitigen *remember me* Anfragen neu geladen wird. Dieses Feature verhindert, dass ein Benutzer zu viele Lizenzen in Anspruch nimmt. *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* 5 *Seit:* 6.8.2

11.2 System-Properties sortiert nach Name der System-Property

Modul	System-Property	Erklärung
cmas-core-security	admin.email	*Beschreibung:* Die E-Mail-Adresse des ConSol*CM Administrators. Anfänglich wird hier der Wert genommen, den Sie beim System-Setup eingegeben haben. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0
cmas-core-security	admin.login	*Beschreibung:* Der (Login-) Name des ConSol*CM Administrators . Anfänglich wird hier der Wert genommen, den Sie beim System-Setup eingegeben haben. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* admin *Seit:* 6.0
cmas-app-admin-tool	admin.tool.session.check.interval	*Beschreibung:* Intervall, in dem inaktive (beendete) Sitzungen im Admin-Tool überprüft werden (in Sekunden) *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 30 *Seit:* 6.7.5
cmas-core-server	attachment.allowed.types	*Beschreibung:* Komma-separierte Liste der erlaubten

Modul	System-Property	Erklärung
		Dateinamen-Erweiterungen (wenn keine Werte definiert werden, sind alle Dateinamen-Erweiterungen erlaubt). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* txt,zip,doc *Seit:* 6.5.0
cmas-core-server	attachment.max.size	*Beschreibung:* Maximal Größe von Attachments in MB. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 100 *Seit:* 6.4.0
cmas-core-security	authentication.method	*Beschreibung:* Methode der Engineer (Bearbeiter)-Authentifizierung für den Web Client (interne CM-Datenbank oder LDAP-Authentifizierung). Erlaubte Werte sind *LDAP* oder *DATABASE*. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* DATABASE *Seit:* 6.0
cmas-dwh-server	autocommit.cf.changes	*Beschreibung:* *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.7.0
cmas-app-admin-tool	autocomplete.enabled ***nur Version 6.9 und höher***	*Beschreibung:* Wenn diese Property fehlt oder der Wert *false* ist, wird der Tab *Adresse*

Modul	System-Property	Erklärung
		Autocomplete im Admin-Tool ausgeblendet. *Typ:* Boolean *Neustart erforderlich:* Nein *System:* Ja *Optional:* Ja *Beispielwert:* true *Seit:* 6.9.2.0
cmas-dwh-server	batch-commit-interval	*Beschreibung:* Anzahl von Objekten in einer JMS-Nachricht. Höhere Werten bedeuten eine bessere Übertragungsperformance und größeren Speicherverbrauch. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 100 *Seit:* 6.0.0
cmas-core-index-common	big.task.minimum.size	*Beschreibung:* Gibt an, wie viele Teile ein Task mindestens haben soll, damit er vom Indexer mit Priorität *low* behandelt wird. *Typ:* Integer *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Beispielwert:* 15 (Standard) *Seit:* 6.8.3
cmas-core-cache	cache-cluster-name	*Beschreibung:* Cache-Cluster-Name des JBoss *Typ:* String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 635a6de1-629a-4129-8299-2d98633310f0 *Seit:* 6.4.0
cmweb-server-adapter	checkUserOnlineIntervalInSeconds	*Beschreibung:* Das Intervall (in Sekunden), in dem geprüft wird,

Modul	System-Property	Erklärung
		welche Benutzer online sind (Standard 180 Sekunden = 3 Minuten). *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 180 *Seit:* 6.0
cmas-core-shared	cluster.mode	*Beschreibung:* Kennzeichnet, ob CMAS in einem Cluster läuft. *Typ:* Boolean *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.1.0
cmweb-server-adapter	cmoffice.enabled	*Beschreibung:* Kennzeichnet, ob CM/Office aktiviert ist. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.4.0
cmweb-server-adapter	commentRequiredForTicketCreation	*Beschreibung:* Kennzeichnet, ob ein Kommentar für die Erstellung eines Tickets notwendig ist. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true (Standard) *Seit:* 6.2.0
cmas-core-server	config.data.version	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 11 *Seit:* 6.0

Modul	System-Property	Erklärung
cmas-core-security	contact.authentication.method ***nur Version 6.9 und höher***	*Beschreibung:* Definiert die Kontakt-Authentifizierungsmethode für CM/Track, mögliche Werte sind *DATABASE* oder *LDAP* oder *LDAP,DATABASE* oder *DATABASE,LDAP.* *Typ:* String *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Seit:* 6.9.3.0
cmas-core-security	contact.inherit.permissions.only.to.own.customer.group ***nur Version 6.9 und höher***	*Beschreibung:* Zeigt an, ob der authentifizierte Kontakt in CM/Track alle Kundengruppen-Berechtigungen vom repräsentierenden Bearbeiter erbt (*false*), oder nur die Berechtigungen der eignen Kundengruppe erbt (*true*). *Typ:* Boolean *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Seit:* 6.9.2.3
cmweb-server-adapter	customizationVersion	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* cd58453e-f3cc-4538-8030-d15e8796a4a7 *Seit:* 6.5.0
cmas-core-shared	data.directory	*Beschreibung:* Verzeichnis für CMAS-Daten (z.B. Index). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* C:\Users\user\cmas *Seit:* 6.0

Modul	System-Property	Erklärung
cmweb-server-adapter	data.optimization	*Beschreibung:* Definiert die Optimierung der Response-Daten. Bis zu der Version, die in diesem *ConSol*CM Administrator handbuch* behandelt wird, werden die folgenden Werte unterstützt (um mehr als einen Wert zu setzen, trennen Sie die Werte durch '|'): MINIFICATION und COMPRESSION. MINIFICATION minimiert HTML-Daten, indem z.B. Kommentare und Leerräume entfernt werden. COMPRESSION wendet gzip-Komprimierung auf HTTP-Response. (Hinweis: Wenn das System im Cluster-Modus läuft und Sie parallel verschiede Konfigurationen testen möchten, können Sie für jeden Cluster-Node verschiedene Werte setzen, indem Sie die System-Property nach dem Muster data.optimization.*nodeId* spezifizieren, um die Standard System-Property zu überschreiben. *Typ:* String *Neustart erforderlich:* COMPRESSION kann ohne Neustart an- und ausgeschaltet werden, MINIFICATION erfordert einen Neustart. *System*: Ja *Optional*: Ja *Beispielwert:* MINIFICATION|COMPRESSION
cmas-core-server	defaultCommentClassName	*Beschreibung:* Standard-Textklasse für Kommentare. *Typ:* String *Neustart erforderlich*: Nein *System*: Nein *Optional*: Ja

Modul	System-Property	Erklärung
		Beispielwert: *Seit:* 6.3.0
cmweb-server-adapter	defaultContentEntryClassName	*Beschreibung:* Standard-Textklasse für neue ACIMs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* default_class *Seit:* 6.3.0
cmas-core-server	defaultIncommingMailClassName	*Beschreibung:* Standard-Textklasse für eingehende E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Nein *Optional*: Ja *Beispielwert:* *Seit:* 6.3.0
cmweb-server-adapter	defaultNumberOfCustomFieldsColumns	*Beschreibung:* Standard-Anzahl von Spalten für Benutzerdefinierte Felder. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 3 *Seit:* 6.2.0
cmas-core-server	defaultOutgoingMailClassName	*Beschreibung:* Standard-Textklasse für ausgehende E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Nein *Optional*: Ja *Beispielwert:* *Seit:* 6.3.0
cmas-core-index-common	disable.admin.task.auto.commit	*Beschreibung:* Alle Tasks, die für ein Index-Update erstellt werden, werden automatisch direkt nach ihrer Erstellung

Modul	System-Property	Erklärung
		ausgeführt *Typ:* Boolean *Neustart erforderlich:* Nein *System:* Ja *Optional:* Nein *Beispielwert:* false *Seit:* 6.6.1
cmas-dwh-server	dwh.mode	*Beschreibung:* Aktueller Modus der DWH-Datenübermittlung. Mögliche Werte sind OFF, ADMIN, LIVE *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* OFF *Seit:* 6.0.1
cmas-esb-core	esb.directory	*Beschreibung:* Verzeichnis, das vom ESB (Mule) verwendet wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* C:\Users\user\cmas\mule *Seit:* 6.0
cmas-core-cache	eviction.event.queue.size	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 200000 *Seit:* 6.4.0
cmas-core-cache	eviction.max.nodes	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 100000 *Seit:* 6.4.0
cmas-core-cache	eviction.wakeup.interval	

Modul	System-Property	Erklärung
		Beschreibung: *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 3000 *Seit:* 6.4.0
cmweb-server-adapter	favoritesSizeLimit	*Beschreibung:* Maximale Anzahl von Favoriten in der Favoriten-Liste. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 10 *Seit:* 6.0
cmas-workflow-jbpm	fetchLock.interval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5000 *Entfernt seit:* 6.8.0
cmas-workflow-jbpm	fetchLock.timeout	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 15000 *Entfernt seit:* 6.8.0
cmas-core-server	fetchSize.strategy	*Beschreibung:* Auswahl der Strategie, für JDBC Ergebnis-Sets *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* FetchSizePageBasedStrategy,

Modul	System-Property	Erklärung
		FetchSizeThresholdStrategy, FetchSizeFixedStrategy *Seit:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizeFixedStrategy.value	*Beschreibung:* Gibt den Wert für Abhol-Größen an, wenn die ausgewählte Strategie für die Abholung der Größe FetchSizeFixedStrategy ist. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 150 *Seit:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizePageBasedStrategy. limit	*Beschreibung:* Gibt den Wert für Abhol-Größen an, wenn die ausgewählte Strategie für die Abholung der Größe FetchSizePageBasedStrategy ist. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10000 *Seit:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizeThresholdStrategy. value	*Beschreibung:* Gibt den Grenzwert für Abhol-Größen an , wenn die ausgewählte Strategie für die Abholung der Größe Fetch SizeThresholdStrategy ist. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 150,300,600,1000 *Seit:* 6.8.4.1
cmweb-server-adapter	globalSearchResultSizeLimit	*Beschreibung:* Maximale Anzahl von Suchergebnissen in der Schnellsuche ("Quick and Easy Search").

Modul	System-Property	Erklärung
		Typ: Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 10 *Seit:* 6.0
cmweb-server-adapter	helpFilePath	*Beschreibung:* URL für die Online-Hilfe. Wenn der Wert nicht leer gelassen wird, wird der *Hilfe* Button im Web Client angezeigt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* http://www.consol.de *Seit:* 6.2.1
cmas-setup-hibernate	hibernate.dialect	*Beschreibung:* Der Hibernate-Dialekt. Normalerweise wird dieser Wert während des initialen Setups gesetzt (abhängig vom Datenbank-System). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* org.hibernate.dialect.MySQL5InnoDBDialect *Seit:* 6.0
cmweb-server-adapter	hideTicketSubject	*Beschreibung:* Wenn der Wert auf *true* gesetzt ist, ist das Thema des Tickets unsichtbar. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.2.1
cmas-dwh-server	ignore-queues	

Modul	System-Property	Erklärung
		Beschreibung: Durch eine Komma-separierte Liste von Queue-Namen wird hier festgelegt, dass Tickets dieser Queues nicht ins DWH übermittelt werden. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* QueueName1, QueueName2,QueueName3 *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-core-index-common	index.attachment	*Beschreibung:* Beschreibt, ob der Inhalt von Attachments indziert wird. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.4.3
cmas-core-index-common	index.history	*Beschreibung:* Beschreibt, ob die Unit und das Ticket-Protokoll indiziert werden *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.1.0
cmas-core-index-common	index.status	*Beschreibung:* Status des Indexers, mögliche Werte sind RED, YELLOW, GREEN, werden im Admin-Tool angezeigt *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* GREEN *Seit:* 6.6.1

Modul	System-Property	Erklärung
cmas-core-index-common	index.task.worker.threads	*Beschreibung:* Beschreibt, wie viele Threads benutzt werden, um Batch-Index-Aufgaben auszuführen (Synchronisierung, Administrations- und Reparatur-Aufgaben). *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1 (Standard) (Wir empfehlen, einen Wert zu benutzen, der nicht größer als zwei ist) *Seit:* 6.6.14, 6.7.3
cmas-core-index-common	index.version.current	*Beschreibung:* Enthält Informationen über die derzeitige (möglicherweise veraltete) Index-Version. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1 (Standard) *Seit:* 6.7.0
cmas-core-index-common	index.version.newest	*Beschreibung:* Enthält Informationen, welche Index-Version als die neueste betrachtet wird. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1 (Standard) *Seit:* 6.7.0
cmas-core-index-common	indexed.assets.per.thread.in.memory	*Beschreibung:* Beschreibt, wie viele Assets während des Indizierens pro Thread auf einmal in den Speicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Nein *Beispielwert:* 200 (Standard) *Seit:* 6.8.0
cmas-core-index-common	indexed.engineers.per.thread.in.memory	*Beschreibung:* Beschreibt, wie viele Bearbeiter während des Indizieren s pro Thread auf einma l in den Speicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 300 (Standard) *Seit:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.tickets.per.thread.in.memory	*Beschreibung:* Beschreibt, wie viele Tickets während des Indizierens pro Thread auf einm al in den Speicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 100 (Standard) *Seit:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.units.per.thread.in.memory	*Beschreibung:* Beschreibt, wie viele Units während des Indizierens pro Thread auf ein m al in den Speicher geladen werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 200 (Standard) *Seit:* 6.6.14, 6.7.3
cmas-setup-manager	initialized	*Beschreibung:* Kennzeichnet, ob CMAS initialisiert ist. Wenn dieser Wert fehlt oder nicht auf *tr ue* gesetzt ist, wird das Setup

Modul	System-Property	Erklärung
		ausgeführt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.0 Seien Sie mit der Verwendung dieser System-Property sehr vorsichtig! Wenn Sie den Wert auf false setzen, wird der ConSol*CM-Sevrer beim nächsten Systemstart das System-Setup ausführen, d.h. alle Daten des bestehenden Systems werden verloren gehen, inklusive der System-Properties!
cmas-dwh-server	is.cmrf.alive	*Beschreibung:* Als Startpunkt sollte die Zeit genommen werden, bei der zuletzt eine Meldung an CMRF gesendet wu rde . Wen n nach diesem Wert (in Sekunden) keine Antwort vom CMRF empfangen wird, wird ein DWH-Operation-Status mit der Fehlermeldung, dass CMRF nicht erreichbar ist, erstellt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1200 *Seit:* 6.7.0

Modul	System-Property	Erklärung
cmas-dwh-server	java.naming.factory.initial	*Beschreibung:* Factory Java Klasse für DWH context factory. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* org.jnp.interfaces.NamingContextFactory *Seit:* 6.0.1
cmas-dwh-server	java.naming.factory.url.pkgs	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* org.jboss.naming:org.jnp.interfaces *Seit:* 6.0.1
cmas-dwh-server	java.naming.provider.url	*Beschreibung:* URL des Naming Providers *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* localhost *Seit:* 6.0.1
cmas-workflow-engine	jobExecutor.adminMail	*Beschreibung:* E-Mail-Adresse, an die Benachrichtigungs-E-Mails, die Probleme der Job Execution betreffen (wenn die Anzahl der Neuversuche überschritten wurde), geschickt werden. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* admin@consol.de *Seit:* 6.8.0
cmas-workflow-jbpm	jobExecutor.idleInterval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Nein *Beispielwert:* 45000 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.idleInterval.seconds
cmas-workflow-engine	jobExecutor.idleInterval.seconds	*Beschreibung:* Legt fest, wie oft de r Job Executer Thread nach n euen Jobs zum Ausführen sucht. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 5 (Standard) *Seit:* 6.8.0
cmas-workflow-jbpm	jobExecutor.jobExecuteRetryNumber	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.jobMaxRetries
cmas-workflow-engine	jobExecutor.jobMaxRetries	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 5 (default) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.jobMaxRetriesReachedSubject	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* Job max retries reached. Job was removed!!! (default) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.lockingLimit	

Modul	System-Property	Erklärung
		Beschreibung: Anzahl der gleichzeitig gelockten (als "in der Ausführung" markierten) Jobs des Job Executor Thread *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10 (Standard) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.lockTimeout.seconds	*Beschreibung:* Legt fest, wie lange ein Job vo m Job Executor gelockt (als "in der Ausführung" markiert) werden kann. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 360 (Standard) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.mailFrom	*Beschreibung:* E-Mail-Adresse, die als FROM-Header für Admin-Benachrichtigungen eingesetzt wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* jobexecutor@consol.de *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.maxInactivityInterval.minutes	*Beschreibung:* Länge der erlaubten Inaktivität des Job Executors in Minuten (z.B. wenn er durch eine Langzeit-Ausführung gesperrt wird). Nach dieser Zeit werden die Executer-Threads neu gestartet. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja. Standardwert ist

Modul	System-Property	Erklärung
		auf 30 Minuten gesetzt *Beispielwert:* 15 (Standard) *Seit:* 6.9.2.0
cmas-workflow-engine	jobExecutor.threads	*Beschreibung:* Maximale Anzahl der Job-Execution-Threads. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 1 (Standard) *Seit:* 6.8.0
cmas-workflow-jbpm	jobExecutor.timerRetryInterval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 10000 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.timerRetryInterval.seconds
cmas-workflow-engine	jobExecutor.timerRetryInterval.seconds	*Beschreibung:* Legt fest, wie lange der Job Executer Thread nach einem Fehler bei der Job-Ausführung wartet. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10 (Standard) *Seit:* 6.8.0
cmas-workflow-engine	jobExecutor.txTimeout.seconds	*Beschreibung:* Übermittlungs-Time-Out für die Job Execution. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 60 (Standard) *Seit:* 6.8.0
cmas-core-security	kerberos.v5.enabled	*Beschreibung:* Kennzeichnung, welche anzeigt, ob SSO mit Kerberos aktiviert ist.

Modul	System-Property	Erklärung
		Typ: Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false (Standard, wenn Kerberos während des System Setups nicht aktiviert wurde) *Seit:* 6.2.0
cmas-core-security	kerberos.v5.username.regex	*Beschreibung:* Regulärer Ausdruck, der für die Zuweisung des Kerberos Principals zum CM-Bearbeiter-Login zuständig ist. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* (.*)@.* *Seit:* 6.2.0
cmas-core-server	last.config.change	*Beschreibung:* Zufällige UUID, die während der letzten Veränderung der Konfiguration der Benutzerdefinierten Felder (via Admin-Tool) generiert wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 2573c7b7-2bf5-47ff-b5a2-bad31951a266 *Seit:* 6.1.0, 6.2.1
cmas-core-security	ldap.authentication	*Beschreibung:* Authentifizierungsmethode für den Web Client, die bei der LDAP-Authentifizierung benutzt wird. *Typ:* String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* simple *Seit:* 6.0

Modul	System-Property	Erklärung
cmas-core-security	ldap.basedn	*Beschrei bung:* Base DN fü r die Suche von LDAP-Benutzer-Accounts (LDAP Authentifizierung im Web Client), wenn LDAP-Authentifizierung verwendet wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* OU=accounts, DC=consol,DC=de *Seit:* 6.0
cmas-core-server	ldap.certificate.basedn	*Beschreibu ng:* Base DN für den Ort der Zertifikate im LDAP-Verzeichnisbaum. Wen n nichts angegeben wird, wird ldap. basedn verwendet. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* OU=accounts, DC=consol,DC=de *Seit:* 6.8.4
cmas-core-server	ldap.certificate.content.attribute	*Beschreibung:* LDAP-Attribut-Name der angibt, wo Zertifikatsdaten im LDAP-Verzeichnisbaum gespeichert werden. Standardwert ist: usercertificate (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* usercertificate *Seit:* 6.8.4
cmas-core-server	ldap.certificate.password	*Beschreibung:* Passwort des LDAP-Zertifikate-Managers. Wenn nichts gesetzt wird, wird

Modul	System-Property	Erklärung
		ldap.password genommen. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.8.4
cmas-core-server	ldap.certificate.providerurl	*Beschreibung:* URL des LDAP-Zertifikate-Providers. Wenn nichts gesetzt wird, wird ldap.providerurl genommen. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* ldap://ldap.consol.de:389 *Seit:* 6.8.4
cmas-core-server	ldap.certificate.searchattr	*Beschreibung:* LDAP-Attribut-Name, der für die Suche nach Zertifikaten im LDAP-Verzeichnisbaum verwendet wird. Standardwert ist: mail. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* mail *Seit:* 6.8.4
cmas-core-server	ldap.certificate.userdn	*Beschreibung:* DN des LDAP-Zertifikate-Managers. Wenn nichts gesetzt wird, wird ldap.userdn genommen. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Ja *Seit:* 6.8.4
cmas-core-security	ldap.contact.name.basedn ***nur Version 6.9 und höher***	*Beschreibung:* BaseDN für die Suche n ach Kontakt-DN mitt els LDAP-ID (z.B. ou=accounts, dc=consol,dc=de) (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.password ***nur Version 6.9 und höher***	*Beschreibung:* Passwort für die Suche nach Kontakt-DN mittels L DAP-ID. Wenn nicht gesetzt, wird ein anonymer Account verwendet. (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.providerurl ***nur Version 6.9 und höher***	*Beschreibung:* Adresse des LDAP-Servers (ldap[s]://host: port). (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.searchattr ***nur Version 6.9 und höher***	*Beschreibung:* Attribut für die Suc he nach Kontakt-DN mittels LDAP-ID (z.B. uid). (LDAP Authentifizierung in CM/Track) *Typ:* String *Neustart erforderlich:* Nein

Modul	System-Property	Erklärung
		System: Nein *Optional:* Ja *Seit:* 6.9.3.0
cmas-core-security	ldap.contact.name.userdn ***nur Version 6.9 und höher***	*Beschreibung:* Benutzer-DN für die Suche nach Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymer Account verwendet. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich:* Nein *System:* Nein *Optional:* Ja *Since:* 6.9.3.0
cmas-core-security	ldap.initialcontextfactory	*Beschreibung:* Name der Klasse für initial context factory der LDAP-Implementierung, wenn LDAP-Authentifizierung verwendet wird. Ist üblicherweise *com.sun.jndi.ldap.LdapCtxFactory* (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* com.sun.jndi.ldap.LdapCtxFactory *Seit:* 6.0
cmas-core-security	ldap.password	*Beschreibung:* Passwort für die Verbindung zum LDAP, um Benutzer zu suchen (wenn LDAP-Authentifizierung verwendet wird). Wird nur benötigt, wenn die Suche nicht anonym durchgeführt werden kann. (LDAP Authentifizierung im Web Client) *Typ:* Password *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Ja *Seit:* 6.1.2
cmas-core-security	ldap.providerurl	*Beschreibung:* LDAP-Provider (wenn LDAP-Authentifizierung im Web Client verwendet wird). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* ldap://ldap.consol.de:389 *Seit:* 6.0
cmas-core-security	ldap.searchattr	*Beschreibung:* Such-Attribute für die Suche nach LDAP-Einträgen, die mit dem CM6-Login verbunden sind. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* uid *Seit:* 6.0
cmas-core-security	ldap.userdn	*Beschreibung:* LDAP-Benutzer für die Verbindung zum LDAP, um Benutzer zu suchen (wenn LDAP-Authentifizierung verwendet wird). Wird nur benötigt, wenn die Suche nicht anonym durchgeführt werden kann. (LDAP Authentifizierung im Web Client) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.1.2
cmas-esb-mail	mail.attachments.validation.info.sender	*Beschreibung:* Setzt den FROM-Header bei *error notification* Mails, die Attachments betreffen.

Modul	System-Property	Erklärung
		Standardmäßig wird die E-Mail-Adresse verwendet, die bei der Systeminstallation als Administrator-E--Mail-Adresse angegeben wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* admin@consolcm.com *Seit:* 6.7.5
cmas-esb-mail	mail.attachments.validation.info.subject	*Beschreibung:* Setzt den Betreff bei *error notification* Mails, die Attachments betreffen. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden! *Seit:* 6.7.5
cmas-esb-mail	mail.callname.pattern	*Beschreibung:* Regulärer Ausdruck für den Betreff von eingehenden E-Mails. Verfügbar als TICKET_NAME_PATTERN_FORMAT in Skripten für eingehende E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* .*?Ticket\s+\((\S+)\).* *Seit:* 6.0
cmas-esb-mail	mail.cluster.node.id	*Beschreibung:* Nur der Node, dessen mail.cluster.node.id gleich cmas.clusternode.id ist, startet den Mule ESB

Modul	System-Property	Erklärung
		Mailservice. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* unspecified *Seit:* 6.6.5
cmas-esb-mail	mail.db.archive	*Beschreibung:* Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* false (Standard) *Seit:* 6.8.5.5
cmas-esb-mail	mail.delete.read	*Beschreibung:* Legt fest, ob CM die per IMAP(S) abgeholten E-Mails löscht. Wenn der Wert auf true gesetzt wird, werden die E-Mails nach der Abholung gelöscht. Standardmäßig werden die per IMAP(S) abgeholten E-Mails nicht gelöscht. Hinweis:E-Mails, die per POP3(S) abgeholt werden, werden immer gelöscht. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.7.3
cmas-esb-mail	mail.encryption	*Beschreibung:* Wenn dieser Wert auf *true* gesetzt ist, ist im Ticket E-Mail Editor die Checkbox zur Verschlüsselung der E-Mail standardmäßig aktiviert. *Typ:* Boolean *Neustart erforderlich*: Nein

Modul	System-Property	Erklärung
		System: Ja *Optional*: Nein *Beispielwert:* true (Standard = false) *Seit:* 6.8.4.0
cmweb-server-adapter	mail.from	*Beschreibung:* Wenn diese E-Mail-Adresse gesetzt wird, wird diese E-Mail-Adresse anstelle der E-Mail-Adresse des Bearbeiters in E-Mail-Konversationen verwendet. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.1.2
cmas-esb-mail	mail.incoming.uri	*Beschreibung:* URL für eingehende E-Mails *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* pop3://cm-incoming-user:password@localhost:10110 *Seit:* 6.0 Dieser Wert sollte mittels der System-Properties nicht verändert werden. Die Posteingänge sollten im Admin-Tool in der Registerkarte *E-Mail* konfiguriert werden (siehe *ConSol*CM Administratorhandbuch* Abschnitt *Registerkarte E-Mail)*. Wenn Sie diese Registerkarte für die Konfiguration benutzen, können Sie

Modul	System-Property	Erklärung
		alle Einträge konfigurieren, d.h. jedes Postfach, das hinzugefügt wird. CM baut hier während der Einrichtung eines Postfachs eine Test-Verbindung auf. Auf diese Weise ist es nicht möglich, falsche Werte einzugeben.
cmas-esb-mail	mail.max.restarts	*Beschreibung:* Maximale Anzahl der Neustarts des Mailservices, bevor aufgegeben wird. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 3 *Seit:* 6.0
cmas-esb-mail	mail.mime.strict	*Beschreibung:* Wenn dieser Wert auf *false* gesetzt wird, werden E-Mail-Adressen nicht auf strikte MIME-Übereinstimmung geparsed. Standard ist *true*, was bedeutet, dass auf strikte MIME-Übereinstimmung geprüft wird. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.17, 6.7.3
cmas-esb-mail	mail.mule.service	*Beschreibung:* FROM-Adresse für E-Mails, die vom Mule-Service aus gesendet werden. *Typ:* EMail *Neustart erforderlich*: Nein *System*: Ja

Modul	System-Property	Erklärung
		Optional: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0
cmas-core-server	mail.notification.engineerChange	*Beschreibung:* Gibt an, ob eine Benachrichtigungsmail verschickt wird, wenn der Bearbeiter eines Tickets wechselt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.1.0
cmas-core-server	mail.notification.sender	*Beschreibung: From*-Adresse der Benachrichtigungsmails, die verschickt werden, wenn der Bearbeiter eines Tickets wechselt. Wenn kein Wert gesetzt wird, wird hierfür *cmas-core-security admin.email* benutzt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* cm6notification@cm6installation *Seit:* 6.6.3
cmas-esb-mail	mail.polling.interval	*Beschreibung:* Abrufintervall für E-Mails in Millisekunden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 60000 *Seit:* 6.0
cmas-esb-mail	mail.process.error	*Beschreibung:* TO-Adresse für E-Mails mit Fehlermeldungen von Mule. Standardmäßig wird die E-Mail-Adresse verwendet, die bei

Modul	System-Property	Erklärung
		der Systeminstallation als Administrator-E--Mail-Adresse angegeben wurde. *Typ:* EMail *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0
cmas-esb-mail	mail.process.retry.attempts	*Beschreibung:* Anzahl der Neuversuche, wenn E-Mails verarbeitet werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 3 *Seit:* 6.0.2
cmas-esb-mail	mail.process.timeout	*Beschreibung:* Timeout für die E-Mail-Verarbeitung in Sekunden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 60 *Seit:* 6.1.3
cmas-esb-mail	mail.redelivery.retry.count	*Beschreibung:* Gibt die Anzahl der Neuversuche an, eine E-Mail aus dem CM-System erneut zuzustellen. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 3 *Seit:* 6.1.0
cmweb-server-adapter	mail.reply.to	*Beschreibung:* Wenn dieser Wert gesetzt wird, zeigt der Web Client die gesetzte E-Mail-Adresse im Ticket-E-Mail-Editor im REPLY-TO-Feld an, und die

Modul	System-Property	Erklärung
		E-Mail wird entsprechend verschickt. Im E-Mail-Client wird somit diese REPLY-TO Adresse ebenfalls angeboten. Typ: String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1 Bitte lesen Sie dazu auch im *ConSol*CM Administratorhandbuch* den Abschnitt *Queue-Verwaltung*. Wenn Sie die *Reply-to*-Adresse in einem Skript für ausgehende E-Mails festlegen, darf die System-Property *mail.reply.to* nicht gesetzt werden, da sie den im Skript konfigurierten Wert überschreiben würde! Das bedeutet, dass wenn Sie ein Skript für ausgehende E-Mails für **eine** Queue verwenden, Sie für **alle** Queues ein Skript für ausgehende E-Mails definieren müssen, da die System-Property *mail.reply.to* nicht länger verwendet werden kann.
cmas-workflow-jbpm	mail.sender.address	*Beschreibung: From*-Adresse für E-Mails, die von der Workflow-Engine heraus versendet werden.

Modul	System-Property	Erklärung
		Typ: String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.mailFrom
cmas-core-server	mail.smtp.email	*Beschreibung:* SMTP-Mail-URL für ausgehende EMails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* smtp://mail.consol.de:25 *Seit:* 6.0
cmas-core-server	mail.smtp.envelopesender	*Beschreibung:* E-Mail-Adresse, die als Absender im SMTP-Envelope benutzt wird. Wenn nichts eingetragen wird, wird die FROM-Adresse der E-Mail benutzt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* mysender@mydomain.com *Seit:* 6.5.7
cmweb-server-adapter	mailTemplateAboveQuotedText	*Beschreibung:* Gibt das Verhalten der Templates im Ticket E-Mail Editor an, wenn eine andere E-Mail zitiert wird, d. h. auf diese geantwortet oder diese weitergeleitet wird. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.2.4

Modul	System-Property	Erklärung
cmas-core-server	max.licences.perUser	*Beschreibung:* Setzt die maximale Anzahl von Lizenzen, die ein einzelner Benutzer benutzen kann (z.B. durch Einloggen von einem anderen Browser aus). Standardmäßig ist dieser Wert nicht beschränkt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 10 *Seit:* 6.8.4.5
cmweb-server-adapter	maxSizePerPagemapInMegaBytes	*Beschreibung:* Maximale Größe (in MB) für jed e Wicket pagemap *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 15 *Seit:* 6.3.5
cmas-core-server	monitoring.engineer.login ***nur Version 6.9 und höher***	*Beschreibung:* Login des Monitoring-Bearbeiters *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:*bartek *Seit:* 6.9.3.0
cmas-core-server	monitoring.unit.login ***nur Version 6.9 und höher***	*Beschreibung:* Login des Monitoring-Benutzers *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:*bartek *Seit:* 6.9.3.0
cmas-dwh-server	notification.error.description	*Beschreibung:* Text für die E-Mails für Fehlermeldungen des DWHs.

Modul	System-Property	Erklärung
		Typ: String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Es ist ein Fehler aufgetreten *Seit:* 6.0.1
cmas-dwh-server	notification.error.from	*Beschreibung: From*-Adresse für E-Mails mit Fehlermeldungen des DWHs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1
cmas-dwh-server	notification.error.subject	*Beschreibung:* Betreff für E-Mails mit Fehlermeldungen des DWHs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Es ist ein Fehler aufgetreten *Seit:* 6.0.1
cmas-dwh-server	notification.error.to	*Beschreibung: To*-Adresse für E-Mails mit Fehlermeldungen des DWHs. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.description	*Beschreibung:* Text für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja

Modul	System-Property	Erklärung
		Optional: Nein *Beispielwert:* Übermittlung erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.from	*Beschreibung: From*-Adresse für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.subject	*Beschreibung:* Betreff für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Übermittlung erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification.finished_successfully.to	*Beschreibung: To*-Adresse für E-Mails des DWHs, wenn eine Übermittlung erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0.1
cmas-dwh-server	notification.finished_unsuccessfully.description	*Beschreibung:* Text für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja

Modul	System-Property	Erklärung
		Optional: Nein *Beispielwert:* Übermittlung nicht erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.from	*Beschreibung: From*-Adresse für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.subject	*Beschreibung:* Betreff für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* Übermittlung nicht erfolgreich beendet. *Seit:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.to	*Beschreibung: To*-Adresse für E-Mails des DWHs, wenn eine Übermittlung nicht erfolgreich beendet wurde. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* maz@consol.de *Seit:* 6.0.1
cmas-dwh-server	notification.host	*Beschreibung:* E-Mail (SMTP) Server-Hostname für das Senden von DWH-E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja

Modul	System-Property	Erklärung
		Beispielwert: mail.consol.de *Seit:* 6.1.0
cmas-dwh-server	notification.password	*Beschreibung:* Passwort für das Senden von DWH-E-Mails. (optional) *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.1.0
cmas-dwh-server	notification.port	*Beschreibung:* SMTP-Port für das Senden von DWH-E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 25 *Seit:* 6.1.0
cmas-dwh-server	notification.protocol	*Beschreibung:* Das Protokoll, welches für das Senden von E-Mails aus dem DWH heraus verwendet wird. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* pop3\
cmas-dwh-server	notification.username	*Beschreibung:* (SMTP) Benutzername für das Senden von DWH-E-Mails. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* maz *Seit:* 6.1.0
cmas-workflow-jbpm	outdated.lock.age	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein

Modul	System-Property	Erklärung
		Beispielwert: 60000 *Entfernt seit:* 6.8.0 *Ersetzt durch:* jobExecutor.lockTimeout.seconds
cmweb-server-adapter	pagemapLockDurationInSeconds	*Beschreibung:* Anzahl der Sekunden, die vergehen müssen, bevor eine Pagemap als z u lange gelockt angesehen wird. *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* 60 *Seit:* 6.7.3
cmweb-server-adapter	postActivityExecutionScriptName	*Beschreibung:* Definiert den Namen des Skripts, das nach jeder Workflow-Aktivität ausgeführt wird (siehe *ConSol*CM Administratorhandbuch*, Abschnitt *Standard-Skript für Workflow-Aktivitäten*). Wenn kein Skript ausgeführt werden soll, lassen Sie diesen Wert leer. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* postActivityExecutionHandler *Seit:* 6.2.0
cmweb-server-adapter	queuesExcludedFromGS	*Beschreibung:* Komma-separierte Liste von Queue-Namen, die von der Schnellsuche ausgeschlossen werden sollen. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0

Modul	System-Property	Erklärung
cmas-workflow-jbpm	refreshTimeInCaseOfConcurrentRememberMeRequests	*Beschreibung:* Legt die Aktualisierungszeit (in Sekunden) fest, nach der die Seite im Falle von gleichzeitigen *remember me* Anfragen neu geladen wird. Dieses Feature verhindert, dass ein Benutzer zu viele Lizenzen in Anspruch nimmt. *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* 5 *Seit:* 6.8.2
cmweb-server-adapter	rememberMeLifetimeInMinutes	*Beschreibung:* Lebensdauer für *remember me* in Minuten. *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 1440 *Seit:* 6.0
cmweb-server-adapter	request.scope.transaction	*Beschreibung:* Mit *true* oder *false* kann hier das Verhalten bzgl. Datenbank-Transaktionen bei HTTP-Requests eingestellt werden. Steht der Wert auf *true* (Standard), werden alle Service-Methoden, die ausgeführt weden müssen, in einer Datenbanktransaktion abgewickelt. Steht der Wert auf *false*, wird für jede Service-Methode eine separate Transaktion verwendet. *Typ:* Boolean *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* true *Seit:* 6.8.1

Modul	System-Property	Erklärung
cmas-setup-scene	scene	*Beschreibung:* Szenario-Datei, die während des Setups importiert wurde (kann leer gelassen werden). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/ *Seit:* 6.0
cmweb-server-adapter	searchPageSize	*Beschreibung:* Standardgröße der Seiten für Suchergebnisse. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 20 *Seit:* 6.0
cmweb-server-adapter	searchPageSizeOptions	*Beschreibung:* Optionen für Seitengröße für Suchergebnisse. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 10|20|30|40|50|75|100 *Seit:* 6.0
cmas-core-server	server.session.archive.reaper.interval	*Beschreibung:* Reaper-Intervall (in Sekunden) von archivierten Server-Sessions. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 60 *Seit:* 6.7.1

Modul	System-Property	Erklärung
cmas-core-server	server.session.archive.timeout	*Beschreibung:* Timeout der Gültigkeit der Server-Session-Archive (in Tagen). Nach diesem Zeitraum werden die Informationen zur Session aus der Datenbank entfernt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 31 *Seit:* 6.7.1
cmas-core-server	server.session.reaper.interval	*Beschreibung:* Interval (in Sekunden) , in dem der sog. *Reaper* inaktive (= beendete) Server-Sessions löscht (aus der Datenbank entfernt) *Typ:* Integer *Neustart erforderlich:* Nur Server Session Service *System*: Ja *Optional*: Nein *Beispielwert:* 60 *Seit:* 6.6.1, 6.7.1
cmas-core-server	server.session.timeout	*Beschreibung:* Server-Session-Timeout (in Sekunden) für verbundene Clients. Jeder Client kann dieses Timeout mit benutzerdefinierten Werten mittels seiner ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8 please use PORTER), ETL, REST), die an den Namen der System-Property angehängt wird, überschreiben z. B. server.session.timeout.ADMIN_TOOL *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja

Modul	System-Property	Erklärung
		Optional: Nein *Beispielwert:* 1800 *Seit:* 6.6.1, 6.7.1
cmweb-server-adapter	serverPoolingInterval	*Beschreibung:* *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Seit:* 6.1.0
cmas-dwh-server	skip-ticket	*Beschreibung:* Tickets werden während Transfer/Update nicht übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	skip-ticket-history	*Beschreibung:* Ticket-Protokoll wird während Transfer/Update nicht übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	skip-unit	*Beschreibung:* Units werden während Transfer/Update nicht übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	skip-unit-history	

Modul	System-Property	Erklärung
		Beschreibung: Unit-Protokoll wird während Transfer/Update nicht übermittelt. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* false *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmas-dwh-server	split.history	*Beschreibung:* Änder t das S QL Statement dahingehend, dass Ticketprotokolle während der DWH-Übermittlung nicht für alle Tickets auf einmal abgeholt werden, sondern ein Ticket pro SQL Statement. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* false *Seit:* 6.8.0
cmweb-server-adapter	supportEmail	*Beschreibung:* *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Seit:* 6.0
cmas-core-index-common	synchronize.master.address	*Beschreibung: Wert der Java System Property -Dcmas.http.host.port, die angibt, unter welcher URL der Index-Master erreichbar ist. Standard ist Null. Seit CM Version 6.6.17 ist dieser Wert beim Set-Up konfigurierbar, um den initialen Index-Master-Server zu bestimmen. Bitte beachten Sie, dass das Verändern dieses Wertes nur erlaubt ist, wenn alle Cluster-Nodes zum Empfang von Index-*

Modul	System-Property	Erklärung
		Veränderungen gestoppt sind. *Typ: Integer* *Neustart erforderlich: Nein* *System: Ja* *Optional: Ja* *Beispielwert: 127.0.0.1:80* *Seit: 6.6.0*
cmas-core-index-common	synchronize.master.security.token	*Beschreibung:* Das Passwort für den URL-Zugriff auf den Index-Snapshot, z.B. für die Index-Synchronisation oder für Backups. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* token *Seit:* 6.6.0
cmas-core-index-common	synchronize.master.security.user	*Beschreibung:* Der Benutzername für den URL-Zugriff auf den Index-Snapshot, z.B. für die Index-Synchronisation oder für Backups. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* user *Seit:* 6.6.0
cmas-core-index-common	synchronize.master.timeout.minutes	*Beschreibung:* Beschreibt, wie oft die Index-Synchronisation ausgehend vom akiuellen Master-Server fehlschlagen darf , bis ein neuer Master für die Index-Reparatur ausgewählt wird. Standard ist 5. Seit CM Version 6.6.17 ist dieser Wert im Setup konfigurierbar, wobei 0 bedeutet, dass der Master-Server nie geändert wird (Failover-Mechanismus

Modul	System-Property	Erklärung
		deaktiviert). *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Seit:* 6.6.0
cmas-core-index-common	synchronize.megabits.per.second	*Beschreibung:* Beschreibt, wie viel Bandbreite der Master-Server verbrauchen darf, um Index-Veränderungen an die Slave-Server zu übermitteln. Standard ist 85. Bitte benutzen Sie nicht die gesamte verfügbare Bandbreite, um die Index-Veränderungen zwischen den Hosts zu übermitteln, da dies dafür sorgen kann, dass die Nodes des Clusters nicht mehr synchron sind. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 85 *Seit:* 6.6.0
cmas-core-index-common	synchronize.sleep.millis	*Beschreibung:* Beschreibt, wie oft jeder Slave-Server den Master-Server auf Veränderungen des Indexes abfragt. Standard ist 1000. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1000 *Seit:* 6.6.0
cmweb-server-adapter	themeOverlay	*Beschreibung:* Name des verwendeten Themen-Overlays. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja

Modul	System-Property	Erklärung
		Optional: Ja *Beispielwert:* kyoEUR *Seit:* 6.0
cmas-core-server	ticket.delete.timeout	*Beschreibung:* Übermittlungs-Timeout (in Sekunden) für einen Unit Replacement Aktionsschritt *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 120 *Seit:* 6.8.2
cmweb-server-adapter	ticketListRefreshIntervalInSeconds	*Beschreibung:* Aktualisierungsintervall für die Ticketliste (in Sekunden) *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 180 *Seit:* 6.0
cmweb-server-adapter	ticketListSizeLimit	*Beschreibung:* Maximale Anzahl von Tickets in der Ticketliste. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 100 *Seit:* 6.0
cmas-core-server	tickets.delete.size	*Beschreibung:* Definiert die Anzahl der Tickets, die pro Transaktion gelöscht werden. Standardmäßig ist dieser Wert 10. *Typ:* Integer *Neustart erforderlich:* Only Session Service *System*: Ja *Optional*: Nein *Beispielwert:* 10 *Seit:* 6.8.1

Modul	System-Property	Erklärung
cmas-core-server	unit.replace.batchSize	*Beschreibung:* Beschreibt die Anzahl der Objekte, die bei einer Unit-Replace-Aktion (Übertrag von Tickets von einem Kontakt auf einen anderen) verarbeitet werden. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 5 *Seit:* 6.8.2
cmas-core-server	unit.replace.timeout	*Beschreibung:* Übermittlungs-Timeout (in Sekunden) für einen Unit-Replacement-Aktionsschritt. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 120 *Seit:* 6.8.2
cmas-dwh-server	unit.transfer.order	*Beschreibung:* Legt fest, in welche Reihenfolge Benutzerdefinierte Felder zum DWH übertragen werden. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert: company;customer* *Seit:* 6.6.19 *Entfernt seit:* 6.8.1
cmweb-server-adapter	unitIndexSearchResultSizeLimit	*Description:* Maximum number of units in unit search result (e.g. when searching for contact) *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.0

Modul	System-Property	Erklärung
cmas-core-server	unused.content.remover.cluster.node.id ***nur Version 6.9 und höher***	*Beschreibung:* Knoten, der die Cluster Node ID angibt, auf der der Content Remover läuft, welcher ungenutzte Ticket-Attachments und Unit-Inhaltseinträge entfernt. *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* 1 (in der Annahme, dass der Cluster-Node mit -Dcmas.clusternode.id=1 parameter gesetzt ist) *Seit:* 6.9.0.0
cmas-core-server	unused.content.remover.enabled ***nur Version 6.9 und höher***	*Beschreibung:* Gibt an, ob das Entfernen ungenutzter Ticket-Attachments und Unit-Inhaltseinträge durchgeführt werden soll. *Typ:* Boolean *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* true *Seit:* 6.9.0.0
cmas-core-server	unused.content.remover.polling.minutes ***nur Version 6.9 und höher***	*Beschreibung:* Gibt an, wie oft überprüft werden soll, ob ungenutzte Ticket-Attachments und Unit-Inhaltseinträge zum Entfernen vorhanden sind. *Typ:* Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 15 *Seit:* 6.9.0.0
cmas-core-server	unused.content.remover.ttl.minutes ***nur Version 6.9 und höher***	*Description:* Minimum der Dauer, nach der interval ungenutzte Ticket-Attachments und Unit-Inhaltseinträge entfernt werden können.

Modul	System-Property	Erklärung
		Typ: Integer *Neustart erforderlich*: Nein *System*: Ja *Optional*: Nein *Beispielwert:* 1440 *Seit:* 6.9.0.0
cmweb-server-adapter	urlLogoutPath	*Beschreibung:* URL die verwendet wird, wenn sich der Bearbeiter ausloggt (wenn keine Werte gesetzt werden, wird nach dem Logout die Login-Seite angezeigt). *Typ:* String *Neustart erforderlich*: Nein *System*: Ja *Optional*: Ja *Beispielwert:* http://intranet.consol.de *Seit:* 6.3.1
cmweb-server-adapter	webSessionTimeoutInMinutes	*Beschreibung:* Session-Timeout in Minuten *Typ:* Integer *Neustart erforderlich*: Ja *System*: Ja *Optional*: Nein *Beispielwert:* 180 *Entfernt seit:* 6.7.1 *Ersetzt durch:* server.session.timeout
cmweb-server-adapter	wicketAjaxRequestHeaderFilterEnabled	*Beschreibung:* Dies aktiviert Filter für Wicket AJAX-Anfragen. *Typ:* Boolean *Neustart erforderlich*: Ja *System*: Ja *Optional*: Ja *Beispielwert:* false *Seit:* 6.8.1

12 Appendix D - Hinweise zu Marken

- Microsoft® – Microsoft und Windows sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Microsoft Webseite zu Markenrichtlinien
- Microsoft® Office – Microsoft und Microsoft Office sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Microsoft Webseite zu Markenrichtlinien
- Windows® Betriebssystem – Microsoft und Windows sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Microsoft Webseite zu Markenrichtlinien
- Microsoft® Active Directory® – Microsoft und Microsoft Active Directorysind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Microsoft Webseite zu Markenrichtlinien
- Microsoft® Word® – Microsoft und Microsoft Word sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Microsoft Webseite zu Markenrichtlinien
- Microsoft® SQL Server® – Microsoft und Microsoft SQL Server sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Microsoft Webseite zu Markenrichtlinien
- MuleSoft™ und Mule ESB™ sind Marken von MuleSoft, Inc. Siehe Mule Soft Website - Terms
- Oracle® – Oracle ist eine eingetragene Marke der Oracle Corporation und/oder ihrer verbundenen Unternehmen. Siehe Oracle Trademarks Webpage
- Oracle® WebLogic – Oracle ist eine eingetragene Marke der Oracle Corporation und/oder ihrer verbundenen Unternehmen. Siehe Oracle Trademarks Webpage
- Pentaho® – Pentaho und das Pentaho-Logo sind eingetragene Marken der Pentaho Inc. Siehe Pentaho Trademark Webpage

Index

A

ACFs (Aktivitätsformulare) 75
Aktivitätsformulare (ACF) 75
Annotationen 215
Annotationen, Benutzerdefinierte Feldgruppen (Version 6.9 und höher) 242
Annotationen, Feld-Annotationen 216
Annotationen, Feldgruppen-Annotationen (Version 6.8 und niedriger) 232
Arbeitszeitkalender 169
Aufgaben (Definition) 10

B

Bearbeiter, zusätzliche (Definition) 20
Bearbeiter (Definition) 19
Bedingungen im Eigenschaften-Editor 43
Benutzerdefinierte Felder (Definition) 19
Beschreibungen im Eigenschaften-Editor 41
Bezeichnungen im Eigenschaften-Editor 41
Business Process Management 7
Business-Prozess 10

C

CM-Arbeitszeitkalender 169

G

Glossar 245

H

Hauptkunde (Definition) 19
Hauptmenü 28

I

Installieren von Workflows 210

K

Kunde, Hauptkunde (Definition) 19
Kunden, zusätzliche (Definition) 20
Kunden (Definition) 10

O

Overlays im Eigenschaften-Editor 42

P

Process Designer, Start 25
Process Designer (Kurzvorstellung) 12
Processdesignerhandbuch 6 8 U 6 9 6, 6, 18, 24, 27, 46, 48, 50, 53, 60, 70, 75, 76, 88,
97, 109, 117, 122, 129, 133, 137, 158, 165, 167, 170, 171, 179, 188, 194, 196, 210, 215, 245, 252, 353
Prozess, Business-Prozess 10
Prozess (Definition) 10

Q

Queue (Definition) 19

R

Rollen (Definition) 10

S

Scope-Eigenschaften 58
Sortier-Index im Eigenschaften-Editor 42
System Properties 252

T

Ticket (Definition) 19
Ticket-Protokoll Sichtbarkeit im Eigenschaften-Editor 44
Trigger 75

V

Verantwortlichkeiten (Definition) 10

W

Workflow (Definition) 19
Workflow (Einleitung zu ConSol*CM Workflows) 11
Workflow-Installation 210

Z

Zugriffsberechtigungen (Definition) 10
Zusätzliche Kunden (Definition) 20
Zusätzlicher Bearbeiter (Definition) 20