

Pinuts digital thinking

Universal Messenger

Handbuch für Entwickler 7.53

Die Informationen in diesem Dokument wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Wir übernehmen keine juristische Verantwortung oder Haftung für eventuell verbliebene fehlerhafte Angaben oder deren Folgen.

Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Wir richten uns im Wesentlichen nach den Schreibweisen der Hersteller.

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten, einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Inhalt

1	Information zur Dokumentation	11
2	Einführung	12
2.1	Browserbasierte Benutzeroberfläche	12
2.2	Anbindung an Content Management Systeme (CMS)	13
2.3	Formularentwicklung zur Integration in die Website	13
2.4	Core Scripting Engine (CSE)	13
2.5	Aussendung der Newsletter und E-Mails	14
2.6	Konfigurationsmöglichkeiten und Schnittstellen	14
2.7	Verzeichnisstruktur	15
3	Konfiguration der Applikation	17
3.1	Zentrale Konfigurationsdatei	17
3.2	Applikationsoptionen	17
3.2.1	Applikationsoptionen	17
3.2.2	Legacy-Optionen	25
3.3	Mehrsprachigkeit	25
3.3.1	Definition der Sprachen in der Attributkonfiguration	26
3.3.2	Abbildung der Sprachen auf die für die Benutzeroberfläche möglichen Sprachen	26
3.4	Datenbank	27
3.5	Logging	33
3.5.1	History-Log	34
3.5.2	Debugging	34
3.5.3	Transaktions-Logging	35
3.6	Versandeesellungen	35
3.6.1	Allgemeine Versandoptionen	35
3.6.2	E-Mail Versand	35
3.6.2.1	Zusätzliche Mail-Header-Elemente	40
3.6.2.2	Archivierung ausgehender E-Mails	41
3.6.2.3	Versand per Amazon SES	42
3.6.2.4	Versand via Inxmail	44
3.6.2.5	Versand via SendGrid	45
3.6.2.6	Versand via Mailjet	46
3.6.3	Mehrere Mail-Relays	46
3.6.3.1	Konfiguration	47
3.6.3.2	Auswahllogik und Failover	48
3.6.4	Inbox Preview	50
3.6.4.1	Konfiguration	50
3.6.4.2	Aufrufen aus der Benutzeroberfläche	51
3.6.4.3	Aufrufen aus einer Event-Datei (nur Inbox Preview)	51

3.6.4.4	Aufrufen aus einer Event-Datei (Inbox Preview parallel zum Test-Versand)	52
3.6.4.5	Auswertung	53
3.6.5	DKIM-Signierung	53
3.6.5.1	Erzeugung eines Schlüssels	53
3.6.5.2	Konfiguration Universal Messenger	53
3.6.5.3	Konfiguration DNS	54
3.6.6	One-Click-Unsubscribe	54
3.7	Einstellungen zum CMS	55
3.8	Prozessbenachrichtigungen	56
3.8.1	Globale Einstellungen	56
3.8.1.1	Automatischer Versand	56
3.8.2	Passwort Erinnerung	57
3.8.3	Benutzerdefinierte Benachrichtigungen	58
3.9	E-Mail- und Mobilfunknummer-Verifizierer	58
3.9.1	E-Mail-Adressen Verifizierer	58
3.9.2	Mobilfunknummern Verifizierer	58
3.10	Benutzerdefinierte Statistiken	58
3.11	Benutzerdefinierte Logdateien	59
3.12	Import/Export	60
3.13	Zusatzmodule	60
3.14	Brute-Force-Angriffe erschweren	60
3.15	Trigger E-Mails	63
3.16	Eingebetteter Tomcat	63
3.16.1	Starten	63
3.16.2	Stoppen	64
3.16.3	Starten im Vordergrund	64
3.16.4	Starten als systemd-Service	64
3.16.5	Konfiguration des eingebetteten Tomcat	64
3.16.5.1	cmsbs.server.port	64
3.16.5.2	cmsbs.server.address	64
3.16.5.3	cmsbs.server.context	65
3.16.5.4	cmsbs.server.retryStartupDelay	65
3.16.5.5	cmsbs.directory.server.webapps	65
3.16.5.6	cmsbs.directory.server.work	65
3.16.5.7	cmsbs.log.server	65
3.16.5.8	cmsbs.server.maxThreads	65
3.16.5.9	cmsbs.server.acceptCount	65
3.16.5.10	cmsbs.server.sessionCookie.global	65
3.16.5.11	cmsbs.server.sessionCookie.name	65
3.16.5.12	cmsbs.server.session.Dir	66

3.16.5.13	cmsbs.server.cookies.secure	66
3.16.5.14	cmsbs.server.cookies.sameSite	66
3.16.5.15	cmsbs.server.accesslog.enabled	67
3.16.5.16	cmsbs.server.accesslog.dir	67
3.16.5.17	cmsbs.server.accesslog.maxDays	67
3.16.5.18	cmsbs.server.accesslog.pattern	67
3.16.5.19	cmsbs.server.accesslog.buffered	67
3.16.5.20	cmsbs.server.h2.port	67
3.16.5.21	cmsbs.server.errorReporting.secure	68
3.16.5.22	JVM-Speichereinstellungen	68
3.16.6	SSL/TLS-Konfiguration	68
3.16.6.1	Zertifikat in KeyStore importieren	69
3.17	API-Token	70
3.18	Virens Scanner einbinden	71
3.18.1	Unterstützte Virens Scanner	71
3.18.2	Einrichtung	71
3.18.3	Testen der Virens Scannerintegration	72
4	Konfiguration des Datenmodells	74
4.1	Überblick zum Datenmodell des Universal Messenger	74
4.2	Objekt-relationales Mapping	74
4.3	Speicherung von Null-Werten oder leeren Werten	75
4.4	Änderungen der Attributkonfiguration	76
4.5	Bereitstellung für externe Schnittstellen	77
4.6	Dateien der Attributkonfiguration	77
4.6.1	Format der Konfigurationsdateien	78
4.6.1.1	Wert einer Liste hinzufügen	78
4.6.1.2	Wert aus Liste entfernen	78
4.6.1.3	Liste überschreiben	78
4.6.1.4	Andere .attributes-Datei inkludieren	78
4.7	Interne und vordefinierte Attribute	78
4.7.1	Interne Attribute	78
4.7.2	Vordefinierte Attribute	79
4.7.3	Änderung von vordefinierten Attributen	80
4.8	Kundenspezifische Attribute	80
4.8.1	Konfiguration zusätzlicher Attribute	81
4.8.1.1	Freie Attribut-Konfigurations-Parameter	85
4.8.1.2	Reservierte Wörter, die nicht als Attributnamen verwendet werden dürfen	86
4.9	Attributtypen	87
4.9.1	STRING	87
4.9.2	TEXT	88

4.9.3	HTML	89
4.9.4	TIMESTAMP und DATE	89
4.9.5	EMAIL	90
4.9.6	COOKIE	90
4.9.7	PASSWORD	91
4.9.7.1	Passwortqualität	92
4.9.8	TABLE	93
4.9.9	REFERENCE	95
4.9.10	ATTACHMENT	96
4.9.11	WIZARD	98
4.10	Gruppierung der Attribute	98
4.10.1	Änderung der Gruppierung von vordefinierten Attributen	100
4.11	Datentypen / Entry-Types	101
4.11.1	Verwendung von Entry-Types aktivieren	102
4.11.2	Auswahlattribut konfigurieren	102
4.11.3	Definition der Entry-Types	103
4.11.3.1	Attributgruppen auswählen	103
4.11.3.2	Attributgruppen hinzufügen	103
4.11.3.3	Attributgruppen umbenennen	103
4.11.3.4	Attribute zuordnen	104
4.11.3.5	Attribute umbenennen	104
4.11.3.6	Defaultwert anpassen	104
4.11.3.7	Wertebereich anpassen	104
4.11.3.8	Bearbeitung zulassen / unterbinden	104
4.11.4	Weitere Hinweise	104
4.11.5	Entry-Types und Adminrollen	104
4.12	Composite Unique Key	105
4.13	Einwilligung und Widerruf	106
4.13.1	Standard-Attributkonfiguration	106
4.13.2	Eigene Einwilligungen	107
4.13.3	Historien-Tabelle consent_history	108
4.14	Konfigurationsfragmente unter conf.d	110
5	Konfiguration der Benutzeroberfläche	112
5.1	Allgemeine Einstellungen	112
5.1.1	Allgemeine Einstellungen	112
5.1.2	Bearbeitung von Listen ("Channels")	114
5.1.2.1	Custom-Attribute	114
5.2	Symbole für Einträge	115
5.3	Überschrift für Einträge	116
5.4	Suchfunktion	116

5.4.1	Suchformular	116
5.4.2	Trefferliste	116
5.4.3	Definition der Spalten	117
5.5	Login	118
5.5.1	User-Authentifizierung	118
5.5.1.1	Login-Modus "default" – seit Release 7.46	118
5.5.1.2	Projektspezifische Authentifizierungsmethode	123
5.5.1.3	Backoffice-Oberfläche ohne User-Authentifizierung	123
5.5.1.4	Legacy-Modi	123
5.5.2	Weitere Optionen	124
5.5.3	Anlegen von Backoffice-Usern	124
5.6	Rechte und Rollen	125
5.6.1	Rechte und Rollen	125
5.6.1.1	Einleitung	125
5.6.1.2	Konfigurationsdateien	125
5.6.1.3	Einrichten einer Rolle	126
5.6.1.4	Funktionen der Benutzeroberfläche	126
5.6.1.5	Zugriff auf Apps	130
5.6.1.6	Zugriff auf Einträge	131
5.6.1.7	Zugriff auf Attribute und Attributgruppen	131
5.6.1.8	Zugriff auf bestimmte Listen, Segmente, Jobs und Mailingvorlagen einschränken	133
5.6.1.9	Symbole für Einträge	133
5.6.1.10	Trefferliste der Suchfunktion	133
5.6.1.11	Aktivieren der Beispielrollen	133
5.6.2	Verwendung von Absender- und Reply-To-Adressen einschränken	134
5.6.3	Darstellung der Attributgruppenliste in der Eintragsansicht- und -Bearbeitungsseite	135
5.7	Mandantenfähigkeit	136
5.7.1	Zusammenhang zwischen Rechte&Rollen und Mandantenfähigkeit	136
5.7.2	Einrichtung	137
5.7.2.1	Attribut "tenant" einrichten	137
5.7.2.2	E-Mail-Adresse nicht mehr eindeutig	138
5.7.2.3	Admin-Rollen	138
5.7.2.4	Login-Modus "default"	139
5.7.2.5	Listen, Segmente, Mailingvorlagen, Jobs und Apps anpassen	140
5.7.2.6	Newsletterversand	140
5.7.2.7	Newsletterformulare / Newsletter App	141
5.7.2.8	Import von Einträgen	141
5.7.2.9	Newsletter-Tracking	142
5.7.2.10	Verwendung von Absender- und Reply-To-Adressen einschränken	143

6	Personalisierung	144
6.1	Verwendbare Variablen	144
6.2	Kontrollstrukturen	147
6.2.1	switch-Anweisung	148
6.2.2	if-Anweisung	148
6.2.3	foreach-Anweisung	149
6.2.4	ifcse-Anweisung	149
6.2.4.1	ifcse specialVarIs	150
6.2.4.2	ifcse specialVarContains	150
6.2.4.3	ifcse specialVarIsSet	150
6.2.4.4	ifcse specialVarIsEmpty	151
6.2.5	withcse-Anweisung	151
6.2.5.1	_pre()-Methode	152
6.2.6	Kontrollstrukturen für Split-Test-Newsletterversand	153
6.2.6.1	splitTestSwitch-Anweisung	153
6.2.6.2	splitTestIf-Anweisung	153
6.3	Verwendbare Funktionen	153
6.4	Benutzerdefinierte Anredeformel	156
6.5	Attachments	157
6.6	Eigene Variablen	158
7	Abfragesprache	161
7.1	Aufbau einer Abfrage	161
7.2	Operatoren	162
7.3	Sortierung der Ergebnisse	166
7.4	Abfragen für Tabellenattribute	167
7.4.1	Extrema	168
7.4.2	Aggregationen	168
7.5	Abfragen für Newsletter	169
7.6	Abfragen für Einwilligungen	172
7.7	Abfragen für Referenzattribute	172
7.7.1	Beispiel	173
7.7.2	Hinweise	173
7.8	Beispiele für Abfragen	173
8	Bounce Management	174
8.1	Konfiguration des Bounce Management	175
8.1.1	IMAP mit OAUTH2	179
8.2	Organisation des Regelwerks	180
8.2.1	Auslieferung	180
8.2.2	Kundenspezifische Anpassungen	181
8.3	Aufbau der Regeln	181

8.4	Spam	182
8.5	Hardbounces	182
8.6	Auto-Respondmessages	183
8.7	Unsubscribes	183
8.8	Mehrere Bouncelistener	184
8.8.1	Reihenfolge der Bouncelistener	185
8.9	Bounce Management per VERP	185
8.9.1	Bounce Management per VERP	185
8.9.2	Konfiguration von VERP	186
8.9.3	Einrichtung der beim Newsletterversand benötigten E-Mail-Adressen	187
8.9.4	Konfiguration von VERP im IMAP-Server	189
8.9.5	Konfiguration von VERP im SMTP-Server	189
8.10	E-Mail-Empfang per POP3	189
8.11	Bounce Management mit AWS	190
8.11.1	Konfiguration in AWS / SNS	191
8.11.2	Konfiguration in AWS / SES	191
8.11.3	Validierung / Bestätigung der SNS-Topics	191
8.12	Bounce Management mit SendGrid	191
8.12.1	Konfiguration in SendGrid	192
8.13	Bounce Management mit Mailjet	193
8.13.1	Konfiguration in Mailjet	193
8.14	Feedback-Loops	194
8.15	Archivierung eingegangener E-Mails	195
8.15.1	Zugeordnete Bounces	195
8.15.2	Nicht-zuordenbare Bounces	196
9	Newsletter Controlling	197
9.1	Konfiguration des Trackings	197
9.1.1	Tracking der Newsletter Öffnungen	199
9.1.2	Link-Tracking	199
9.1.2.1	Delivery-Ticket / msgid an Links anhängen	200
9.1.2.2	Conversion beim Linktracking	201
9.1.3	Konfiguration im Event-File	201
9.2	Installation und Anwendung	201
9.2.1	Tracking per REST-Proxy	202
9.2.2	Integration in Newsletter	204
9.2.2.1	Link-Tracking Konfiguration	204
9.2.3	Linktracker gegen URL-Manipulationen absichern	206
9.3	Schlagworte und Filter	207
10	Newsletterversand	209
10.1	Parametrisierter GUI Einsprung	209

10.2 XML-Eventdatei	210
10.2.1 Variablen zur Personalisierung mit <global>	211
10.2.1.1 <global>	211
10.2.2 XML-Elemente und Attribute	212
10.2.2.1 <event>	212
10.2.3 Übergabe der Eventdatei via REST-Aufruf	219
10.2.3.1 Status eines Newsletterversands abfragen	219
11 Anhang	222
11.1 Formatierungszeichen für Datum und Uhrzeit	222
11.1.1 Beispiele	223
11.1.2 Datumsformat der XML-Eventdatei	223
11.2 Reguläre Ausdrücke	223
11.3 Tracking-Client Integration	225
12 Handbuch für Entwickler 7.52	227

1 Information zur Dokumentation

Dieses Handbuch wendet sich an Entwickler, die den Universal Messenger konfigurieren und über die offenen Schnittstellen in ihre Website bzw. ihre Anwendungen integrieren möchten.

Die Installation des Universal Messenger ist im Installationshandbuch beschrieben, die weiterführende Konfiguration wird jedoch in dieser vorliegenden Dokumentation erläutert, so dass sie sich in Teilen ebenso an Systemadministratoren richtet.

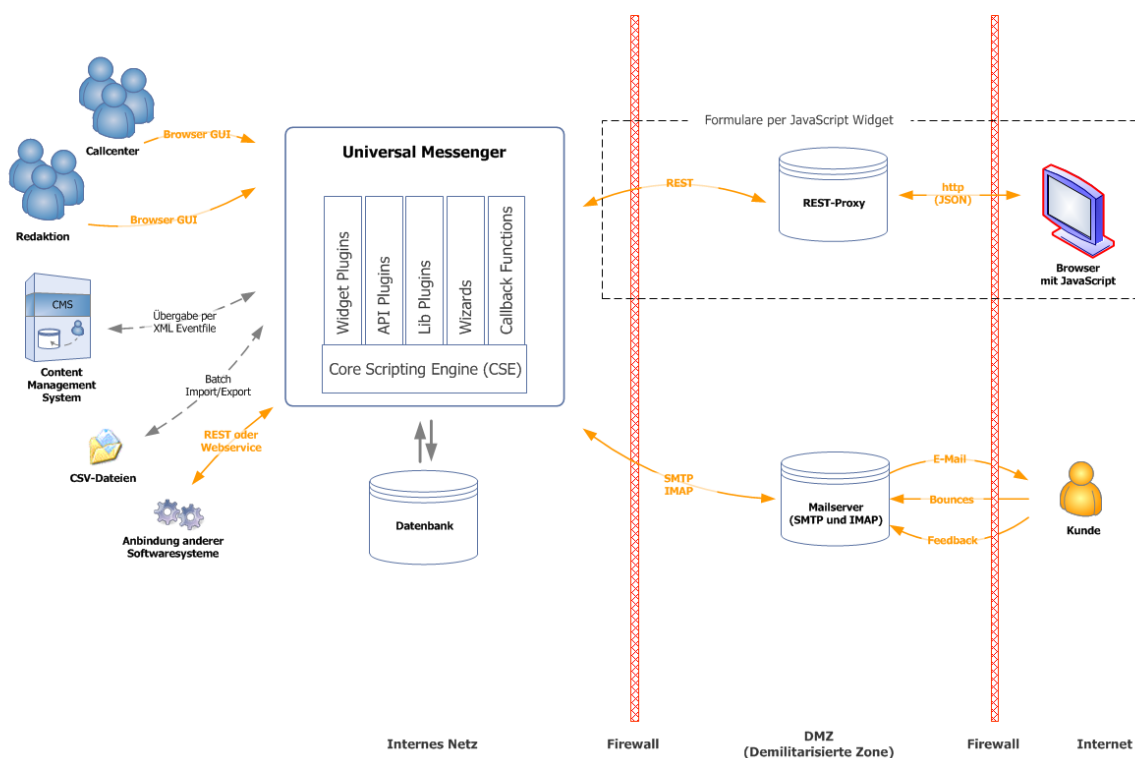
Der Universal Messenger kann ebenso zum Newsletter-Management, wie als Kundendatenbank eingesetzt werden, in den Handbüchern sprechen wir deswegen auch allgemein von "Einträgen" statt von "Abonnenten" oder "Kunden".

Dieses Handbuch beschreibt den vollen Funktionsumfang des Universal Messenger, das je nach Lizenzierung nicht in allen Editionen (siehe Handbuch "Einführung und Systemüberblick") zur Verfügung steht. Nähere Informationen zum Lizenzmodell entnehmen Sie bitte unseren Produktbeschreibungen.

2 Einführung

Der Universal Messenger ist technisch gesehen eine Webanwendung nach der Java-Servlet-Spezifikation, die als Web-Archiv ausgeliefert wird und in einem standardkonformen Webcontainer (z.B. dem frei verfügbaren Apache Tomcat) lauffähig ist. Zur persistenten Datenspeicherung wird eine per JDBC verbundene SQL-Datenbank über ein Object-Relational-Mapping (OR-Mapper) verwendet.

Die Architektur des Universal Messenger und die Einbindung in die Systemlandschaft sind im Handbuch "Einführung und Systemüberblick" beschrieben:



2.1 Browserbasierte Benutzeroberfläche

Ein direkter Zugriff auf die browserbasierte Benutzeroberfläche erfolgt wie im linken Teil des Schaubilds zu sehen in der Regel nur durch die Anwender (z.B. Customer Service Center, Newsletter-Redakteure, Marketing) in Ihrem Unternehmen. Der Import und Export von Datensätzen kann alternativ zur browserbasierten Benutzeroberfläche auch über die Kommandozeile bzw. per Shell-Skript als Batchlauf ausgeführt werden.

2.2 Anbindung an Content Management Systeme (CMS)

Der Universal Messenger kann mit einem für die Pflege der Website vorhandenen Content Management System (CMS) über eine XML-basierte Schnittstelle verbunden werden, so dass die Erstellung des Newsletters in der gewohnten Umgebung des Content Management Systems und mit den vorhandenen Workflows erfolgen kann. Die vom Content Management System erstellten Newsletter werden mit weiteren Versandinformationen per XML Eventfile an den Universal Messenger übergeben.

2.3 Formularentwicklung zur Integration in die Website

Bei der "Formulare per JavaScript Widget" wird in den HTML-Code der Website lediglich ein kurzer JavaScript-Codeabschnitt ("Snippet") eingefügt. Mit diesem Code wird der Browser bei der Darstellung der Webseite angewiesen, das Formular per Ajax nachzuladen und für den Anwender im Browser darzustellen. Die Formulareingaben werden vom Browser per Ajax im JSON-Format an den Universal Messenger übertragen. Die Übertragung erfolgt über einen REST-Proxy, der zwischen dem Internet und dem Universal Messenger positioniert ist, um den Zugriff auf den Universal Messenger abzusichern. Der Universal Messenger sollte aus Sicherheitsgründen immer in einem geschützten internen Netzbereich installiert werden. Der REST-Proxy stellt die Verbindung zwischen dem öffentlichen Internet und dem Universal Messenger her und führt einige weitere Sicherheitsfunktionen aus.

2.4 Core Scripting Engine (CSE)

Die Core Scripting Engine (CSE) ist das Herzstück des Universal Messenger und dient zur Realisierung der Business Logik. Die CSE stellt eine API mit diversen Erweiterungspunkten innerhalb des Universal Messenger für die Implementierung kundenspezifischer Anforderungen zur Verfügung. Die Entwicklung für die CSE erfolgt in der Sprache JavaScript (ECMAScript ECMA-262) mit einer Implementierung durch Mozilla Rhino. Für die Entwickler stehen diverse Klassen und spezielle Objekte zur Verfügung, um auf das Datenmodell des Universal Messenger zugreifen zu können. Per Callback können CSE-Funktionen z.B. beim Speichern eines Datensatzes im Universal Messenger automatisch aufgerufen werden. Die Benutzeroberfläche des Universal Messenger kann über Wizards erweitert werden, die mit der CSE realisiert werden. Als Schnittstelle zu externen Anwendungen aber auch für die Formularentwicklung können CSE-Funktionen über die REST-Schnittstelle zugänglich gemacht werden.

Für die im Newsletter-Marketing üblichen Prozesse wie z.B. das Double Opt-In Verfahren sind die CSE-Funktionen im Lieferumfang des Universal Messenger enthalten. Weitere Funktionen können über Plugins ergänzt oder individuell entwickelt werden. Für die Entwicklung eigener CSE-Funktionen wird mindestens der Universal Messenger Customer Interaction Edition benötigt und die Teilnahme an einem Workshop bzw. einer Schulung gilt als Voraussetzung.

2.5 Aussendung der Newsletter und E-Mails

Die Aussendung der E-Mails für den Newsletterversand und der Empfang der Rückläufer ("Bounces") und eingehenden E-Mails erfolgt über einen vorhandenen oder zusätzlich bereitgestellten Mailserver, der mit den Standardprotokollen SMTP und IMAP mit dem Universal Messenger zusammenarbeitet.

2.6 Konfigurationsmöglichkeiten und Schnittstellen

Da der Universal Messenger vielfältige Möglichkeiten zur Konfiguration und zahlreiche Schnittstellen bietet, soll hier zunächst ein Überblick vermittelt werden, der auch als Einstiegspunkt in die Dokumentation dienen kann.

Konfiguration der Applikation

Alle grundsätzlichen Einstellungen zum Universal Messenger werden in der zentralen Konfigurationsdatei `cmsbs.properties` vorgenommen. Dies reicht von allgemeinen Applikationsoptionen (z.B. zur Datenbank) bis zur Konfiguration von benutzerdefinierten Benachrichtigungen, siehe [Konfiguration der Applikation](#).

Konfiguration des Datenmodells

Die in der Standardkonfiguration zu jedem Eintrag gespeicherten Attribute können in der Konfigurationsdatei `additional.attributes` durch zusätzliche Attribute mit verschiedenen Attributtypen beliebig erweitert werden. Die Attribute können in Attributgruppen sortiert und den Datentypen ("Entry-Types") zugeordnet werden, siehe [Konfiguration des Datenmodells](#).

Konfiguration der Benutzeroberfläche

Die Benutzeroberfläche des Universal Messenger kann an die individuellen Anforderungen angepasst werden, das Login zu der Benutzeroberfläche kann z.B. per LDAP erfolgen. Die in der Benutzeroberfläche zur Verfügung stehenden Funktionen und die Rechte der Benutzer können über die Konfiguration der Rechte und Rollen eingeschränkt werden, siehe [Konfiguration der Benutzeroberfläche](#).

Personalisierung und Abfragen

Zur Personalisierung des Newsletters und auch der übrigen Kundenkommunikation können die zu jedem Eintrag gespeicherten Attribute verwendet werden. Mit den Kontrollstrukturen und der Abfragesprache stehen weitere sehr umfangreiche Möglichkeiten zum Zugriff auf die vorhandenen Einträge und Attribute zur Verfügung, siehe [Personalisierung](#) und [Abfragesprache](#).

Bounce Management

Die Erkennung, Zuordnung und Bearbeitung der E-Mail Rückläufer ("Bounces") ist die Aufgabe des Bounce Managements, das für optimale Ergebnisse eine abgestimmte Konfiguration des Universal Messenger mit dem SMTP-Server für den Versand der E-Mails und mit dem IMAP-Server für den Empfang der Bounces erfordert, siehe [Bounce Management](#).

Newsletter Controlling

Über die Auswertung der Öffnungsrate eines Newsletters und der Klickraten der darin enthaltenen Links können wichtige Erkenntnisse für die Personalisierung und das Kundenbeziehungsmanagement gewonnen werden, siehe [Newsletter Controlling](#).

Anbindung von Content Management Systemen

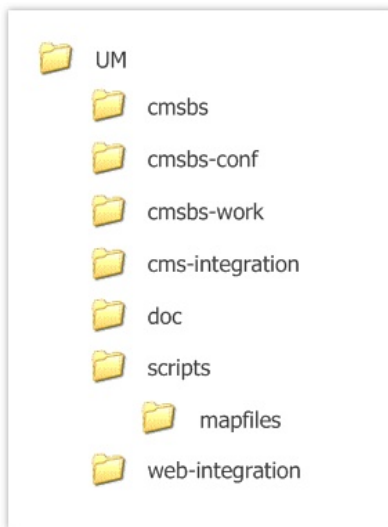
Der Universal Messenger kann mit einem für die Pflege der Website vorhandenen Content Management System (CMS) über XML-basierte Schnittstellen verbunden werden, so dass die Erstellung des Newsletters in der gewohnten Umgebung des Content Management Systems und mit den vorhandenen Workflows erfolgen kann. Die Anbindung an das CMS muss in der Konfigurationsdatei aktiviert werden, siehe [Einstellungen zum CMS](#). Weitere Details zur Anbindung an das CMS sind in einer separaten Dokumentation ausführlich beschrieben, die zu Grunde liegende XML-Eventdatei wird jedoch in dieser Dokumentation erläutert, siehe [XML-Eventdatei](#).

Import und Export

Der Massenimport bzw. Export von Einträgen erfolgt häufig über CSV-Dateien, da diese von vielen anderen Programmen und auch von Excel gelesen werden können. In den hierfür benötigten Map-Dateien können vielfältige Optionen und Konvertierungsfunktionen für den Datenaustausch definiert werden. Die Funktionen des hierfür benötigten Hilfsprogramms UserTransfer sind im Handbuch für Administratoren beschrieben.

2.7 Verzeichnisstruktur

Nachdem der Universal Messenger wie im Installationshandbuch beschrieben installiert wurde, ergibt sich folgende Verzeichnisstruktur:



Im Verzeichnis `cmsbs` befindet sich die Universal Messenger Webapplikation.

Im Verzeichnis `cmsbs-conf` befinden sich die Konfigurationsdateien des Universal Messenger mit der zentralen Konfigurationsdatei `cmsbs.properties`.

Im Verzeichnis `cmsbs-work` bzw. in einigen automatisch angelegten Unterverzeichnissen werden in der Standardkonfiguration die Logfiles des Universal Messenger abgelegt. Weiterhin dient dieses Verzeichnis dem Datenaustausch mit einem angebundenen Content Management System.

Im Verzeichnis `cms-integration` befindet sich einige vorbereitete Dateien für die Anbindung an Content Management Systeme, die in einer separaten Dokumentation beschrieben sind. Für Informationen zur Anbindung an weitere Content Management Systeme fragen Sie bitte bei unserem Support an.

Im Verzeichnis `doc` können Sie die Handbücher zum Universal Messenger finden. Für einen ersten Systemüberblick beginnen Sie bitte mit dem Handbuch `Introduction.pdf`.

Im Verzeichnis `scripts` befindet sich Utilities zu dem Universal Messenger, mit denen z.B. der Import und Export per Batch-Job ausgeführt werden kann.

Im Verzeichnis `web-integration` befindet sich der mitgelieferte REST-Proxy. Wir empfehlen jedoch diesen direkt über die GUI des Universal Messenger unter "Extras > REST-Proxy" zu konfigurieren und herunterzuladen.



Mit Version 7.2 des Universal Messenger wurden "Channels" in "Listen" und "virtuelle Channels" in "Segmente" umbenannt. Als interne Namen werden weiterhin `channel` und `vchannel` verwendet.

3 Konfiguration der Applikation

3.1 Zentrale Konfigurationsdatei

Der Universal Messenger wird über die zentrale Konfigurationsdatei

`<UM_HOME>/cmsbs-conf/cmsbs.properties` konfiguriert. Nach einer manuellen Änderung der Einstellungen muss der Tomcat neu gestartet werden.

Einige Einstellungen zu den Benachrichtigungen können auch über die Benutzeroberfläche des Universal Messenger geändert werden. Der Universal Messenger schreibt nach Abschluss der Änderungen eine neue Version der Konfigurationsdatei. Eine Sicherheitskopie der vorhergehenden Einstellungen wird in das Verzeichnis `<UM_HOME>/cmsbs-conf/backup` (bzw. das in der Konfigurationsdatei eingestellte Verzeichnis) kopiert und mit dem aktuellen Datum und aktueller Uhrzeit benannt.

Format der Konfigurationsdatei

Kommentare beginnen mit einem `#` und reichen bis zum Ende der Zeile. Eine Option hat den Aufbau `<name> = "<wert>"` und steht innerhalb einer Zeile. Die Einstellungen sollten in der Regel mit Anführungszeichen eingeschlossen werden, bei der Verwendung von Leerzeichen und Sonderzeichen sind sie zwingend erforderlich.

Alle Pfade können relativ zu `<UM_HOME>` oder absolut angegeben werden. Unter Windows-Betriebssystemen müssen die Pfade statt des Backslash mit einem vorwärtsgeneigten Schrägstrich eingegeben werden (z.B. `C:/tomcat/log`), alternativ ist auch `c:\\tomcat\\log` möglich.

Die Konfigurationsdateien `cmsbs.properties`, `additional.attributes` und die Dateien zur Definition der Rechte und Rollen haben immer das Encoding `latin1 (iso-8859--1)`, auch wenn an anderen Stellen in dem Universal Messenger ein abweichendes Encoding eingestellt ist. Falls in diesen Dateien spezielle Zeichen aus anderen Encodings verwendet werden müssen, empfehlen wir das Programm `native2ascii` zu verwenden. Von `native2ascii` werden alle nicht-ASCII Zeichen in eine Umschreibung der Form `\uXXXX` umgewandelt. Für den Einsatz von `native2ascii` muss die Umgebungsvariable `LC_TYPE` auf das Encoding der Eingabedatei gesetzt werden, z.B. also auf `de_DE`, `de_DE@euro` oder `de_DE.UTF-8`.

3.2 Applikationsoptionen

3.2.1 Applikationsoptionen

```
cmsbs.server.home = <Pfad>
```

Pfad zum Installationsverzeichnis `<UM_HOME>`. Alle anderen Pfade können relativ zu diesem Homeverzeichnis angegeben werden.

Unter Windows-Betriebssystemen müssen die Pfade statt des Backslash mit einem vorwärtsgeneigten Schrägstrich eingegeben werden. Der Pfad darf nicht durch einen Backslash am Ende abgeschlossen werden!

```
cmsbs.license = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Lizenzdatei, die für den Start des Universal Messenger notwendig ist. Ist diese Konfigurationsoption nicht angegeben, wird die Lizenzdatei `cmsbs.license` aus dem Installationsverzeichnis `<UM_HOME>` gelesen.

```
cmsbs.license.sender = <Dateiname mit Pfad>
```

```
cmsbs.license.replyto = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der sender und replyto CSV-Dateien.

```
cmsbs.license.csv.encoding = <Encoding>
```

Encoding der CSV-Dateien für sender und replyto. Der Default-Wert ist "iso-8859-1".

```
cmsbs.pid = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Prozessidentifikationsdatei, die sicherstellt, dass der Universal Messenger nur in einer Instanz gleichzeitig gestartet wird. Ist diese Konfigurationsoption nicht angegeben, wird die Prozessidentifikationsdatei `cmsbs.pid` aus dem Installationsverzeichnis `<UM_HOME>` gelesen.

```
cmsbs.runas = <Username>
```

Nur Linux: Name des Users, unter dem der Installer ausgeführt wurde und unter dem der Universal Messenger laufen soll.

```
cmsbs.startsender = true|false
```

`true` startet den Versandmechanismus, `false` verhindert den Start. Der Versand erfolgt über einen Prozess, der in regelmäßigen Abständen nach in `cmsbs.directory.listen` abgelegten XML-Eventdateien sucht.

```
cmsbs.startbounce = true|false
```

`true` startet die Bounceverarbeitung, `false` verhindert den Start. Die Bounceverarbeitung erfolgt über einen Prozess, der in regelmäßigen Abständen das IMAP-Postfach `cmsbs.mail.imap.user` nach eingehenden Bounces abfragt.

```
cmsbs.jobs.shellExec = true|false
```

Seit Version 7.13 des Universal Messenger steuert dieser Schalter, ob der Jobtyp "Programm ausführen" zur Verfügung steht. Der Default-Wert steht auf `true`, so dass existierende Installation den Job weiterhin einfach benutzen können.



In der Konfigurationsdatei `cmsbs.properties`, die bei neuen Installationen (ab Version 7.13) erzeugt wird, steht der Schalter jedoch auf `false`, so dass neue Systeme diesen Jobtyp nicht direkt benutzen können.

```
cmsbs.jobs.lock = <Dateiname mit Pfad>
```

Wenn die mit dieser optionalen Konfigurationsoption angegebene Datei existiert, werden keine Jobs ausgeführt. In der Standardkonfiguration ist diese Einstellung auf `<UM_HOME>/cmsbs-work/nojobs.lock` gesetzt.

Durch das Anlegen einer leeren Datei kann auf diese Weise das Starten aller neuen Jobs unterbrochen werden, ohne den Universal Messenger anhalten zu müssen.

```
cmsbs.jobs.appendLog = true|false
```

Wenn Logfiles von Jobs überschrieben werden sollen, muss diese Option auf `true` gesetzt werden.

```
cmsbs.directory.confdir = <Pfad>
```

Legt das Verzeichnis fest, in dem spezifische Konfigurationsfragmente abgelegt werden, die beim Starten des des Universal Messenger geladen werden sollen.

```
cmsbs.directory.listen = <Pfad>
```

Legt das Verzeichnis fest, in dem der Versandmechanismus regelmäßig nach neuen Nachrichten sucht und der Versandmechanismus der Weboberfläche neue Versandaufträge ablegt.

```
cmsbs.directory.listen.syncWithVfs = true|false
```

Legt fest, ob der Inhalt des Listenverzeichnisses (s.o.) in die Datenbank synchronisiert werden soll. Das ist vor allem in Setups ohne persistenten Storage nützlich (z.B. "Cloud"-Setups). Default: `false`

```
cmsbs.directory.processed = <Pfad>
```

In dieses Verzeichnis werden alle erfolgreich bearbeiteten XML-Eventdateien nach dem Versand der Nachricht verschoben. Das Verzeichnis muss auf dem selben Datenträger liegen wie `cmsbs.directory.listen`, da das Verschieben durch ein Umbenennen der Datei geschieht!

```
cmsbs.directory.error = <Pfad>
```

In dieses Verzeichnis werden alle XML-Eventdateien, bei deren Abarbeitung ein Fehler aufgetreten ist, verschoben. Das Verzeichnis muss auf dem selben Datenträger liegen wie `cmsbs.directory.listen`, da das Verschieben durch ein Umbenennen der Datei geschieht!

```
cmsbs.directory.data = <Pfad>
```

In diesem Verzeichnis sucht der Versandmechanismus alle Dateien, die aus der XML-Eventdatei referenziert werden (HTML-Mail Texte, Dateianhänge, etc.).

```
cmsbs.directory.backup = <Pfad>
```

Nachdem über die Benutzeroberfläche Änderungen an der Konfiguration vorgenommen wurden, wird diese Konfigurationsdatei von dem Universal Messenger neu geschrieben. Zuvor wird eine Sicherheitskopie in dem hier angegebenen Verzeichnis abgelegt.

```
cmsbs.directory.jobs = <Pfad>
```

In diesem Verzeichnis wird für jeden neu eingerichteten Job ein eigenes Arbeitsverzeichnis mit dem internen Namen des Jobs angelegt. In der Standardkonfiguration werden die Arbeitsverzeichnisse der Jobs in `<UM_HOME>/cmsbs-work/data/jobs/` angelegt.

```
cmsbs.directory.doc = <Pfad>
```

In diesem Verzeichnis wird die Dokumentation zu dem Universal Messenger abgelegt, die von der Benutzeroberfläche aufrufbar ist. Die Angabe erfolgt relativ zu `<UM_HOME>` und ist in der Standardkonfiguration auf `doc` gesetzt.

```
cmsbs.directory.upload = <Pfad>
```

In diesem Verzeichnis werden temporär hochgeladene Dateien abgelegt. Die Angabe erfolgt relativ zu `<UM_HOME>` und ist in der Standardkonfiguration auf `cmsbs-work/upload` gesetzt.

```
cmsbs.directory.tmp = <Pfad>
```

Verzeichnis für temporäre Dateien. Default ist das tmp-Directory des Systemusers, also unter Linux z.B. zumeist `/tmp/`. Das Verzeichnis muss jedoch bereits existieren, da es nicht automatisch angelegt wird.

```
cmsbs.emptypassword = true|false
```

Bei `true` kann das Passwort beim Eintragen eines neuen Abonnenten leer bleiben, bei `false` muss ein Passwort angegeben werden. Falls das Passwort fehlt, wird automatisch ein zufälliges Passwort erzeugt.

Hinweis: Diese Option ist ab Version 7.25 nicht mehr verfügbar. Stattdessen kann nun bei jedem einzelnen PASSWORD-Attribut in der `additional.attributes` mit `".allowEmpty"` angegeben werden, ob leere Passworte zugelassen werden sollen.

```
cmsbs.password.length = <Länge des Passworts>
```

Länge des automatisch erzeugten Passworts, falls das Passwort von dem Universal Messenger generiert wird, siehe auch `cmsbs.emptypassword`.

```
cmsbs.password.alphabet = <Zeichenauswahl>
```

Angabe der Zeichen (ohne Leerstellen als Zeichenkette angeben), aus denen das Passwort automatisch generiert werden soll.

```
cmsbs.password.encoding = <Encoding>
```

Angabe des Encodings für Passwörter. Die Standardeinstellung ist "UTF-8".

```
cmsbs.password.salt = <Hashing Salt>
```

Salt zum Verhaschen von Passwörtern. Wird nur für die veralteten Hashingalgorithmen SSHA und MD5 verwendet. Für PBKDF2_1 wird für jedes Passwort ein individueller Salt verwendet.

Setzen des Wertes auf "{random}" bewirkt, dass beim ersten Start ein Zufallswert erzeugt und in der Datenbank gespeichert wird. Bei allen folgenden Starts wird dann der Wert aus der Datenbank verwendet.

Wird der Wert später in der Konfigurationsdatei auf einen anderen festen Wert geändert, so wird dieser Wert beim nächsten Start in die Datenbank übernommen.

```
cmsbs.msgId.secret = <Hashing Salt>
```

Salt zum Verhaschen von Message IDs.

Setzen des Wertes auf "{random}" bewirkt, dass beim ersten Start ein Zufallswert erzeugt und in der Datenbank gespeichert wird. Bei allen folgenden Starts wird dann der Wert aus der Datenbank verwendet.

Wird der Wert später in der Konfigurationsdatei auf einen anderen festen Wert geändert, so wird dieser Wert beim nächsten Start in die Datenbank übernommen.

```
cmsbs.cookies.length = <Länge der Cookies>
```

Länge des zu jedem Eintrag vergebenen Cookies (Identifikationscode aus Buchstaben und Zahlen). Als Länge muss eine gerade Zahl angegeben werden, die Standardeinstellung sind 32 Zeichen Länge. Aus Sicherheitsgründen wird eine Länge von mindestens 8 Zeichen empfohlen.

```
cmsbs.core.verifyCookie = true|false
```

Der Cookie wird zu jedem Eintrag eindeutig vergeben, indem ein zufälliger Wert generiert und die Eindeutigkeit durch den Vergleich mit allen bereits vergebenen Cookies kontrolliert wird. Zur Optimierung ist der Vergleich in der Standardeinstellung deaktiviert, da nur bei sehr kurzen Cookies die doppelte Vergabe wahrscheinlich ist. In sicherheitsrelevanten Anwendungsfällen sollte der Vergleich aktiviert werden.

```
cmsbs.slavemode = true|false
```

Falls mehrere Instanzen des Universal Messenger parallel auf die gleiche Datenbank zugreifen sollen (z.B. zu Testzwecken oder zur Lastverteilung), muss es einen Master und mehrere Slaves geben. In den Slaves werden bestimmte Prozesse (z.B. die Ausführung der Jobs) nicht ausgeführt, die nur einmal gestartet werden dürfen.

```
cmsbs.open = <Passwort>
```

Angabe des Passworts, das beim Zugriff auf die Schnittstellen des Universal Messenger als API-Token angegeben werden muss.

Setzen des Wertes auf "{random}" bewirkt, dass beim ersten Start ein Zufallswert erzeugt und in der Datenbank gespeichert wird. Bei allen folgenden Starts wird dann der Wert aus der Datenbank verwendet.

Ab Version 7.41 sollte diese Konfigurationsvariable nicht mehr verwendet werden, sondern immer auf "{random}" gesetzt bleiben.

Die Vergabe eines neuen API-Tokens wird in [API-Token](#) beschrieben.

```
cmsbs.lang = <Spracheinstellung>
```

Sprache der Benutzeroberfläche, falls für den Benutzer keine Sprache eingestellt ist. Als Standard ist "de" vorgegeben, alternativ verfügbar ist eine englische Benutzeroberfläche mit der Einstellung "en".

```
cmsbs.template.debug = <true|false>
```

Bei `true` werden zusätzliche Debug-Ausgaben bei der Bearbeitung der Personalisierungsanweisungen beim Versand eines Newsletters ausgegeben.

```
cmsbs.core.stat.interval = <Millisekunden>
```

Optionale Angabe des Aktualisierungsintervalls bei der Berechnung der Statistiken in Millisekunden, die Standardeinstellung sind 5.000 Millisekunden.

```
cmsbs.proxy = http://benutzername:passwort@host:port/
```

HTTP(s)-Proxy für das Abholen der Newsletterinhalte von einem Webserver. Benutzername und Passwort können optional angegeben werden. `cmsbs.noproxy = host.domain1.tld .domain2.tld`

Definiert Ausnahmen für die Verwendung des in `cmsbs.proxy` angegebenen HTTP-Proxys:
Leerzeichenseparierte Liste von Hostnamen oder Domainnamen, für die kein Proxy verwendet werden soll.

Im Beispiel gilt die Ausnahme für den Host `host.domain1.tld` und für alle Hosts mit der Domain `domain2.tld`.

Zum Ändern der Verzeichnisse der CSE müssen folgende Parameter angepasst werden:

```

cmsbs.directory.cse.shared = <Pfad, default: "cmsbs-conf/cse/shared">
cmsbs.directory.cse.api = <Pfad, default: "cmsbs-conf/cse/api">
cmsbs.directory.cse.callback = <Pfad, default: "cmsbs-conf/cse/callback">
cmsbs.directory.cse.custom = <Pfad, default: "cmsbs-conf/cse/custom">
cmsbs.directory.cse.wizard = <Pfad, default: "cmsbs-conf/cse/wizard">
cmsbs.directory.cse.wizardapi = <Pfad, default: "cmsbs-conf/cse/api/wizardapi">
cmsbs.directory.cse.restapi = <Pfad, default: "cmsbs-conf/cse/api/restapi">
cmsbs.directory.cse.plugins = <Pfad, default: "cmsbs-conf/cse/plugins">
cmsbs.directory.cse.vendorplugins = <Pfad, default: "cmsbs-conf/cse/api/plugins">
cmsbs.directory.cse.pagecode = <Pfad, default: "cmsbs-conf/cse/api/jsdoc/UM-GUI">

```

`cmsbs.gui.keepSqlBeansTime = <Sekunden>`

Gibt in Sekunden an, wie lange SqlBeans gecached werden. 0 ist die Standardeinstellung und steht für unendlich.

`cmsbs.gui.keepSqlBeans = <Anzahl>`

Gibt die Anzahl der SqlBeans an, die gecached werden sollen.

`cmsbs.gui.keepAsynchronousJobsTime = <Sekunden>`

Gibt die Sekunden an, die ein laufender Job offen sein darf.

`cmsbs.gui.keepAsynchronousJobs = <Anzahl>`

Gibt die Anzahl der asynchronen Jobs an, die offen sein dürfen.

`cmsbs.gui.allowRestProxyDownload = true`

Gibt an, ob die Web-App des REST-Proxy aus der GUI heraus konfiguriert und heruntergeladen werden kann. Diese Option sollte auf *false* gesetzt werden, wenn der REST-Proxy automatisch deployt wird, so dass ein manueller Download nicht sinnvoll wäre.

`cmsbs.rest.publicUrl = <URL>`

Gibt die Standard-URL der öffentlichen REST-Schnittstelle an (z.B.: `http://www.example.org/rest`). Diese Einstellung wird insbesondere von Apps zum Generieren von JavaScript Code verwendet. An dieser Stelle darf auch ein relativer Pfad (wie z.B. `"/rest"` oder `"/p"`) anstelle der kompletten URL angegeben werden.) Die URL kann pro App-Instanz überschrieben werden.



Verwenden Sie Apps oder die REST-Schnittstelle des Universal Messenger empfehlen wir Ihnen, diese Variable zu setzen.

```
cmsbs.ssl.checks = <none|relaxed|secure>
```

Gibt an, wie der Universal Messenger als HTTP Client mit SSL-Verbindungen umgehen soll.

Wert	Beschreibung
none	Es findet keine Zertifikateprüfung statt. Die Verbindung ist verschlüsselt, aber die Echtheit der Gegenstelle wird nicht geprüft. (Standardkonfiguration)
relaxed	Die Prüfung auf Host-Namen entfällt und "self-signed" Zertifikate werden akzeptiert.
secure	Alle Prüfungen werden durchgeführt. Das Zertifikat muss gültig sein und zur Gegenstelle passen.

```
cmsbs.contentDownloader.urlAllowList = <Liste>
```

Ab dem Universal Messenger 7.43.0 muss hier eine Allow-List aller Dateipfade und URLs festgelegt werden, die beim Newsletter- bzw. Transaktionsmailversand als Inhaltsquellen für Textinhalt oder Anhänge verwendbar sein sollen.

Es können mehrere Dateipfade oder URLs durch Leerzeichen getrennt aufgezählt werden. Als Wildcards können "*" und "?" verwendet werden. Sollen sehr viele und/oder Pfade, die selbst Leerzeichen enthalten, angegeben werden, kann die Liste alternativ auch wie folgt angegeben werden:

```
cmsbs.contentDownloader.urlAllowList.01 = /data/newsletter/*
cmsbs.contentDownloader.urlAllowList.02 = /data/attachments/*
cmsbs.contentDownloader.urlAllowList.99 = http://localhost:8080/newsletter/assets/*
cmsbs.contentDownloader.urlAllowList.xx = https://www.acme.com/static/*
cmsbs.contentDownloader.urlAllowList.win01 = C:\\Users\\pinuts\\Newsletter\\*
cmsbs.contentDownloader.urlAllowList.share = \\hostname\\some-directory\\Newsletter\\*
```

(Das Suffix hinter "urlAllowList." kann beliebig gewählt werden, muss jedoch eindeutig sein. Backslashes müssen immer durch einen weiteren Backslash *geschützt* werden.)

```
cmsbs.editlock.maxtime = <Sekunden>
```

Gibt an, wie lange in Sekunden ein Eintrag zur Bearbeitung gesperrt bleibt (Default: 60). Die Bearbeitungsseite sendet im Hintergrund regelmäßig (alle 1/3 cmsbs.editlock.maxtime Sekunden) ein Signal an den Universal Messenger, damit die Sperrzeit immer wieder verlängert wird, während die Seite noch geöffnet und die Browsersitzung aktiv ist.

```
cmsbs.networkaddress.cache.ttl = <Sekunden>cmsbs.networkaddress.cache.negative.ttl =
<Sekunden>
```



Gibt an, wie lange in Sekunden ein positives bzw. negatives Ergebnis der DNS Namensauflösung durch die Java-Laufzeitumgebung zwischengespeichert wird. Beide Einstellungen werden ohne das Präfix "cmsbs." direkt an Java übergeben. Als Standardwerte sind implizit 60 bzw. 10 Sekunden gesetzt. Die genaue Bedeutung dieser beiden Optionen ist aus

<https://docs.oracle.com/javase/8/docs/technotes/guides/net/properties.html> ersichtlich.

```
cmsbs.cse.cache_size.users = <Anzahl>
```

Gibt die maximale Größe des Entry-Caches pro Thread in der Core Scripting Engine an. (Default: 512)

3.2.2 Legacy-Optionen

 Seit Version 7.38 sind die Endpunkte der alten HTTP-Schnittstelle (PHP-, Java- bzw. .NET-Connector) deaktiviert. Requests auf diese Schnittstellen werden mit Status-Code 502 *Bad Gateway* beantwortet.

Die alte Schnittstelle kann mit den unten folgenden Optionen wieder aktiviert werden:

```
# HTTP-Connector aktivieren; z.B. für Anmeldeformulare mit PHP- oder Java-Connector# Pfad:
"/cmsbs/UM"cmsbs.legacy.enableConnector.UM = true # Alte REST-Schnittstelle aktivieren; wird
sehr wahrscheinlich nicht benötigt!# Pfad:
"/cmsbs/UMService"cmsbs.legacy.enableConnector.UMService = true # Alte
Streaming-Schnittstelle aktivieren; wird sehr wahrscheinlich nicht benötigt!# Pfad:
"/cmsbs/UMStreamingService"cmsbs.legacy.enableConnector.UMStreamingService = true
```

3.3 Mehrsprachigkeit

Für die Konfiguration des Systems werden einerseits die Sprachen definiert, in denen die Benutzeroberfläche des Universal Messenger angezeigt werden soll, und andererseits die Sprachen, die für einen Eintrag ("Datensatz") in dem Universal Messenger gesetzt und gespeichert werden können. Die Benutzeroberfläche des Universal Messenger steht nur in einigen wenigen Sprachen (derzeit deutsch, englisch und französisch) zur Verfügung, da auch bei einem internationalem Einsatz des Universal Messenger für die Administratoren häufig englisch als gemeinsame Sprache verwendet wird. Die Formulare auf der Website und auch der Newsletter werden aber häufig in wesentlich mehr verschiedenen Sprachen erstellt.

Beide Einstellungen werden über ein gemeinsames Attribut definiert. Da auch Administratoren mit Zugang zur Benutzeroberfläche als ein Eintrag im Universal Messenger angelegt werden können und in diesem Eintrag folglich eine Sprache festgelegt werden kann, für die die Benutzeroberfläche des Universal Messenger nicht verfügbar ist, kann in der Konfiguration eine zusätzliche Zuordnung von den vielen auf der Website möglichen Sprachen zu den wenigen für die Benutzeroberfläche verfügbaren Sprachen definiert werden ("Fallback Lösung").

3.3.1 Definition der Sprachen in der Attributkonfiguration

Mit dem vordefinierten Attribut `lang` wird die Sprache zu einem Eintrag gespeichert. Im Abschnitt "Konfiguration des Datenmodells" ist detailliert beschrieben, wie die möglichen Werte des Attributs definiert werden können. Die folgende Konfiguration entspricht den Standardeinstellungen, die bei einer normalen Installation automatisch gesetzt werden. Durch Übernahme dieser Konfiguration in die Datei `additional.attributes` und Hinzufügen weiterer Zeilen können leicht zusätzliche Sprachen definiert werden.

```
# Standard
lang.values.0 = ""
lang.values.1 = "de"
lang.values.2 = "en"
lang.values.3 = "fr"
# Neu
lang.values.4 = "es"
```

Es ist auch möglich, anstelle der vordefinierten Werte eigene Sprachkürzel zu definieren:

```
lang.values.0 = ""
lang.values.1 = "DEU"
lang.values.2 = "ENG"
lang.values.3 = "ESP"
```

3.3.2 Abbildung der Sprachen auf die für die Benutzeroberfläche möglichen Sprachen

Wenn zum Login für interne Administratoren die Einstellung `cmsbs.gui.login = "internal"` verwendet wird und die Administratoren also als Einträge im Universal Messenger angelegt werden, müssen die für einen Eintrag möglichen Sprachen auf die für die Benutzeroberfläche des Universal Messenger möglichen Sprachen abgebildet werden. Das erfolgt in der Datei `cmsbs.properties`. Für das oben gezeigte Beispiel muss z.B. die Spanische Sprache für die Benutzeroberfläche auf Englisch abgebildet werden, weil die Benutzeroberfläche nicht in Spanisch zur Verfügung steht:

```
cmsbs.lang.de = "|de|"
cmsbs.lang.en = "|en|es|"
cmsbs.lang.fr = "|fr|"
```

Für das Beispiel mit eigenen Sprachkürzeln würde diese Einstellung in der Datei `cmsbs.properties` wie folgt gesetzt werden müssen:

```
cmsbs.lang.de = "|DEU|"
cmsbs.lang.en = "|ENG|ESP|"
cmsbs.lang.fr = ""
```

Zu jeder verfügbaren Sprache der Benutzeroberfläche werden die möglichen Sprachen des Eintrags im Multiselect-Format (Pipe-separiert, mit je einer Pipe an Anfang und Ende) angegeben. Der erste passende Wert (nach obiger Reihenfolge) bestimmt die Sprache der Benutzeroberfläche.

Sollte kein Wert passen, greift das globale Default `cmsbs.lang` aus der Konfigurationsdatei `cmsbs.properties`.

3.4 Datenbank



Der Datenbank-User des Universal Messenger muss exklusiv für diese Datenbank verwendet werden. Das bedeutet, dass die Datenbank-Credentials nur für genau diese Datenbank bzw. dieses Schema und diese Applikations-Instanz verwendet werden. Es ist nicht möglich, die selben Credentials z.B. für zwei verschiedene Entwicklungsdatenbanken zu verwenden. Die Applikation geht davon aus, nach dem Anmelden an der Datenbank in ihrem Schema zu arbeiten.

Bei SQL Server muss die Default-Datenbank des Users auf diese Datenbank gesetzt werden.

```
cmsbs.database.url = <JDBC-URL>
```

JDBC-URL zu einer SQL-Datenbank. Die folgenden Formate können verwendet werden:

MySQL	<code>jdbc:mysql://server.firma.de/cmsbs</code>
SAP DB	<code>jdbc:sapdb://server.firma.de/SAPDB</code>
Oracle	<code>jdbc:oracle:thin:@server:1521:cmsbs</code>

```
cmsbs.database.user = <Benutzername>
```

Benutzername für die SQL-Datenbank.

```
cmsbs.database.password = <Passwort>
```

Passwort für den SQL-Benutzer.

```
cmsbs.database.defaultCollation = <Collation>
```

Die Konfigurationsoption wird momentan nur bei MS-SQL Server verwendet. Möchten Sie die Collation für String-Spalten und damit den unterstützten Zeichensatz ändern, muss die Einstellung vor dem ersten Start gesetzt werden, da er nur beim Anlegen der Tabellen verwendet wird. Damit die VChannels korrekt funktionieren, muss die Collation immer "cs" enthalten, also Groß-/Kleinschreibung beachten ("case sensitive").

Der MS-SQL-Server 2000 unterstützt das Encoding UTF-8 nicht. Der MS SQL-Server 2005 unterstützt UTF-8, wenn statt `VARCHAR` `NVARCHAR` genutzt wird. Deshalb wird die Collation-Option bei der Auswahl des "MS-SQL-Servers" intern vom Default-Wert "<>" auf "latin1_general_cs_as" gesetzt.

```
cmsbs.database.useNCHAR = true|false
```

Wenn diese Option auf `true` gesetzt wird (Standardeinstellung ist `false`), werden `N[VAR]CHAR/NCLOB` anstatt `[VAR]CHAR/CLOB` genutzt.

```
cmsbs.database.schema = <Name des Schemas>
```

Mit diesem optionalen Parameter kann das Schema in der Datenbank angegeben werden, in dem der Universal Messenger die Tabellenstruktur anlegen soll. Üblicherweise bleibt der Schemaname leer oder wird gleich dem Benutzernamen gewählt; diese Option bietet eine in Spezialfällen benötigte Konfigurationsmöglichkeit. Wenn ein Schema angegeben wird, dann sollte auch die folgende Konfigurationsoption `cmsbs.database.qualifyNames` beachtet werden.

```
cmsbs.database.qualifyNames = <true|false>
```

Mit diesem optionalen Parameter kann angegeben werden, ob bei allen Datenbankzugriffen der Name des Schemas zur Qualifizierung angegeben werden soll. Diese Konfigurationsoption ist in der Standardeinstellung auf `false` gesetzt.

```
cmsbs.database.readOnlySchema = <true|false>
```

Mit diesem optionalen Parameter kann angegeben werden, dass SQL-Objekte (Tabellen, Indizes etc.) von dem Universal Messenger beim Start nicht automatisch angelegt werden, sondern dass das erwartete Datenbankschema lediglich in die mit der Konfigurationsoption `cmsbs.database.ddl` angegebene Datei (in der Standardkonfiguration `cmsbs-work/dbschema.sql`) geschrieben wird. In speziellen Anwendungsfällen können Sie die Angaben in dieser Datei verwenden, um das Datenbankschema selbst manuell einzurichten.

Diese Konfigurationsoption ist in der Standardeinstellung auf `false` gesetzt.

```
cmsbs.database.catalog = <Name des Katalogs>
```

Mit diesem optionalen Parameter kann der Katalog angegeben werden, der beim Datenbankzugriff verwendet werden soll. Üblicherweise bleibt der Name des Katalogs leer, diese Option bietet eine in Spezialfällen benötigte Konfigurationsmöglichkeit. Der Name des Katalogs wird in

`java.sql.Connection::setCatalog()` verwendet.

`cmsbs.database.ddl = <Dateiname mit Pfad>`

Beim Starten schreibt der Universal Messenger das aktuelle Datenbankschema in die angegebene Datei, in der Standardeinstellung nach `cmsbs-work/dbschema.sql`. Die Kenntnis über das Datenbankschema kann erfahrenen Administratoren bei der Systemoptimierung helfen.

```
cmsbs.database.tab.channels = <Tabelle "rights">
cmsbs.database.tab.channels_attr = <Tabelle "rights_attr">
cmsbs.database.tab.metas = <Tabelle "metas">
cmsbs.database.tab.metas_attr = <Tabelle "metas_attr">
cmsbs.database.tab.newsarchive = <Tabelle "newsarchive">
cmsbs.database.tab.newsarchive_attr = <Tabelle "newsarchive_attr">
cmsbs.database.tab.users = <Tabelle "users">
cmsbs.database.tab.users_attr = <Tabelle "users_attr">
cmsbs.database.tab.newslettercontact = <Tabelle "newslettercontact">
cmsbs.database.tab.newsletterdelivery = <Tabelle "newsletterdelivery">
cmsbs.database.tab.newsletterdelivery_attr = <Tabelle "newsletterdelivery_attr">
cmsbs.database.tab.newsletterclick = <Tabelle "newsletterclick">
cmsbs.database.tab.newsletterview = <Tabelle "newsletterview">
cmsbs.database.tab.newsletterconversion = <Tabelle "newsletterconversion">
cmsbs.database.tab.newsletterconversion_attr = <Tabelle "newsletterconversion_attr">
cmsbs.database.tab.newsletterbounce = <Tabelle "newsletterbounce">
cmsbs.database.tab.newsletterunsubscribe = <Tabelle "newsletterunsubscribe">
cmsbs.database.tab.newslettertag = <Tabelle "newslettertag">
cmsbs.database.tab.maildelivery = <Tabelle "maildelivery">
cmsbs.database.tab.mailbounce = <Tabelle "mailbounce">
```

Von dem Universal Messenger wird beim ersten Systemstart automatisch die Tabellenstruktur in der Datenbank angelegt, wobei die oben aufgeführten Tabellen mit den in Anführungszeichen genannten Namen angelegt werden. Für eine weitgehende Integration des Universal Messenger mit anderen Systemen ist es möglich, die Namen der Tabellen mit den aufgeführten optionalen Konfigurationsoptionen vorzugeben. Diese Konfigurationsoptionen sollten nur in Ausnahmefällen verwendet werden.

`db.pool.size = <Anzahl der Datenbankverbindungen>`

Der Universal Messenger öffnet zum Zugriff auf die Datenbank eine Reihe paralleler Datenbankverbindungen, deren maximale Anzahl hier optional eingestellt werden kann. Bitte beachten Sie bei der Einstellung, dass Ihre Datenbank die angegebenen gleichzeitigen Verbindungen verarbeiten kann.

Die Anzahl der Datenbankverbindungen sollte immer um eins größer sein als die maximale Anzahl Threads, mit denen der Universal Messenger im Applikationsserver ausgeführt wird.

```
db.pool.connectCode = <SQL-Code>
```

Mit dieser Konfigurationsoption kann eine mit Semikolon getrennte Liste von SQL-Kommandos angegeben werden, die beim Öffnen einer neuen Datenbankverbindung ausgeführt werden. Dies wird verwendet, um SQL Session Parameter zu setzen. Ein commit erfolgt erst, wenn die Datenbankverbindung das erste Mal genutzt wird.

```
db.pool.statementCache = <true|false>
```

Gibt an, ob ein Statement Cache verwendet werden soll. In der Standardeinstellung ist der Statement Cache für alle Datenbanken bis auf MySQL und Oracle aktiviert, bei denen der JDBC-Treiber zu viel Speicherplatz belegt.

```
db.pool.maxStatementReUse = <max. Nutzungen pro prepared Statement>
```

Gibt an, wie häufig ein bereits im Cache vorhandenes prepared Statement verwendet werden soll, bevor es verworfen wird. Die Standardeinstellung beträgt 1.000 für alle Datenbanken bis auf die Oracle, bei der keine prepared Statements verwendet werden.

```
db.pool.maxProfiles = <max. Anzahl Statements im SQL Profiler>
```

Gibt die maximale Anzahl der Statements für den SQL Profiler an. Die Standardeinstellung beträgt 100.

```
db.pool.maxUseTime = <max. Anzahl Sekunden pro DB Connection>
```

Gibt die maximale Anzahl der Sekunden für eine DB Connection an. Die Standardeinstellung beträgt 120.

```
cmsbs.database.user.keycolSize = <Anzahl in Byte>
```

Breite der `keycol`-Spalte der Datenbanktabelle `users_attr`, mit der die maximale Länge der internen Namen für zusätzliche Benutzerattribute definiert wird. In der Standardkonfiguration sind 128 Bytes definiert.

```
cmsbs.database.user.valcolSize = <Anzahl in Byte>
```

Breite der `valcol`-Spalte der Datenbanktabelle `users_attr`, mit der die maximale Länge der Werte in zusätzlichen Benutzerattributen definiert wird. Falls bei der Konfiguration von Attributen vom Typ `TABLE` keine Spaltenbreite mit `dbSize` angegeben wird, wird ebenfalls diese Einstellung übernommen. In der Standardkonfiguration sind 512 Bytes definiert.

```
cmsbs.database.batchLoad = true|false
```

Mit dieser Konfigurationsoption kann der Zugriff auf die Datenbank kontrolliert werden, so dass das effiziente Laden mehrerer Objekte in einem Abruf aktiviert wird. In der Standardkonfiguration ist diese Einstellung aktiviert, bei Problemen mit der Datenbank ist die Deaktivierung möglich.

```
cmsbs.database.maxParams = <Anzahl Objekte>
```

Mit dieser Konfigurationsoption kann die maximale Objektanzahl angegeben werden, die beim Laden mehrerer Objekte zusammengefasst geladen werden. In der Standardkonfiguration ist diese Einstellung auf -1 gesetzt, womit ein Standardwert von 1.000 Objekten geladen wird.

```
cmsbs.database.readSize = <Anzahl der Zeilen>
```

Anzahl der Zeilen, die von Hilfsprogrammen (CopyTool) in einem Durchlauf gelesen werden. Die Standardeinstellung beträgt 100 Zeilen.

```
cmsbs.database.copyFifo = <Anzahl der Datenpakete>
```

Anzahl der Datenpakete, die von Hilfsprogrammen (CopyTool) im Speicher gehalten werden, um parallel auf die Quelle und das Ziel zugreifen zu können. cmsbs.database.copySize = <Anzahl der Zeilen>

Anzahl der Zeilen, die von Hilfsprogrammen (CopyTool) in einer Aktion kopiert werden. Die Standardeinstellung beträgt 10 Zeilen und sollte bei Unverträglichkeit mit der Datenbank bzw. JDBC-Treiber auf 1 gesetzt werden.

```
cmsbs.database.writeSize = <Anzahl der Zeilen>
```

Anzahl der Zeilen, die von Hilfsprogrammen für den Import zusammengefasst werden, bevor ein commit ausgeführt wird. Die Standardeinstellung beträgt 1.000 Zeilen.

```
db.query.debug = true|false
```

Zu Testzwecken können interne Informationen bei der Zusammenstellung von Abfragen auf Abonentendaten in die Logdatei ausgegeben werden.

```
db.pool.debug = true|false
```

Zu Testzwecken können interne Informationen zu den parallelen Datenbankverbindungen in die Logdatei ausgegeben werden.

```
db.pools = <Liste der Poolnamen>
```

Um mehrere Datenbankpools in der CSE verwenden zu können, müssen Sie diese Einstellung verwenden. Mehr Informationen zur Konfiguration finden Sie in der CSE Dokumentation.

```
db.lock.maxtime = <max. Anzahl der Sekunden>
```

Gibt die maximale Anzahl der Sekunden an, die eine Sperrung gehalten wird (LRU limit). Default-Wert: 600

```
db.lock.maxwait = <max. Anzahl der Sekunden>
```

Gibt die maximale Anzahl der Sekunden an, die auf eine erfolgreiche Sperrung gewartet wird. Defaultwert: 5

```
db.lock.global = <Globale Lock-Einstellung>
```

0 = off; 1 = on; 2 = DB-spezifisch (Default-Einstellung)

```
db.mysql.binarycast = true|false
```

Wenn die Konfigurationsoption auf `true` gesetzt ist, erfolgen Vergleiche nicht bezüglich der Collation sondern binärer Gleichheit. Zu beachten ist, dass dadurch Indizes nicht benutzt werden. Der Schalter sollte auf `false` gesetzt werden, wenn die Default-Collation in der MySQL richtig gesetzt ist.

```
db.pool.mysqlCloseCheck = true|false
```

Da MySQL nicht immer korrekt angibt, ob eine Verbindung geschlossen ist (d.h. sie gibt an, eine Verbindung sei geschlossen, wobei sie noch offen ist), wird mit der Standardeinstellung `true` definitiv geprüft, ob sie geschlossen ist.

```
cmsbs.database.csvBulksize = <Anzahl der Einträge>
```

Gibt an, wie viele Einträge bei einem CSV-Import gleichzeitig in die Datenbank geschrieben werden. Die Standardeinstellung ist 100. Bei der Definition der `map`-Datei kann dieser Wert mit `import.bulksize` überschrieben werden.

```
cmsbs.database.maildelivery.a1.length = <Spaltengröße, default: 32>
```

```
cmsbs.database.maildelivery.a2.length = <Spaltengröße, default: 64>
```

```
cmsbs.database.maildelivery.a3.length = <Spaltengröße, default: 128>
```

Diese Werte bestimmen die Größe der Spalten der Maildelivery-Attribute in der `newsletterdelivery` Tabelle. Alle weiteren Maildelivery-Attribute werden unter `newsletterdelivery_attr` gespeichert.

```
cmsbs.database.readOnly = true|false
```

Mit dieser Option kann global verhindert werden, dass die Datenbank vom Universal Messenger beschrieben wird.

Der Universal Messenger lädt die Listen für (V)Channels, Jobs und Newsletter-Tags hauptsächlich asynchron nach. Die Defaulteinstellung sehen hierfür wie folgt aus:


```

cmsbs.database.ChannelCache.async = true|false (default: true)
cmsbs.database.ChannelCache.ttl = <Sekunden, default: 600>
cmsbs.database.VChannelCache.async = true|false (default: true)
cmsbs.database.VChannelCache.ttl = <Sekunden, default: 605>
cmsbs.database.CronJobCache.async = true|false (default: false)
cmsbs.database.CronJobCache.ttl = <Sekunden, default: 120>
cmsbs.database.NewsletterTagCache.async = true|false (default: true)
cmsbs.database.NewsletterTagCache.ttl = <Sekunden, default: 3600>

```

Im Clusterbetrieb sollten die TTLs ggf. runtergesetzt werden.

```
cmsbs.database.maxNamedQueries = <max. Anzahl Segmente pro SQL-Query>
```

Gibt an, wie viele Segmentzugehörigkeiten in einem einzigen SQL-Statement geprüft werden sollen. Default: 100.

Sind mehr Segmente vorhanden, werden sie bei der Zugehörigkeitsprüfung entsprechend dieses Wertes auf mehrere SQL-Statements verteilt.

3.5 Logging

```
cmsbs.log.syslog = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Logdatei, in der Systemmeldungen des Universal Messenger protokolliert werden sollen.

```
cmsbs.log.syslog.rotate = true|false
```

Bei `true` wird jeweils nächtlich die Logdatei rotiert und die vorhergehende Version mit dem Datum im Dateinamen archiviert. Bei `false` werden neue Einträge an eine bestehende Logdatei angehängt.

```
cmsbs.log.syslog.last = <Anzahl>
```

Anzahl der Einträge aus der Logdatei, die in der Benutzeroberfläche mit dem Menüpunkt "Extras - Log-Datei" angezeigt werden sollen. In der Standardkonfiguration werden 1.000 Zeilen der Logdatei angezeigt.

```
cmsbs.log.userlog = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Logdatei, in der Änderungen der Abonentendaten (z.B. Neuansmeldung, Änderungen) über die Benutzeroberfläche des Universal Messenger und die Zugriffe über die Schnittstellen protokolliert werden sollen. In der Datei wird außerdem die Ausführungszeit der internen Requestverarbeitung in Millisekunden festgehalten.

```
cmsbs.log.userlog.rotate = true|false
```

Bei `true` wird jeweils nächtlich die Logdatei rotiert und die vorhergehende Version mit dem Datum im Dateinamen archiviert. Bei `false` werden neue Einträge an eine bestehende Logdatei angehängt.

```
cmsbs.log.maillog = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Logdatei, in der alle von dem Universal Messenger verschickten E-Mails mit Ausnahme des Newsletterversands protokolliert werden soll. Die Protokollierung erfolgt nur, wenn die angegebene Datei existiert. Um die Protokollierung zu aktivieren, erzeugen Sie also ggf. eine leere Datei, um die Protokollierung zu stoppen, löschen Sie die angegebene Datei.

```
cmsbs.log.maillog.news = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Logdatei, in der alle von dem Universal Messenger E-Mail verschickten Newsletter protokolliert werden soll. Die Protokollierung erfolgt nur, wenn die angegebene Datei existiert. Um die Protokollierung zu aktivieren, erzeugen Sie also ggf. eine leere Datei, um die Protokollierung zu stoppen, löschen Sie die angegebene Datei. In der Standardkonfiguration ist die Datei

`cmsbs-work/mail-news-out.log` als Logdatei vorgegeben.

```
cmsbs.log.cse = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Logdatei, in der alle von der CSE erzeugten Log-Ausgaben protokolliert werden. Default-Wert: `"cmsbs-work/cse.log"`. Die Logdatei wird nächtlich automatisch rotiert.

```
cmsbs.log.allFilesToStdout = true|false
```

Bei `true` werden alle Logausgaben zusätzlich nach stdout geschrieben. Das ist vor allem beim Betrieb des Universal Messenger in einem Docker-Container oder im Development-Modus mit integriertem Tomcat sinnvoll.

3.5.1 History-Log

```
cmsbs.history = true|false
```

Konfigurationsoption, um das History-Log des Universal Messenger anzuschalten.

```
cmsbs.history.retention.$area.$action = <Zeitangabe in Stunden>
```

Retention Policy für das History-Log gibt an wie lange ein Eintrag eines bestimmten Bereiches und eines bestimmten Aktionstyps im Log vorgehalten werden soll. Die Angabe einer Retention Policy ist optional.

3.5.2 Debugging

```
cmsbs.debug = true|false
```

Konfigurationsoption, um den Debug-Modus des Universal Messenger anzuschalten.

```
cmsbs.smtp.debug = true|false
```

Konfigurationsoption, um den SMTP Debug-Modus anzuschalten.

3.5.3 Transaktions-Logging

```
cmsbs.txnlog = true|false
```

```
cmsbs.txnlog.dir = <Dateiname mit Pfad>
```

Sollen sämtliche schreibenden Transaktionen protokolliert werden, muss `cmsbs.txnlog` auf `true` gesetzt werden. Die Log-Dateien im XML-Format werden daraufhin im Verzeichnis `cmsbs-work/txnLog` gespeichert. Sollen sie woanders liegen, muss der Wert von `cmsbs.txnlog.dir` angepasst werden.

3.6 Versandeinstellungen

3.6.1 Allgemeine Versandoptionen

```
cmsbs.event.noMail = <Abfrage>
```

Falls eine Abfrage angegeben ist, werden beim Versand eines E-Mail-Newsletters die gefundenen Abonnenten übergangen, d.h. es werden keine Newsletter an diese Abonnenten versendet, Benachrichtigungen werden jedoch weiterhin versendet.

Mit dieser Option können z.B. Abonnenten ab einer bestimmten Anzahl an Bounces vom Newsletter-Versand ausgenommen werden, während alle anderen Benachrichtigungen weiterhin versendet werden.

3.6.2 E-Mail Versand

```
cmsbs.mail.smtpserver = <Hostname oder IP-Adresse oder none>
```

Der für sämtlichen Mailversand zu benutzende SMTP-Mailserver.

Die Option `"none"` bzw. `" "` kann für Testzwecke genutzt werden, die E-Mails werden lediglich in den Logfiles protokolliert, aber nicht versendet. Der Versand wird erst so spät unterdrückt, dass alle Performance-Einstellungen (bzgl. Threads und Delays) und Callbacks beachtet werden. Komplexe Szenarien können auf diesem Weg getestet werden, ohne dass echte E-Mails versendet werden müssen. Mit dieser Option lassen sich auch einzelne [Mail-Relays](#) sinnvoll "ab-" und wieder "hinzuschalten".

```
cmsbs.mail.ssl = true | tls | false
```

Zugriff auf der Sever per SSL (`true`), TLS/STARTLS (`tls`) oder unverschlüsselt (`false`).

```
cmsbs.mail.smtpauth = true|false
```

Falls der für den Mailversand zu benutzende SMTP-Mailserver eine Authentifizierung benötigt, kann diese Anmeldung mit `true` aktiviert werden.

```
cmsbs.mail.smtpuser = <Benutzername>
```

Der Benutzer, über den sich der Universal Messenger zum Mailversand am SMTP-Server anmelden soll.

```
cmsbs.mail.smtppassword = <Passwort>
```

Das Passwort des Benutzers, über den sich der Universal Messenger zum Mailversand am SMTP-Server anmelden soll.

```
cmsbs.smtp.maxRetries = <Anzahl>
```

Anzahl der Wiederholungen, wenn der Versand aufgrund von Netzwerkproblemen fehlgeschlagen ist. (Default: 10)

```
cmsbs.smtp.retryDelay = <Sekunden>
```

Wartezeit in Sekunden vor jedem Retry (siehe auch `cmsbs.smtp.retryDelay`; Default: 30)

```
cmsbs.mail.plain.encoding = <Encoding>
```

Definition der Kodierung der Text-E-Mails und der Textabschnitte in MIME Multipart-Messages, die möglichen Werte sind abhängig von Betriebssystem und Java Virtual Machine, z.B. iso-8859-1, iso-8859-15, UTF-8, UTF-16 oder cp-1252. Die Standardeinstellung entspricht der systemweiten Standardeinstellung.

```
cmsbs.mail.html.encoding = <Encoding>
```

Definition der Kodierung der HTML-E-Mails. Die möglichen Werte sind abhängig von Betriebssystem und Java Virtual Machine, z.B. iso-8859-1, iso-8859-15, UTF-8, UTF-16 oder cp-1252. Die Standardeinstellung entspricht der systemweiten Standardeinstellung.

```
cmsbs.mail.file.encoding = <Encoding>
```

Definition der Kodierung der als Attachment angehängten Text-Dateien. Die möglichen Werte sind abhängig von Betriebssystem und Java Virtual Machine, z.B. iso-8859-1, iso-8859-15, UTF-8, UTF-16 oder cp-1252. Die Standardeinstellung entspricht `cmsbs.mail.plain.encoding`.

```
cmsbs.url.encoding = <Encoding>
```

Definition der Kodierung der im Newsletter enthaltenen URLs. Die Standardeinstellung ist `UTF-8` und sollte möglichst nicht verändert werden.

```
cmsbs.mail.threads = <Anzahl der Versandprozesse>
```

Der Mailversand erfolgt parallel über die hier angegebene Anzahl an Versandprozessen.

```
cmsbs.mail.delay = <Verzögerung in Millisekunden>
```

Über diesen Parameter kann eine Verzögerung in Millisekunden definiert werden, die jeder Versandprozess nach dem Versand einer E-Mail dauert. Diese Verzögerung kann notwendig sein, um den SMTP-Server nicht zu überlasten.

```
cmsbs.mail.signature = <Pfad und Datei>
```

Der Text in dieser Datei wird in der Benutzeroberfläche im Formular zum Versenden eines Newsletters als Vorlage übernommen und kann an einen Newsletter angefügt werden. Üblicherweise enthält die Signatur Angaben zum Absender des Newsletters und Hinweise zum Abbestellen oder Ändern des Abonnements. In der Signatur können Personalisierungsanweisungen verwendet werden.

Es wird empfohlen, dass Signaturen mit der Zeile `--` (Minus-Minus-Leerzeichen-Zeilenvorschub) beginnen, da sie dann von einigen Mailprogrammen automatisch erkannt werden können.

```
cmsbs.mail.to = <Personalisierungstext>
```

Beim Versand einer E-Mail kann neben der E-Mail-Adresse des Empfängers ein Klartextname angegeben werden. In der Standardkonfiguration werden der Vorname und Nachname des Empfängers eingetragen oder alternativ der Firmenname, falls nur dieser bekannt ist. Als E-Mail-Adresse wird der Wert aus dem Attribut `email` verwendet. Wenn der Newsletter an ein anderes E-Mail-Attribut geschickt werden sollen, kann dies ebenfalls angepasst werden.

Das Format zur Angabe des Empfängers in der E-Mail kann mit dieser Konfigurationsoption geändert werden. Der Personalisierungstext muss zu einer konformen Adresse gemäß RFC822 führen, z.B. "Hans Meier <hans@meier.com>". Zur Formatierung kann die Formatierungsanweisung `makeEmailAddress` verwendet werden.

```
cmsbs.mail.messageId.hostname
```

Ermöglicht das Überschreiben des Hostnamen-Anteils in den Message-IDs, welche der Universal Messenger beim Mailversand generiert.

Format der Message-ID: `UM.<delivery_ticket>@<hostname>`

delivery_ticket ist das Delivery-Ticket der ausgehenden Mail.

hostname ist per Default der in `cmsbs.mail.smtpserver` angegebene Name bzw. die IP-Adresse des angesprochenen SMTP-Servers. Dieser Anteil kann in `cmsbs.mail.messageId.hostname` überschrieben werden.

```
cmsbs.mail.check_links = true|false
```

Syntaktische Prüfung aller Links in den href-Attributen in der Element `a` und `area` vor dem Versand. Werden defekte Links gefunden, wird der Versand abgebrochen. Die Voreinstellung ist `false`.

Standard-Konfiguration:

```
cmsbs.mail.to = {if:defined(firstname) && defined(lastname):{makeEmailAddress|{firstname}
{lastname}}|{email}}:elif:defined(company):{makeEmailAddress|{company}|{email}}:else:{email}}
```

Zu Testzwecken können mit dieser Konfigurationsoption alle ausgehenden E-Mails an eine Testadresse umgeleitet werden, indem in die Konfigurationsdatei folgende Einstellung aufgenommen wird:

```
cmsbs.mail.to = "{makeEmailAddress:Testadresse:test@aol.com}"
```

Wenn ein Personalisierungstext wie hier beschrieben zum Erzeugen der E-Mail-Adresse angegeben wird, kann als Ergebnis eine leere oder ungültige E-Mail-Adresse erzeugt werden, so dass diese Option sorgfältig eingesetzt werden muss.

```
cmsbs.mail.sendername = <eMail Absender Name>
```

Dieser Klartextname wird bei allen verschickten E-Mails als Absender eingetragen.

```
cmsbs.mail.senderaddress = <eMail Adresse>
```

Diese Adresse wird bei allen verschickten E-Mails als Absender eingetragen. Es muss die E-Mail-Adresse angegeben werden, unter der die Bounces empfangen werden sollen und über den IMAP-Account

`cmsbs.mail.imap.user` abgerufen werden können.

```
cmsbs.mail.replyto = <eMail Adresse>
```

Diese Adresse wird bei allen verschickten E-Mails als Reply-To Adresse eingetragen. Durch Angabe dieser Adresse können Sie Bounces von persönlichen Antworten trennen, da die meisten Mailprogramme eine persönliche Antwort an die Reply-To Adresse versenden.

```
cmsbs.bounce.verp.envelopeFrom = "um-%0@SERVER.de"
```

Um VERP für den Versand zu aktivieren, muss die Konfigurationsvariable `cmsbs.bounce.verp.envelopeFrom` auf die gewünschte envelopeFrom-Adresse gesetzt werden. Der Platzhalter "%0" wird jeweils durch eine individuell pro E-Mail generierte ID ersetzt und bildet damit die individuelle Envelope-From-Adresse. Mehr zum Thema VERP finden Sie unter [Bounce Management](#).

```
cmsbs.template.metatag.contentType = remove|set|fix
```

Gibt an, wie mit einem vorhandenen Meta-Tag mit Angabe des Content-Types umgegangen wird.

- "remove" (Default): Bisheriges Verhalten, Meta-Tag wird immer entfernt.
- "set": Wert im vorhandenen Meta-Tag wird ersetzt durch "text/html; charset=[CHARSET]".
- "fix": Charset-Angabe im vorhandenen Meta-Tag wird ersetzt, der Mime-Type bleibt erhalten.

```
cmsbs.smtp.maxIdleTime = <Sekunden>
```

Gibt in Sekunden an, wie lange eine Verbindung im "Leerlauf" sein darf.

```
cmsbs.smtp.maxUseTime = <Sekunden>
```

Gibt in Sekunden an, wie lange eine einzelne Verbindung offen sein darf.

```
cmsbs.smtp.maxMails = <Anzahl E-Mails>
```

Gibt die maximale Anzahl der E-Mails an, die über einen Transport gehen sollen.

```
cmsbs.mail.checkIdleTime = <Sekunden>
```

Gibt das Intervall in Sekunden an, in dem eine Anfrage an eine SMTP-Verbindung im "Leerlauf" gesendet wird.

```
cmsbs.mail.restUrl = <URL>
```

Gibt die URL der öffentlichen REST-Schnittstelle (z.B.: <http://www.example.org/rest>) zum Nachladen von Assets in HTML-Mails an. Diese URL kann pro Mailingvorlage überschrieben werden.

```
cmsbs.mail.baseUrl = <URL>
```

Gibt die Base-URL der Website an. Wird als Default für die *Base-URL* neu angelegter Mailingvorlagen verwendet. Dazu wird diese URL allen relativen Links einer ausgehenden HTML-E-Mail vorangestellt. Kann in jeder Mailingvorlage überschrieben werden.

```
cmsbs.mail.guiUrl = <URL>
```

Gibt die Base-URL des Backoffice an. Wird u.a. in internen Benachrichtungen der Service Desk App verwendet, um Deep-Links auf Tickets zu ermöglichen.

```
cmsbs.quota.outgoingMailsPerMonth = <max. Anzahl pro Monat>
```

Gibt an, dass pro Kalendermonat nicht mehr als eine bestimmte Anzahl an Mail versendet werden soll. Ist diese Option gesetzt, erfolgt eine Fehlermeldung in der Logdatei ("sys.log"), falls um Mitternacht eine Überschreitung für den laufenden Monat festgestellt wurde. Eine tatsächliche Beschränkung des E-Mailversand findet jedoch nicht statt.

Darüber hinaus wird bei gesetzter Option unter *Extras / System* der aktuelle Stand und der erlaubte monatlich Wert angezeigt:

Quotas	
Ausgehende E-Mails im laufenden Monat	51.479 / 50.000 

Der Link führt auf ein Diagramm mit den täglichen Versandzahlen.

Dieses Feature ist i.A. für on-premise-Installationen nicht relevant, da dort üblicherweise keine Beschränkung der Anzahl der versendeten E-Mails gewünscht ist.

3.6.2.1 Zusätzliche Mail-Header-Elemente

Die Konfigurationsdatei `cmsbs.properties` kann um zusätzliche Mail-Header-Elemente erweitert werden, die automatisch mit jeder E-Mail und speziell für einzelne Newsletter und Benachrichtigungen versendet werden.

Davon sind jedoch die folgenden Header ausgeschlossen:

- From
- Reply-To

Variablen zur Personalisierung können hierbei verwendet werden.

Das folgende Beispiel konfiguriert fünf neue Mail-Header-Elemente:

```
cmsbs.mail.headers = "List-Id List-Unsubscribe List-Owner List-Help List-Archive"
cmsbs.mail.header.List-Id = "Newsletter" <newsletter.KUNDE.de>
cmsbs.mail.header.List-Unsubscribe = "<http://www.KUNDE.de/unsubscribe.php?c={msgid}>"
cmsbs.mail.header.List-Owner = "<mailto:support@KUNDE.de>"
cmsbs.mail.header.List-Help = "<http://www.KUNDE.de/help>"
cmsbs.mail.header.List-Archive = "<http://www.KUNDE.de/archive>"
```

Eine versendete E-Mail enthält damit folgende zusätzliche Zeilen im Header:

```
List-Help: <http://www.KUNDE.de/help>
List-Unsubscribe:
<http://www.KUNDE.de/unsubscribe.php?c=4ZYY.508X.EF529E6E8A8CC6B1D9D78805F7A14AF4>
List-Owner: <mailto:support@KUNDE.de>
List-Archive: <http://www.KUNDE.de/archive>
List-Id: "Newsletter" <newsletter.KUNDE.de>
```


Mail-Header-Elemente pro Newsletter

Die für den Newsletter-Versand verwendete XML-Event-Datei kann ebenfalls Mail-Header-Elemente definieren. Dies ermöglicht eine Trennung zwischen Header-Elementen für alle E-Mails und weiteren Header-Elementen, die nur für Newsletter verwendet werden sollen.

Das folgende Beispiel konfiguriert ein neues Mail-Header-Element:

```
<event>
  <destination>
    ...
  </destination>
  <data>
    ...
  </data>
  <global
header="List-Unsubscribe">&lt;http://www.KUNDE.de/unsubscribe.php?c={msgid}&gt;</global>
</event>
```

Werden in der XML-Event-Datei die gleichen Header-Elemente definiert, die auch in der Konfigurationsdatei `cmsbs.properties` definiert wurden, so werden die Werte der XML-Event-Datei bevorzugt.

Zusätzliche Mail-Header mittels CSE verwenden

Das folgende Beispiel konfiguriert ein neues Header-Element beim Versand einer Benachrichtigung mittels der CSE:

```
var r2 = UM.getEntryByUid('12345');
var n = r2.notifications.add('newuser');
n.mailHeaders['List-Unsubscribe'] = '<http://www.KUNDE.de/unsubscribe.php?c={uid}>';
UM.commitEntries();
```

3.6.2.2 Archivierung ausgehender E-Mails

```
cmsbs.mail.outlog.transaction = true|false
```

Gibt an, wie die Vorbelegung der Einstellung *Archivierung ausgehender E-Mails* bei Erstellung einer neuen Transaktionsmail in der Verwaltung der Mailingvorlagen ist. Die Einstellung kann in der Vorlage der Transaktionsmail individuell angepasst werden. Die Voreinstellung ist `false`.

```
cmsbs.mail.outlog.newsletter = true|false
```

Gibt an, wie die Vorbelegung der Einstellung *Archivierung ausgehender E-Mails* bei Erstellung eines neuen Newsletter in der Verwaltung der Mailingvorlagen ist. Die Einstellung kann in der Vorlage des Newsletters individuell angepasst werden. Die Voreinstellung ist `false`.

3.6.2.3 Versand per Amazon SES

Ausgehende E-Mails können auch über den Simple E-Mail Service der Amazon Web Services verschickt werden. Dazu wird ein spezieller SMTP-Server angegeben:

```
cmsbs.mail.smtpserver = AmazonSES:us-east-1
```

Dabei gibt der Teil nach `AmazonSES`: die AWS-Region an (derzeit ist nur `us-east-1` möglich), die angesprochen werden soll. Die Regionen sind in `<UM_HOME>/cmsbs-conf/aws-regions.xml` definiert. Diese Datei ist Teil des AWS SDKs und muss normalerweise nicht manuell bearbeitet werden. Sie wird beim Update überschrieben.

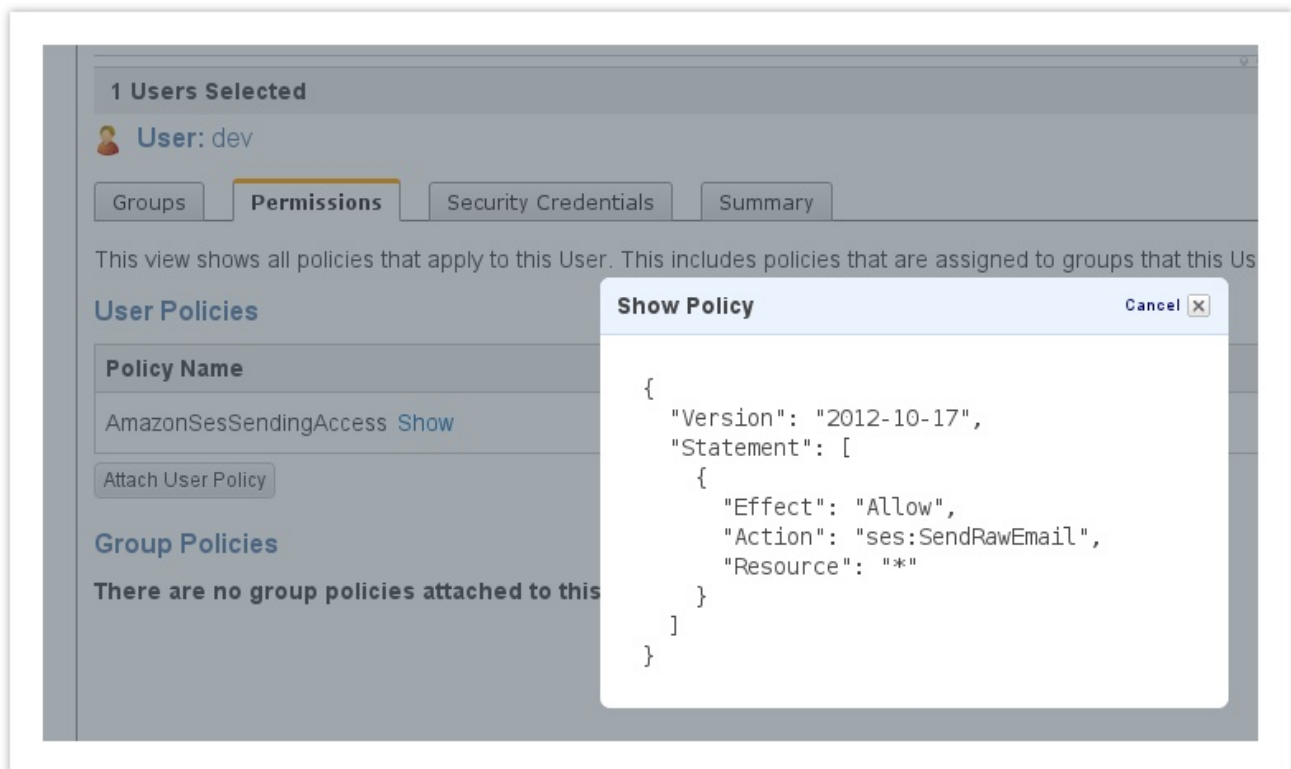
Sollen andere Definitionen verwendet werden, ist die Konfigurationseinstellung `cmsbs.aws.regions` auf einen anderen Dateinamen zu setzen.

Für die minimale Konfiguration als einziges E-Mail-Relay sind nur noch User und Passwort anzugeben:

```
cmsbs.mail.smtpuser = <Benutzername>
```

```
cmsbs.mail.smtppassword = <Passwort>
```

Diese Credentials werden im IAM der AWS-Console vergeben. Die zugehörige Policy muss **ses:SendRawEmail** enthalten (siehe Bild).



Die folgenden Kopiervorlagen enthalten weitere detaillierte Konfigurationseinstellungen. Soll zum Versand ausschließlich die AWS-Anbindung zum Einsatz kommen, sind die folgenden Einstellungen relevant.

Als Standard-Relay

```

cmsbs.mail.smtpserver = AmazonSES:us-east-1
cmsbs.mail.smtpuser = User
cmsbs.mail.smtppassword = Pass

# Use this proxy for connecting
cmsbs.mail.aws.proxy = http://proxyuser:proxypass@proxyhost:1234/

# Technical details, usually left to default
cmsbs.mail.aws.connectionTimeout = 50000
cmsbs.mail.aws.socketTimeout = 50000
cmsbs.mail.aws.maxErrorRetry = 3

# Use "http" to force plain protocol
cmsbs.mail.aws.protocol = https

# Change the actual URL to communicate to
cmsbs.mail.aws.endpoint = http://debughost/behaves/like/AWS

# (Standard performance features)
#cmsbs.mail.threads = 1
#cmsbs.mail.delay = 0
#cmsbs.mail.checkIdleTime = 10
#cmsbs.smtp.maxIdleTime = 30
#cmsbs.smtp.maxUseTime = 1800
#cmsbs.smtp.maxMails = 10

```


Soll die AWS-Anbindung als weiteres Relay verwendet werden, kommen folgende Einstellungen zum Einsatz:

Als weiteres Relay


```

cmsbs.mail.relays = XXX
cmsbs.mail.relay.XXX.smtpserver = AmazonSES:us-east-1
cmsbs.mail.relay.XXX.smtpuser = User
cmsbs.mail.relay.XXX.smtppassword = Pass
cmsbs.mail.relay.XXX.cmsbs.mail.aws.proxy = http://proxyuser:proxypass@proxyhost:1234/
cmsbs.mail.relay.XXX.cmsbs.mail.aws.connectionTimeout = 50000
cmsbs.mail.relay.XXX.cmsbs.mail.aws.socketTimeout = 50000
cmsbs.mail.relay.XXX.cmsbs.mail.aws.maxErrorRetry = 3
cmsbs.mail.relay.XXX.cmsbs.mail.aws.protocol = https
cmsbs.mail.relay.XXX.cmsbs.mail.aws.endpoint = http://debughost/behaves/like/AWS
cmsbs.mail.relay.XXX.maxIdleTime
cmsbs.mail.relay.XXX.maxUseTime
cmsbs.mail.relay.XXX.maxMails
cmsbs.mail.relay.XXX.delay
cmsbs.mail.relay.XXX.threads
cmsbs.mail.relay.XXX.checkIdleTime

```

 Bei der Verwendung von Amazons Simple E-Mail Service besteht die Wahl zwischen normalem SMTP und einem nativen API-Aufruf. Diese beiden Schnittstellen sind äquivalent. Der Vorteil der API-Methode ist jedoch, dass sie über normales HTTP abgewickelt wird, wodurch es möglich ist, einen Proxy zwischenschalten.

3.6.2.4 Versand via Inxmail

 Wir empfehlen, auch für den Versand per Inxmail das Standardprotokoll SMTP zu verwenden.

Soll der Versand aber trotzdem per Inxmail-API erfolgen, müssen ab der Universal Messenger 7.50 die folgenden JAR-Dateien nach erfolgter Installation unter `UM/cmsbs/WEB-INF/lib/` abgelegt werden:

- `axis.jar`
- `axis-jaxrpc.jar`
- `axis-saaj.jar`
- `hessian.jar`
- `inxmail-apiimpl.jar`
- `wsdl4.jar`

Bei Bedarf können die genannten Dateien durch den Support zur Verfügung gestellt werden.

E-Mails ohne Attachments in Plain-Text und/oder HTML können auch via Inxmail versendet werden. Für die minimale Konfiguration als einziges E-Mail-Relay sind nur SMTP-Server, Benutzername und Passwort (Pflichtfelder) anzugeben.

```
cmsbs.mail.smtpserver = Inxmail:login.inxmail.com/<INXMAIL_MANDANT> cmsbs.mail.smtpuser =
<Benutzername>cmsbs.mail.smtppassword = <Passwort>
```

Die folgenden beiden Konfigurationen sind optional:

```
cmsbs.inxmail.proto = hessian | hessians | http | https
```

Die Werte `hessian` und `hessians` stehen für ein schnelles, binäres Protokoll. Die Werte `http` und `https` steht für SOAP. Die Varianten mit "s" beinhalten eine SSL-Verschlüsselung auf dem Transportweg. Per Default wird "hessians" verwendet, da es schnell und sicher ist. Für Debugzwecke kann "http" interessant sein.

```
cmsbs.inxmail.listContext = <Listenname>
```

Name der Inxmail-Liste, deren Absender-Adresse verwendet werden soll. Per Default wird die globale "system"-Liste verwendet.



Beim Versand via Inxmail ist insbesondere zu beachten, dass der Versand von Attachments und eingebetteten Grafiken nicht möglich ist.

3.6.2.5 Versand via SendGrid

SendGrid ist ein Dienstleister (Software as a Service, SaaS), der auf den Versand von E-Mails und die Behandlung damit verbundener Events wie Bounces, Clicks und Views spezialisiert ist. SendGrid betreibt eine Cloud-basierte E-Mail Versandinfrastruktur, mit der auch große Versandmengen performant und zuverlässig ausgeliefert werden können. Für die vertragliche Abwicklung wenden Sie sich bitte an unseren Vertrieb, in diesem Abschnitt wird lediglich die technische Konfiguration erläutert.

Die Verwendung von SendGrid in Zusammenhang mit Dem Universal Messenger erspart sowohl den Betrieb eines eigenen SMTP-Servers für den Versand als auch die Einrichtung von [VERP](#) für die zuverlässige Bounce-Behandlung. Darüber hinaus gewährleistet SendGrid eine gute Zustellbarkeit durch eine konstante Überwachung des Mailverkehrs, von Blacklists und der eigenen Reputation.

Konfiguration im Universal Messenger

Im Universal Messenger wird der SMTP-Server von SendGrid in der `cmsbs.properties`-Datei eingetragen:

```
cmsbs.mail.smtpserver = "smtp.sendgrid.net"
cmsbs.mail.smtpauth = "true"
cmsbs.mail.smtpuser = "[[SENDGRID-USERNAME]]"
cmsbs.mail.smtppassword = "[[SENDGRID-PASSWORT]]"
```

Damit SendGrid auch das Tracken von Bounces und optional von Klicks und Öffnungen übernehmen kann, muss die `msgid` als Mail-Header mitgesendet werden. Die folgenden Konfigurationszeilen sorgen dafür, dass in jeder ausgehende E-Mail der Header `X-SMTPAPI` entsprechend gesetzt wird:

```
cmsbs.mail.headers = "X-SMTPAPI"
cmsbs.mail.header.X-SMTPAPI = "{msgid:sendgrid}"
```

Zur Einrichtung der Bounce-Behandlung siehe [Bounce Management mit SendGrid](#).

3.6.2.6 Versand via Mailjet

Mailjet ist ein Dienstleister (Software as a Service, SaaS), der auf den Versand von E-Mails und die Behandlung damit verbundener Events wie Bounces, Clicks und Views spezialisiert ist. Mailjet betreibt eine Cloud-basierte E-Mail Versandinfrastruktur, mit der auch große Versandmengen performant und zuverlässig ausgeliefert werden können. Für die vertragliche Abwicklung wenden Sie sich bitte an unseren Vertrieb, in diesem Abschnitt wird lediglich die technische Konfiguration erläutert.

Die Verwendung von Mailjet in Zusammenhang mit dem Universal Messenger erspart sowohl den Betrieb eines eigenen SMTP-Servers für den Versand als auch die Einrichtung von [VERP](#) für die zuverlässige Bounce-Behandlung. Darüber hinaus gewährleistet Mailjet eine gute Zustellbarkeit durch eine konstante Überwachung des Mailverkehrs, von Blacklists und der eigenen Reputation.

Konfiguration im Universal Messenger

Im Universal Messenger wird der SMTP-Server von Mailjet (siehe dazu [Mailjet / SMTP-Einstellungen](#)) in der `cmsbs.properties`-Datei eingetragen:

```
cmsbs.mail.smtpserver = "in-v3.mailjet.com"
cmsbs.mail.ssl = "true"
cmsbs.mail.smtpauth = "true"
cmsbs.mail.smtpuser = "[[MAILJET-SMTP-USERNAME]]"
cmsbs.mail.smtppassword = "[[MAILJET-SMTP-PASSWORD]]"
```

Damit Mailjet Bounces an den Universal Messenger melden kann, muss die `msgid` als Mail-Header mitgesendet werden. Außerdem muss das Tracking von Mailöffnungen und Klicks per Mailjet deaktiviert werden, da es sonst zu Doppelzählungen dieser Ereignisse käme. Die folgenden Konfigurationszeilen sorgen dafür, dass in jeder ausgehenden E-Mail die entsprechenden Header gesetzt werden:

```
cmsbs.mail.headers = X-MJ-CustomID X-Mailjet-TrackOpen X-Mailjet-TrackClick
cmsbs.mail.header.X-MJ-CustomID = "{msgid}"
cmsbs.mail.header.X-Mailjet-TrackOpen = 0
cmsbs.mail.header.X-Mailjet-TrackClick = 0
```

Zur Einrichtung der Bounce-Behandlung siehe [Bounce Management mit Mailjet](#).

3.6.3 Mehrere Mail-Relays

Der Universal Messenger kann so konfiguriert werden, dass er für den Mailversand mehrere Mail-Relays (SMTP-Server) ansprechen kann.

Die Auswahl des jeweils zu verwendenden Mail-Relays kann dabei unter anderem anhand folgender Gesichtspunkte erfolgen:

- Typ der Mail: Massenmailings (Newsletter) bzw. Transaktionsmails (Benachrichtigungen),
- Empfänger der Mail (z.B. Unterscheidung interne und externe Empfänger),
- Absender der Mail (z.B. wichtige Mailings sollen über ein bestimmtes Relay versendet werden),
- zur Erhöhung der Ausfallsicherheit (Failover, nur Performance Edition)
- und Lastverteilung zwischen mehreren gleichwertigen Mail-Relays (nur Performance Edition).

Alternativ kann per CSE-Callback eine projektspezifische Auswahllogik implementiert werden.

3.6.3.1 Konfiguration

Die Konfiguration weiterer Mail-Relays zusätzlich zum primären Relay erfolgt in der `cmsbs-conf/cmsbs.properties`-Datei. Die Namen aller zusätzlichen Relays müssen aufgelistet werden:

```
cmsbs.mail.relays = a b c
```

Damit werden drei zusätzliche Relays "a", "b" und "c" eingerichtet. (In der Customer Interaction Edition ist nur ein zusätzliches Mail-Relay erlaubt, in der Performance Edition können beliebige viele hinterlegt werden.)

Für jedes dieser Relays muss mindestens der zu verwendende SMTP-Server angegeben werden, z.B.:

```
cmsbs.mail.relay.a.smtpserver = "smtp-a.local"
cmsbs.mail.relay.b.smtpserver = "smtp-b.local"
cmsbs.mail.relay.c.smtpserver = "smtp-c.local"
```

Falls erforderlich, müssen auch Benutzername und Passwort für alle Relays einzeln angegeben werden, z.B.:

```
cmsbs.mail.relay.a.smtpauth = true
cmsbs.mail.relay.a.smtpuser = "username"
cmsbs.mail.relay.a.smtppassword = "xxx"
...
```

Die weiteren Einstellungen pro Relay gleichen denen für das primäre Relay, jedoch muss jeweils immer der Name des Relays in den Variablennamen in der Konfigurationsdatei eingesetzt werden, z. B. :

```
# 5 Threads senden parallel über dieses Relay:
cmsbs.mail.relay.a.threads = 5

# Nach jeder Mail 2 Sekunden warten:
cmsbs.mail.relay.a.delay = 2000

# Nach 120 Sekunden der Inaktivität Verbindung zum SMTP-Server schließen:
cmsbs.mail.relay.a.maxIdleTime = 120

# Nach spätestens 30 Minuten Verbindung zum SMTP-Server schließen:
cmsbs.mail.relay.a.maxUseTime = 1800

# Maximal 10 E-Mails über eine Verbindung zum SMTP-Server versenden:
cmsbs.mail.relay.a.maxMails = 10

# Alle 10 Sekunden auf inaktive Verbindungen prüfen:
cmsbs.mail.relay.a.checkIdleTime = 10

# Andere VERP-EnvelopeFrom-Adresse verwenden:
cmsbs.mail.relay.a.verp.envelopeFrom = "xyz-%0@SERVER.de"

# Verschlüsselung: ssl, tls, false
cmsbs.mail.relay.a.ssl = tls
```

Bis auf `.smtpserver`, `.smtpuser` und `.smtppassword` wird für alle Konfigurationsvariablen jeweils die Einstellung des primären Relays übernommen, falls sie nicht explizit überschrieben wird.

Die `EnvelopeFrom`-Adresse überschreibt den default-Wert und kann selbst beim Mail-Versand in der CSE oder in der Event-Datei überschrieben werden.

3.6.3.2 Auswahllogik und Failover

Auswahl nach Empfängeradresse

Für jedes Relay kann angegeben werden, für welche Empfängeradressen es verwendet werden darf bzw. nicht verwendet werden darf. Dazu kann jeweils ein regulärer Ausdruck für `.allow.toPattern` bzw. für `.deny.toPattern` angegeben werden.

```
# "a" darf nur an Adressen versenden, die auf "<tt>example.com</tt>" enden:
cmsbs.mail.relay.a.allow.toPattern = "@example.com$"

# "b" darf nicht an Adressen versenden, die auf "<tt>example.com</tt>" enden:
cmsbs.mail.relay.b.deny.toPattern = "@example.com$"

# Das primäre Relay ("default") darf nicht an Adressen versenden, die auf "<tt>example.com</tt>" enden:
cmsbs.mail.relay.default.deny.toPattern = "@example.com$"
```


Auswahl nach Absenderadresse

Für jedes Relay kann angegeben werden, für welche Absenderadressen es verwendet werden darf bzw. nicht verwendet werden darf. Dazu kann jeweils ein regulärer Ausdruck für `.allow.fromPattern` bzw. für `.deny.fromPattern` angegeben werden. Ein Anwendungsfall hierfür ist beispielsweise, daß bestimmte Mailings schneller versendet werden sollen und dafür ein anderes Mail-Relay verwenden sollen.

```
# E-Mails mit Absender wichtig@SERVER.de nicht über Default-Relay versenden:
cmsbs.mail.relay.default.deny.fromPattern = "^wichtig@SERVER.de$"
# ... sondern über das zusätzliche Relay "wichtig":
cmsbs.mail.relays = "wichtig"
cmsbs.mail.relay.wichtig.allow.fromPattern = "^wichtig@SERVER.de$"

# ... und dabei optional ein anderes VERP-Pattern verwenden:
cmsbs.mail.relay.wichtig.envelopeFrom = "wichtig+%0@SERVER.de"
```

Auswahl nach Mail-Typ

Für jedes Relay kann angegeben werden, für welche Typen von ausgehenden E-Mails es verwendet werden darf bzw. nicht verwendet werden darf.

```
# "a" darf nur Newsletter versenden:
cmsbs.mail.relay.a.allow.mailCategories = Newsletter

# "b" darf keine Benachrichtigungen versenden:
cmsbs.mail.relay.b.deny.mailCategories = Notification

# Das primäre Relay ("default") darf weder Newsletter noch Benachrichtigungen versenden:
cmsbs.mail.relay.default.deny.mailCategories = Newsletter Notification
```

Folgende Mail-Typen sind verfügbar:

- Newsletter: Massenmailings
- Notification: Benachrichtigungen (Transaktionsmails)
- Forward: E-Mails, die vom Bouncehandler weitergeleitet werden

Vorauswahl per Newsletter-Tagging

Durch die Angabe spezieller Tags beim Newsletterversand kann die Auswahl der zu verwendenden Mail-Relays eingeschränkt werden. So kann beispielsweise für einen Kampagnen-Newsletter ein anderer SMTP-Server verwendet werden als für den täglichen Newsletter.

So kann im Event-File beispielsweise angegeben werden, dass nur die Relays "a" und "c" verwendet werden dürfen:

```
<event>
  ...
  <tag>__allowMailRelay_a</tag>
  <tag>__allowMailRelay_c</tag>
</event>
```


Auf die so reduzierte Liste zu verwendender Mail-Relays wird anschließend die oben beschriebene Auswahllogik angewendet.

Lastverteilung (nur Performance Edition)

Aus der Menge der Relays, welche nach der Filterung nach Empfängeradresse bzw. Mail-Typ noch für den Versand einer konkreten Mail in Frage kommen, wählt der Universal Messenger schließlich das Relay mit der aktuell größten Anzahl freier Threads aus. Sollten mehrere Relays die gleiche Anzahl freier Threads aufweisen, wird nach dem Round-Robin-Prinzip eines davon ausgewählt.

Failover (nur Performance Edition)

Falls das ausgewählte Relay nicht erreichbar ist oder bei der Übertragung der Mail ein Fehler auftritt, wird das nächste in Frage kommende Relay angesprochen.

 Zur Überwachung der Mail-Relays gibt es in der Benutzeroberfläche des Universal Messenger unter "Extras - Mail-Relays" eine Übersichtsseite, mehr dazu im [Handbuch zur Benutzeroberfläche](#).

3.6.4 Inbox Preview

Der Universal Messenger kann den E-Mail-Vorschaudienst [Litmus](#) für die Inbox Preview einbinden. Mit Inbox Preview können automatisch Screenshots der wichtigsten Desktop Mailprogramme, Webmailer und mobilen Clients erstellt werden, um die korrekte Darstellung des Newsletters kontrollieren zu können. Darüber hinaus wird die Reaktion verschiedener Spamfilter auf den Newsletter ausgewertet und dargestellt.

Zur Nutzung der Inbox Preview ist eine gesonderte Anmeldung bei [Litmus](#) notwendig.

3.6.4.1 Konfiguration

Für die Verwendung von Litmus ist ein eigener Benutzeraccount bei Litmus erforderlich. Die Logindaten müssen wie folgt in der `cmsbs-conf/cmsbs.properties`-Datei angegeben werden:

```
cmsbs.mail.preview.litmus.apiurl = "https://[FIRMA].litmus.com"
cmsbs.mail.preview.litmus.username = "[LOGINNAME]"
cmsbs.mail.preview.litmus.password = "[PASSWORT]"
cmsbs.mail.preview.litmus.debug = false
```

Dabei sind [FIRMA], [LOGINNAME] und [PASSWORT] die Zugangsdaten des Litmus-Benutzeraccounts.

Der Universal Messenger wird für den Versand einen Test bei Litmus einrichten und den Newsletter anschließend im Rahmen des regulären Versands an die nur für diesen Test von Litmus erstellte E-Mail-Adresse versenden (...@emailtests.com). Es muss deshalb sichergestellt werden, dass der Universal Messenger E-Mails an emailtests.com zustellen kann.

Die Anzeige der Vorschau im Newsletterarchiv wird mit folgender Konfigurationsoption definiert:

```
cmsbs.mail.preview.litmus.iframe = true|false
```

Standardmäßig wird die Vorschau über den Link in der Navigation in einem neuen Browserfenster geöffnet. Um die Vorschauansichten innerhalb der Benutzeroberfläche des Universal Messenger in einem iFrame zu öffnen, muss dieser Wert auf `true` gesetzt werden.

3.6.4.2 Aufrufen aus der Benutzeroberfläche

Die Wizards "Newsletter versenden" und "Neue Mailingvorlage anlegen" enthalten im Schritt "Vorschau" den Abschnitt Inbox Preview.

Neben der Auswahl eines Eintrags oder eines Channels (Liste) kann hier auch der Inbox Preview-Versand gestartet werden. In das Adressfeld ist die E-Mail-Adresse eines existierenden Eintrags einzugeben, welcher für die Personalisierung des Newsletters verwendet werden soll. Dieser Eintrag wird zwar für die Personalisierung des Newsletters verwendet, der Newsletter wird jedoch nicht an diesen Eintrag gesendet.



Hinweis zu den Statistiken

Die E-Mail Adresse für den Versand an die Inbox Preview gehört nicht zum Zielsegment des Newsletters, der Versand wird aber als "versendet" in der Statistik gezählt, so diese Kennzahl größer als das "Zielsegement" ist. Es wird deswegen empfohlen, die Inbox Preview nicht mit einem Live-Versand zu kombinieren, sondern immer nur mit Testversendungen, damit die Statistiken nachvollziehbar bleiben.

3.6.4.3 Aufrufen aus einer Event-Datei (nur Inbox Preview)

Wenn die Inbox Preview per Event-Datei gestartet werden soll, muss im <destination>-Element statt der Channels bzw. VChannels ein entsprechendes <preview>-Element eingefügt werden:

```
...
<destination>
  <preview service="litmus">
    <baseEntry email="[EMAIL]" />
  </preview>
</destination>
...
```

Für [EMAIL] ist die E-Mail-Adresse eines vorhandenen Eintrags anzugeben, der für die Personalisierung des Newsletters verwendet werden soll. Dieser Eintrag wird zwar für die Personalisierung des Newsletters verwendet, der Newsletter wird jedoch nicht an diesen Eintrag gesendet.

3.6.4.4 Aufrufen aus einer Event-Datei (Inbox Preview parallel zum Test-Versand)

Mit Version 7.1 des Universal Messenger kann die Inbox Preview (Litmus) parallel zu einem normalen (Test-)Versand ausgeführt werden. Die Auswahl eines Channels und der Inbox Preview schließen sich nun nicht mehr gegenseitig aus.

Der Litmus-Versand wurde so geändert, dass keine E-Mail mehr an den "baseEntry" verschickt wird, also an den Eintrag, für den die Inbox Preview bei Litmus personalisiert ist. Daher kann im CMS für den Test-Versand nun etwas in dieser Art verwendet werden:

```
<destination>
  <channel>testchannel</channel>
  <preview service="litmus">
    <baseEntry email="test@localhost.local"/>
  </preview>
</destination>
```

Mit diesen Angaben wird sowohl `testchannel` beschickt als auch gleichzeitig die Inbox Preview ausgeführt.



Hinweis zu den Statistiken

Die E-Mail Adresse für den Versand an die Inbox Preview gehört nicht zum Zielsegment des Newsletters, der Versand wird aber als "versendet" in der Statistik gezählt, so diese Kennzahl größer als das "Zielsegement" ist. Es wird deswegen empfohlen, die Inbox Preview nicht mit einem Live-Versand zu kombinieren, sondern immer nur mit Testversendungen, damit die Statistiken nachvollziehbar bleiben.

3.6.4.5 Auswertung

Die Auswertung kann im Menü des Universal Messenger direkt unter "Newsletter - Newsletter-Archiv" abgerufen werden. Nach dem Versand des Newsletters kann es einige Zeit dauern, bis alle Vorschauansichten verfügbar sind.

Für Einzelauswertungen können Sie auch nach Erstellung eines Newsletters unter "Newsletter - Newsletter versenden" im Schritt "Vorschau" Inbox Preview verwenden, um sich die Vorschauansichten direkt dort oder später in einem extra Eintrag im Newsletter-Archiv anzusehen. Mehr Informationen dazu finden Sie im *Handbuch zur Benutzeroberfläche* unter "Newsletter versenden" und "Newsletterarchiv".

3.6.5 DKIM-Signierung

DomainKeys Identified Mail (DKIM) ist ein E-Mail-Validierungssystem zur Sicherstellung der Authentizität von E-Mail-Absendern. Es ermöglicht Empfängern, mit Hilfe der in der Nachricht enthaltenen Signatur und eines öffentlichen Schlüssels, zu prüfen, ob Mails (inklusive Anhängen) vom angegebenen Absender stammen und festzustellen, ob sie während der Übertragung manipuliert wurden.

3.6.5.1 Erzeugung eines Schlüssels

Ein neues Schlüsselpaar kann per Kommandozeilentool `openssl` erzeugt werden. Die Schlüssellänge muss mindestens 1024 Bit (besser 2048 Bit) betragen. Manche Werkzeuge zur Pflege von DNS-Einträgen können nicht mit langen Werten umgehen, so dass möglicherweise ein kleiner Schlüssel (1024 Bit) verwendet werden muss.

```
# Neuen privaten RSA-Schlüssel mit 2048 Bit Länge generieren:
openssl genrsa -out example.com-private.key.pem 2048

# Privaten Schlüssel ins DER-Format konvertieren:
openssl pkcs8 -topk8 -nocrypt -in example.com-private.key.pem -out example.com-private.key.der
-outform der

# Öffentlichen Schlüssel erzeugen:
openssl rsa -inform PEM -in example.com-private.key.pem -pubout
```

3.6.5.2 Konfiguration Universal Messenger

Der Universal Messenger kann ausgehende E-Mails nach DKIM-Standard signieren. Dazu muss in der globalen Konfigurationsdatei `cmsbs.properties` für jede Absender-Domain angegeben werden, welcher private Schlüssel jeweils zur Signierung verwendet werden soll. Für die beiden Absenderdomains *example.com* und *example.org* im Beispiel kann das wie folgt aussehen:

```

cmsbs.mail.dkim.domains = example.com example.org

cmsbs.mail.dkim.privatekey.example.com = cmsbs-conf/example.com-private.key.der
cmsbs.mail.dkim.selector.example.com = default

cmsbs.mail.dkim.privatekey.example.org = cmsbs-conf/example.org-private.key.der
cmsbs.mail.dkim.selector.example.org = default

```

3.6.5.3 Konfiguration DNS

Die passenden öffentlichen Schlüssel müssen jeweils im DNS-Eintrag der Absenderdomains hinterlegt werden. Für das obige Beispiel müsste folgender Eintrag im DNS-Eintrag von *example.com* hinterlegt werden:

```

$ORIGIN example.com.
default._domainkey TXT "v=DKIM1; g=*; k=rsa; p=MFwwDQYJKoZ ... AwEAAQ=="

```

3.6.6 One-Click-Unsubscribe

Der Universal Messenger unterstützt seit Release 7.35.0 One-Click-Unsubscribe nach [RFC 8058](#). Dabei werden in einem versendeten Newsletter zwei zusätzliche Header gesetzt, sodass Mail-Clients eine automatisierte Abmeldung auslösen können.

Dazu ist es notwendig, beim Newsletterversand eine Verknüpfung zu einer Newsletter App-Instanz herzustellen. So kann eine Abmeldung gemäß den Einstellungen aus der Newsletter App durch den Mail-Client des Empfängers ausgelöst werden.

Zunächst muss die Funktion durch Festlegen der URL des neuen One-Click-Unsubscribe-HTTP-Endpunkts aktiviert werden:

```

cmsbs.mail.one_click_url = http://my-domain.com/p/de.pinuts.cmsbs.newsletter.OneClick

```

Dieser neue HTTP-Endpunkt muss in die [Whitelist des REST-Proxy](#) eingetragen werden; z.B.:

```

cmsbs.restproxy.limit.controller.whitelist.1 = "de.pinuts.cmsbs.newsletter.OneClick"

```

Sind diese Einstellungen erfolgt, erweitert sich der [Wizard für den Newsletterversand](#) um ein neues Feld zur Zuordnung der Newsletter App-Instanz:

Optionen

Newsletter-Serie: ... ✕

Newsletter App für One-Click-Unsubscribe: ... ✕

Sendezeitpunkt:

Schlagwörter / Tags:

Bei der Erstellung eines Event-Files kann die Zuordnung wie folgt vorgenommen werden:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<event ...>
  <data>
    <appInstance name="985fb277-2451-471a-a747-2ba074f70569" />
    <email ... >
```


Beim Newsletterversand setzt der Universal Messenger dann entsprechend den Einstellungen die folgenden Header:

```
List-Unsubscribe:
<http://my-domain.com/p/de.pinuts.cmsbs.newsletter.OneClick/unsubscribe?name=985fb277-2451-471a-a747-2ba074f70569>
List-Unsubscribe=One-Click
```

Wenn eine Transaktions-E-Mail oder ein Newsletter ohne Zuordnung zu einer Newsletter App-Instanz versendet wird, werden die Header entsprechend ohne Angabe der Newsletter App-Referenz gesetzt:

```
List-Unsubscribe:
<http://my-domain.com/p/de.pinuts.cmsbs.newsletter.OneClick/unsubscribe?msgid=1F1D8.2CID4.F77FB27D94479A2C>
List-Unsubscribe=One-Click
```

Die Abmeldung erfolgt in diesem Fall dann von allen Listen, die nicht explizit von der automatischen Abmeldung ausgenommen sind (siehe dazu *Listen und Segmente* im Benutzerhandbuch).

 Laut RFC wird eine DKIM-Signierung der ausgehenden E-Mails vorausgesetzt.

Bislang unterstützen nur sehr wenige Mail-Clients den neuen Standard.

3.7 Einstellungen zum CMS

```
cmsbs.cms.type = nps|reddot|none
```

Aktiviert die Integration für das angegebene Content Management System.

Hinweis: Diese und einige weitere Konfigurationsoptionen sind in den Handbüchern zur Anbindung der Content Management Systeme beschrieben, die zusammen mit dem Universal Messenger ausgeliefert werden.

```
reddot.mode = true|false
```

Bei `true` werden die Standardwerte einiger Konfigurationsoptionen auf für die Integration mit dem OpenText Content Management Server optimierte Einstellungen gesetzt, u.a. wird das mit der Konfigurationsoption `cmsbs.jsp.css` wählbare Stylesheet auf das OpenText-Layout gesetzt. Zur Integration mit dem OpenText Content Management Server müssen zusätzlich zu dieser Konfigurationsoption die beiden Einstellungen `cmsbs.cms.type` und `reddot.listener.start` gesetzt werden.

3.8 Prozessbenachrichtigungen

Für die typischen Prozesse des Newsletter-Marketings gibt es im Universal Messenger vordefinierte Prozessbenachrichtigungen, die automatisch per E-Mail versendet werden können. Darüber hinaus können beliebige weitere benutzerdefinierte Benachrichtigungen über die Benutzeroberfläche angelegt werden.



Seit Version 6.4 des Universal Messenger können Benachrichtigungen nicht mehr über die Konfigurationsdatei sondern nur noch über die Benutzeroberfläche für Administratoren angelegt und/oder geändert werden. Siehe "Handbuch zur Benutzeroberfläche" unter "Extras".

3.8.1 Globale Einstellungen

```
cmsbs.changes.nochannels.de = <Text>
```

Text in Deutsch, der für die Personalisierungsvariable `{channels}` ausgegeben wird, wenn keine Channels abonniert sind.

```
cmsbs.changes.nochannels.en = <Text>
```

Text in Englisch, der für die Personalisierungsvariable `{channels}` ausgegeben wird, wenn keine Channels abonniert sind.

3.8.1.1 Automatischer Versand

Aus historischen Gründen gibt es für die Prozesse "Neuanmeldung", "Bestätigung", "Änderung" und "Abmeldung" Einstellungen, die definieren, ob bei der Ausführung dieser Prozesse automatisch eine E-Mail verschickt wird.


```
cmsbs.newuser.sendmail = true|false  
cmsbs.confirmrequest.sendmail = true|false  
cmsbs.changes.sendmail = true|false  
cmsbs.unsubscribe.sendmail = false|true
```

Bis zur Version 7.0 des Universal Messenger sind die Default-Werte dieser Einstellungen alle auf `true` gesetzt, werden von den aktuellen Connectoren allerdings automatisch abgeschaltet, so dass ein Versand der Prozessbenachrichtigungen explizit implementiert werden muss.

Seit Version 7.1 ist der Default-Wert für `cmsbs.unsubscribe.sendmail` auf `false` gesetzt, da bei einer Abmeldung per E-Mail weiterhin die Prozessnachricht `unsubscribe` verschickt wurde. Dies muss nun explizit aktiviert werden.

Neben dem automatischen Versand von Prozessbenachrichtigungen können eingehende E-Mails vom Bounce Management auf das Stichwort "Unsubscribe" untersucht werden, um eine Abmeldung vom Newsletter auszuführen. Die Regeln für das Erkennen einer Abmeldung sind in der Datei `unsubscribe.keywords` definiert, siehe [Bounce Management - Unsubscribes](#).

3.8.2 Passwort Erinnerung

Verschlüsselte Passwörter zurücksetzen

Da ein als Einweghash gespeichertes Passwort nicht mehr rekonstruiert werden kann, muss die Logik beim Versand von Passwort-Vergessen-E-Mails angepasst werden. Dabei muss innerhalb einer Transaktion ein neues Passwort vergeben und die E-Mail versendet werden. Nur innerhalb dieser selben Transaktion kann der Platzhalter `{password}` durch das Klartextpasswort ersetzt werden.

Beispiel: Core Scripting Engine

```
var e = UM.getEntryByUid(uid);  
e.set('password', '{reset}');  
e.notifications.add('newpassword');  
UM.commitEntries();
```

Wenn in einer E-Mail-Benachrichtigung der Platzhalter `{password}` verwendet wird, ohne dass in der gleichen Transaktion das Passwort gesetzt wurde, liefert `{password}` den leeren String und es wird im Log folgende Warnung ausgegeben: "WARN cmsbs: Clear text password not available for attribute password".

3.8.3 Benutzerdefinierte Benachrichtigungen

Mit dem Universal Messenger können über die Programmierschnittstellen beliebige Benachrichtigungen an Einträge versendet werden. In benutzerdefinierten Benachrichtigungen werden alle Attribute einer Benachrichtigung zusammengefasst und im Server gespeichert, so dass zum Versand einer solchen Benachrichtigung über die Programmierschnittstellen lediglich der Name der Benachrichtigung angegeben werden muss. Benutzerdefinierte Benachrichtigungen können in der Benutzeroberfläche für Administratoren angelegt und bearbeitet werden.

3.9 E-Mail- und Mobilfunknummer-Verifizierer

3.9.1 E-Mail-Adressen Verifizierer

```
cmsbs.emailverifier.dns = <IP-Nummer Nameserver>
```

Optional kann der Universal Messenger bei der Eintragung einer E-Mail-Adresse über eine Abfrage des Nameservers prüfen, ob eine korrekte Domain angegeben wurde. Falls die IP-Nummer eines Nameservers eingetragen ist, wird bei jeder Eintragung einer E-Mail-Adresse geprüft, ob eine korrekte Domain angegeben wurde

```
cmsbs.emailverifier.autodomain = <Top Level Domain>
```

Optional kann eine Top Level Domain angegeben werden, die von dem Universal Messenger an allen E-Mail-Adressen angefügt wird, bei denen die Domain nur aus einem Teil besteht. Aus `meier@t-online` wird automatisch `meier@t-online.de` korrigiert. Die Top Level Domain muss ohne Punkt angegeben werden.

3.9.2 Mobilfunknummern Verifizierer

```
cmsbs.mobileverifier.autocc = <Ländervorwahl>
```

Optional kann eine Ländervorwahl angegeben werden, die von dem Universal Messenger statt einer führenden Null vor alle Mobilfunknummern geschrieben wird.

3.10 Benutzerdefinierte Statistiken

Die in dem Universal Messenger vorhandenen Statistiken zu den Einträgen und Channels können mit eigenen Statistiken erweitert werden.

```
custom.stat = <Liste der Statistiknamen>
```

Durch Leerzeichen getrennte Liste der (internen) Namen der benutzerdefinierten Statistiken. Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten. Für jede benutzerdefinierte Statistik müssen die folgenden Optionen angegeben werden.

```
custom.stat.<Statistikname>.type = count|incr
```

Benutzerdefinierte Statistiken können Ereignisse zählen, die über die Programmierschnittstellen übermittelt werden (Typ 'incr') oder es können täglich um 0:00 Uhr alle Einträge gezählt werden, die eine mit der folgenden Option angegebene Bedingung erfüllen (Typ 'count').

```
custom.stat.<Statistikname>.query = <Abfrage>
```

Für benutzerdefinierte Statistiken vom Typ 'count' muss in dieser Option eine Abfrage definiert werden, über die die Einträge täglich um 0:00 Uhr gezählt werden. Zur Beschreibung der Abfragen siehe [Abfragesprache](#).

Für benutzerdefinierte Statistiken vom Typ 'incr' entfällt diese Option.

```
custom.stat.<Statistikname>.title = <Text>
```

Titel der Statistik, der in der Benutzeroberfläche des Universal Messenger angezeigt wird.

3.11 Benutzerdefinierte Logdateien

Zusätzlich zu den von dem Universal Messenger erstellten Logdateien können benutzerdefinierte Logdateien geschrieben werden. Die Einträge für diese Logdateien werden über die Programmierschnittstellen übermittelt. Bevor eine Logdatei über die Programmierschnittstellen verwendet werden kann, muss sie mit den folgenden Optionen in der Konfigurationsdatei eingerichtet werden.

```
custom.log = <Liste der Logdateinamen>
```

Durch Leerzeichen getrennte Liste der (internen) Namen der benutzerdefinierten Logdateien. Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten. Für jede benutzerdefinierte Logdatei müssen die folgenden Optionen angegeben werden.

```
custom.log.<Logdateiname>.format = clf|plain
```

Benutzerdefinierte Logdateien können im Common Log Format (Typ `clf`) geschrieben werden, das den Logdateien eines Webserverns entspricht und von den entsprechenden Statistikprogrammen ausgewertet werden kann. Alternativ könne die Logdateien in einem eigenen Textformat (Typ `plain`) geschrieben werden.

```
custom.log.<Logdateiname>.file = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der benutzerdefinierten Logdatei.

```
custom.log.<Logdateiname>.rotate = true|false
```

Bei `true` wird jeweils nächtlich die Logdatei rotiert und die vorhergehende Version mit dem Datum im Dateinamen archiviert. Bei `false` werden neue Einträge an eine bestehende Logdatei angehängt.

```
custom.log.<Logdateiname>.time = <Formatierungsstring>
```

Für benutzerdefinierte Logdateien vom Typ `plain` kann ein Formatierungsstring für die Ausgabe von Datum und Uhrzeit angegeben werden. Jedem Eintrag in die Logdatei fügt der Universal Messenger am Beginn der Zeile einen entsprechend formatierten Zeitstempel hinzu.

Beispiel für einen Formatierungsstring: `"dd.MM.yy HH.mm.ss"`

3.12 Import/Export

```
cmsbs.transfer.replace.debug = true|false
```

Bei `true` gibt der UserTransfer beim Start die aus der Map-Datei gelesenen Regeln für die Transformatoren zum Import/Export aus. Diese Option kann zu Testzwecken verwendet werden.

3.13 Zusatzmodule

Anstelle von Modulen kann der Universal Messenger über Wizards erweitert werden. Die hier beschriebenen Konfigurationsoptionen sind deswegen nur noch für ältere Entwicklungen relevant.

```
cmsbs.modules.names = <Modulnamen| " ">
```

Liste der mit Leerzeichen getrennten (internen) Namen der Zusatzmodule, die in den Universal Messenger eingebunden werden sollen. Sollen keine Module verwendet werden, muss an dieser Stelle ein `" "` eingetragen werden und es müssen die folgenden Konfigurationsoptionen mit einem `#` am Zeilenanfang auskommentiert werden.

```
cmsbs.module.<Modulname>.title.<Sprache> = <Text>
```

Titel des Moduls, der zur Auswahl in dem Universal Messenger angezeigt wird. Optional kann der Titel durch die Angabe einer Sprache lokalisiert werden.


```
cmsbs.module.<Modulname>.link = <HTTP-URL>
```

URL der Loginseite des Moduls, die bei der Auswahl des Moduls in dem Universal Messenger als Linkziel verwendet wird. Ist das Modul passwortgeschützt, müssen Benutzername und Passwort als Parameter an die URL angehängt werden.


3.14 Brute-Force-Angriffe erschweren


Um Brute-Force-Angriffe auf die Login-Seite der Benutzeroberfläche zu erschweren, wurde ein Mechanismus hinzugefügt, der zu schnelle Login-Versuche verhindert. Dazu gibt es eine weitere Log-Datei

`cmsbs-work/audit.log`, in der Login-/Logout-Vorgänge und die Ausgaben des Schutzmechanismus festgehalten werden. Die Ausgabe der Login-Vorgänge in der globalen Log-Datei entfällt dadurch.

 Die Default-Werte sind so gewählt, dass im erwarteten Normalbetrieb kein Login verhindert werden muss. Wird ein Login-Versuch blockiert, ist das im Webbrowser jedoch nicht zu erkennen, um einem Angreifer keine Information darüber zu liefern. Es wird weiterhin die "Login verweigert"-Meldung angezeigt.

Der Schutz besteht aus mehreren Stufen. Zunächst wird die IP-Adresse des anfragenden Client (und die davorliegender Reverse-Proxies) geprüft und nur eine maximale Anzahl von Versuchen innerhalb der letzten Minute erlaubt. Sowohl die Anzahl als auch dieser Zeitraum sind konfigurierbar. Der Default von "60 pro Minute" erlaubt im normalen Betrieb etwa einen Login-Versuch pro Sekunde (eben vom selben Client aus).

 Durch transparente Reverse-Proxies kann die tatsächliche IP-Adresse des Clients u.U. nicht korrekt bestimmt werden. In diesem Fall werden häufig die selbe Adresse für alle Clients geliefert.

 Wenn viele Redakteure gleichzeitig im dem Universal Messenger arbeiten oder in Schulungssituationen, kann es nötig sein, hier mehr Login-Versuche zu erlauben, oder den Zeitraum zu vergrößern. Dann kann es nämlich nötig sein, allen 50 Personen das Einloggen innerhalb von einer Minute zu ermöglichen.

Die zweite Stufe bezieht sich auf den Login-Namen, ohne Betrachtung der IP-Adresse. Auch hier werden wieder nur eine bestimmte Anzahl von Versuchen pro Zeiteinheit erlaubt. Die IP-Adresse wird nicht betrachtet, um einen verteilten Brute-Force-Angriff aus einem Cluster von Clients heraus zu verhindern.

Da diese zweite Stufe jedoch kurze Spitzen ("Bursts") erlaubt, erzwingt nun die dritte Stufe einen zeitlichen Abstand zwischen zwei Login-Versuchen des selben Nutzers. Der Default-Wert von 0.9s (900ms) soll einen schnellen Vertipper erlauben, bremst aber einen Brute-Force-Angriff effektiv aus.

Die Einstellungen mit ihren Default-Werten auf einen Blick:

```
# Daily rotated log file
cmsbs.log.audit = cmsbs-work/audit.log

# Allow X login attempts for each IP Client ...
cmsbs.gui.login.protect.client = 60
# ... per minute
cmsbs.gui.login.protect.client.time = 60

# Allow X login attempts for each User (name) ...
cmsbs.gui.login.protect.user = 60
# ... per minute
cmsbs.gui.login.protect.user.time = 60

# ...and require Z ms between each attempt
cmsbs.gui.login.protect.user.min = 900
```

Die Ausgaben im `audit.log` beginnen mit einer Zusammenfassung der Einstellungen:

```
2017-01-19 10:46:11,511 * Now allowing: 60 attempts per client in 60s, 60 attempts per
user in 60s, forcing 1500ms between user attempts.
```

Erfolgreiche Versuche zeigen den verwendeten Usernamen und die IP-Adresse:

```
2017-01-19 10:49:49,837 Login failed: admin [0:0:0:0:0:0:0:1]
```

Bei erfolgreichen Logins wird der Username und ggf. seine Rolle angezeigt:

```
2017-01-19 12:12:12,395 Login: someuser (redakteur) [0:0:0:0:0:0:0:1]
```

Wenn ein Anmeldeversuch blockiert wird, werden Grund, Username und IP-Adresse angezeigt:


```
2017-01-19 10:49:51,326 BLOCKED: user re-trying too fast, for login: admin
[0:0:0:0:0:0:0:1]
```

Ein Login-Versuch kann auch fehlschlagen, weil die Rollendefinition ungültig ist. Diese Meldung wird auch im globalen Log ausgegeben, da es sich um einen Fehler in der Entwicklungsphase handelt.

```
2017-01-19 12:16:51,120 Login failed, User notab has illegal admin role `notab'
[0:0:0:0:0:0:0:1]
```


Es werden sowohl explizite Logouts als auch abgelaufene Sessions protokolliert. So ist im `audit.log` nachträglich nachvollziehbar, wer zu einem gegebenen Zeitpunkt am System angemeldet war.

```
2017-01-19 12:22:42,659 Logout: admin
2017-01-19 12:25:46,335 Timeout: admin
```

 Dieser Schutzmechanismus wirkt sich auf alle konfigurierbaren Login-Methoden aus. Die GUI ist also immer geschützt. Wenn eine CSE-Programmierung jedoch intern auf die Login-Methoden zugreift (Klasse GuiUserLogin), greift der Schutz nicht.

3.15 Trigger E-Mails

Der Universal Messenger ermöglicht das Einrichten und Konfigurieren von Trigger E-Mails. Hier erfahren Sie, wie Sie die entsprechenden Apps installieren können.

 **Hinweis**

Wenn Sie die Apps bereits eingerichtet haben, erfahren Sie in dem [Handbuch zur Benutzeroberfläche](#) unter dem Abschnitt [Trigger E-Mails](#), wie Sie neue App-Instanzen anlegen und konfigurieren.

Die Apps selbst werden seit Version 7.34.0 mit dem Installer automatisch installiert. Zur Inbetriebnahme muss jedoch eine weitere Attributgruppe in die Attributkonfiguration aufgenommen werden.

Dazu wird die folgende Zeile in die `additional.attributes`-Datei eingefügt:

```
include: cmsbs-conf/cse/api/plugins/de.pinuts.cmsbs.mailschedule/ailschedule.attributes
```

Bitte beachten Sie, dass für die Inbetriebnahme zwingend eine entsprechende Universal Messenger-Lizenz erforderlich ist.

3.16 Eingebetteter Tomcat

Der Universal Messenger kann grundsätzlich auf zwei verschiedene Arten betrieben werden:

1. Als Webapplikation, die in einen vorhandenen Apache Tomcat Application Server "deployt" wird
2. oder "standalone" mit Hilfe des eingebetteten Apache Tomcat Application Server.

Bei Verwendung des eingebetteten Tomcat wird der Universal Messenger über folgende Skripte gestartet bzw. gestoppt:

3.16.1 Starten

```
UM/scripts/startup.(sh | bat)
```

3.16.2 Stoppen

```
UM/scripts/shutdown.(sh | bat)
```

3.16.3 Starten im Vordergrund

Alternativ kann der Universal Messenger zu Test- und Debuggingzwecken auch im Vordergrund gestartet werden. So werden alle Logausgaben direkt in die Konsole umgeleitet:

```
UM/scripts/run.(sh | bat)
```

In diesem Fall kann der Universal Messenger mit Strg+C wieder gestoppt werden.

3.16.4 Starten als systemd-Service

Auf Linux-Systemen mit systemd kann der Universal Messenger auch als Service eingerichtet werden. Dazu muss mit root-Rechten die vom Installer erzeugte Service-Definitionsdatei in das entsprechende Verzeichnis der systemd-Konfiguration kopiert werden:

```
cp UM/scripts/um.service /etc/systemd/system/
```

Danach kann der Universal Messenger wie üblich per *systemctl* gestartet und gestoppt werden.

3.16.5 Konfiguration des eingebetteten Tomcat

Die Konfiguration diverser Server-Parameter erfolgt über die folgenden Konfigurationsoptionen, die wie üblich in `cmsbs-conf/cmsbs.properties` oder einer beliebigen `*.properties`-Datei im Verzeichnis `cmsbs-conf/conf.d/` gesetzt werden können.

3.16.5.1 cmsbs.server.port

Gibt den TCP-Port an, auf welchem der Universal Messenger erreichbar sein soll. Default: 8080

3.16.5.2 cmsbs.server.address

Gibt den Namen bzw. die IP-Adressen des Netzwerkinterfaces an, auf welchem der Universal Messenger erreichbar sein soll. Default: Leerstring für *alle*.

Soll der Universal Messenger nur lokal auf dem gleichen Host erreichbar sein, kann hier z.B. "localhost" angegeben werden.

3.16.5.3 cmsbs.server.context

Gibt den Namen des Webapplikationskontextes an, unter welchem der Universal Messenger erreichbar sein soll. Default: "cmsbs"

3.16.5.4 cmsbs.server.retryStartupDelay

Wenn der Universal Messenger nicht korrekt starten kann – z.B. weil die Datenbank (noch) nicht erreichbar ist – wird automatisch ein Neustart ausgeführt. Diese Konfigurationsoption gibt die Wartezeit in Sekunden vor dem nächsten Neustartversuch an. Default: 15

3.16.5.5 cmsbs.directory.server.webapps

Gibt das Verzeichnis an, aus welchem weitere Webapplikationen zusammen mit dem Universal Messenger deployt werden sollen. Das entspricht dem "webapps/"-Verzeichnis eines regulären Tomcat. Default: "cmsbs-work/webapps".

3.16.5.6 cmsbs.directory.server.work

Gibt an, wo das temporäre *Work*-Verzeichnis des Tomcat liegt. Default: "cmsbs-work/tomcat-work".

3.16.5.7 cmsbs.log.server

Gibt den Namen der Logdatei an, in welche alle Ausgaben des Tomcat laufen sollen. Default: "cmsbs-work/server.log".

3.16.5.8 cmsbs.server.maxThreads

Gibt die maximale Anzahl an parallel laufenden Worker-Threads an. Default: 200

3.16.5.9 cmsbs.server.acceptCount

Gibt die Länge der Warteschlange für eingehende Verbindungsanfragen an. Default: 100

3.16.5.10 cmsbs.server.sessionCookie.global

Gibt an, ob der Session-Cookie für die GUI global für alle Webapplikationen im gleichen Tomcat gelten soll, also den Pfad "/" bekommt. Andernfalls bekommt er als Pfad den Namen des Kontextes, in welchem der Universal Messenger läuft (z.B. "/cmsbs"). Default: *true*

3.16.5.11 cmsbs.server.sessionCookie.name

Gibt den Namen des Session-Cookies an, welchen der Universal Messenger setzt. Default: "UMSESSIONID"

3.16.5.12 cmsbs.server.session.Dir

Gibt das Verzeichnis für die Session-Serialisierung an. Default: "cmsbs-work/tomcat-work"

3.16.5.13 cmsbs.server.cookies.secure

Gibt an, ob die Session-Cookies, welche der Universal Messenger setzt, mit dem *Secure*-Attribut versehen werden sollen.

Der Default ist *false*, was in aktuellen Browsern unter bestimmten Umständen zu Warnmeldungen oder Loginproblemen führen kann.

Generell gilt: Ist der Universal Messenger nur per HTTPS zugänglich, sollte diese Option auf *true* gesetzt werden. Ist einheitlich nur der Zugang per HTTP möglich, sollte die Option auf *false* verbleiben. Ein Mischbetrieb kann problematisch sein, erfordert aber auf jeden Fall die Einstellung *false*.

3.16.5.14 cmsbs.server.cookies.sameSite

Gibt den Wert für das *SameSite*-Cookie-Attribut an, das für jeden Cookie gesetzt werden soll, den der Universal Messenger setzt. Default: *Leerstring* für *Unset*.

Erlaubte Werte:

Wert	Beschreibung
Unset	<i>SameSite</i> -Option wird nicht gesetzt. (Default)
None	Für Cross-Domain-Anwendungen; ermöglicht, dass Cookies auch bei Cross-Origin-Requests mitgesendet werden. Erfordert, dass das <i>Secure</i> -Flag (siehe oben) gesetzt ist. (Ist teilweise noch der Default der führenden Browser; Stand Mai 2020.)
Lax	Cookies sollen bei Cross-Origin-Requests nicht mitgesendet werden. (Wird zukünftig der Default der führenden Browser werden.)
Strict	Cookies sollen bei Cross-Origin-Requests mitgesendet werden, allerdings nicht beim ersten Aufruf der Ursprungsseite über einen Link von einer Cross-Origin-Seite.

Siehe auch:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie#Browser_compatibility
- <https://web.dev/samesite-cookies-explained/>
- <https://web.dev/samesite-cookie-recipes/>

3.16.5.15 cmsbs.server.accesslog.enabled

Gibt an, ob der eingebettete Tomcat ein Access-Log schreiben soll. Default: *false*

3.16.5.16 cmsbs.server.accesslog.dir

Gibt das Verzeichnis an, in das das Access-Log geschrieben wird. Default: "cmsbs-work/".

3.16.5.17 cmsbs.server.accesslog.maxDays

Gibt an, nach wie vielen Tagen die rotierten Access-Logs gelöscht werden sollen. Default: 14

3.16.5.18 cmsbs.server.accesslog.pattern

Gibt das Formatierungsmuster für einzelne Zeilen im Access-Log an. Siehe dazu https://tomcat.apache.org/tomcat-8.0-doc/config/valve.html#Access_Log_Valve

3.16.5.19 cmsbs.server.accesslog.buffered

Gibt an, ob das Schreiben ins Access-Log zur Performanceoptimierung zwischengepuffert und dadurch verzögert erfolgen soll. Default: *true*

3.16.5.20 cmsbs.server.h2.port

Die häufig in Entwicklungsumgebungen verwendete eingebettete H2-Datenbank kann optional im Server-Modus gestartet werden, sodass während der Laufzeit des Universal Messenger im eingebetteten Tomcat von außen mit einem externen DB-Tool (z.B. DbVisualizer) per JDBC auf die Datenbank zugegriffen werden kann.

Dazu muss folgende globale Option gesetzt werden:

```
cmsbs.server.h2.port = 9092
```

Damit wird die H2-Datenbank beim Start des eingebetteten Tomcat auf localhost:9092 erreichbar gemacht. Die JDBC-URL wird bei jedem Start ausgegeben, z.B.:

```
* Starting H2 server on port 9092, JDBC URL: jdbc:h2:tcp://localhost:9092//UM/cmsbs-work/db.h2
```

In DbVisualizer kann die Verbindung mit folgenden Einstellungen hergestellt werden:

- Database Driver: *H2 embedded*
- Settings Format: *Database URL*

- Database URL: (wie im Log angegeben)
- Database Userid: sa
- Database Password: (leer)

3.16.5.21 cmsbs.server.errorReporting.secure

Gibt an, ob die Fehlerseiten des eingebetteten Tomcat aus Sicherheitsgründen keine Fehlerdetail- oder Tomcat-Versionsangaben enthalten sollen. Default: *true*

3.16.5.22 JVM-Speichereinstellungen

Standardmäßig wird die Java Virtual Machine mit 1 GB Heap-Größe gestartet (Stand der Universal Messenger 7.39.0). Diese Vorgabe kann in der Datei `UM/scripts/.umrc` überschrieben werden. Durch folgende Anpassung wird die Heap-Größe auf 3 GB erhöht:

```
[...]
# Passed directly to the JVM
UMJVMOPTS="-Xmx3G -Djava.security.egd=file:/dev/./urandom -Dfile.encoding=UTF-8"
[...]
```

3.16.6 SSL/TLS-Konfiguration

Der eingebettete Tomcat kann seit Release 7.45.2 per Konfigurationsanpassung auf SSL/TLS umgestellt werden.

Der HTTP-Connector des Tomcat wird durch Setzen der folgenden Konfigurationsoption auf HTTPS umgestellt:

```
cmsbs.server.ssl = true
```

Die zu verwendenden Zertifikate können entweder als einzelne Dateien im PEM-Format vorliegen oder aus dem lokalen KeyStore entnommen werden:

Beispiel mit Let's Encrypt-Zertifikaten (PEM-Format)

```
cmsbs.server.ssl.certificateFile      = /opt/um/ssl-certs/cert.pem
cmsbs.server.ssl.certificateKeyFile   = /opt/um/ssl-certs/key.pem
cmsbs.server.ssl.certificateChainFile = /opt/um/ssl-certs/chain.pem
```

Beispiel mit Verwendung des Java-Keystore

```
cmsbs.server.ssl.certificateKeystoreFile      = /opt/um/.keystore
cmsbs.server.ssl.certificateKeystorePassword = changeit
cmsbs.server.ssl.certificateKeyAlias         = tomcat
# cmsbs.server.ssl.certificateKeyPassword    = changeit
```

In beiden Fällen können bei Bedarf noch weitere Einstellungen angepasst werden:

```
cmsbs.server.ssl.protocols = -TLSv1,-TLSv1.1,+TLSv1.2,+TLSv1.3
```

Wie bei HTTP können TCP-Port und -Interface festgelegt werden, z.B.:

```
cmsbs.server.address = um.acme.de
cmsbs.server.port    = 8443
```

3.16.6.1 Zertifikat in KeyStore importieren

Das folgende Beispiel zeigt, wie ein im PEM-Format vorliegendes Zertifikat (z.B. von Let's encrypt) in den KeyStore importiert werden kann.

Folgende Dateien müssen vorliegen:

- `cert.pem`: Zertifikat
- `privkey.pem`: privater Schlüssel
- `chain.pem`: Kette aus Zertifikaten ausgehend der Root-CA

Folgende Umgebungsvariablen müssen entsprechend gesetzt sein:

- `KEYSTORE_FILE`: Pfad der KeyStore-Datei
- `STOREPASS`: KeyStore-Passwort

Unter diesen Voraussetzungen können die Zertifikate wie folgt auf der Kommandozeile konvertiert und importiert werden:

```
# Add Root Certificate
keytool -import -alias root -keystore ${KEYSTORE_FILE} -trustcacerts -file chain.pem -storepass
${STOREPASS} -noprompt

# Convert the key to pkcs12
openssl pkcs12 -export -inkey privkey.pem -in cert.pem -out tempstore.p12 -passout pass:

# Import to keystore:
keytool -importkeystore -srckeystore tempstore.p12 -srcstorepass "" -srcstoretype PKCS12
-destkeystore ${KEYSTORE_FILE} -storepass ${STOREPASS}

# Change alias:
keytool -changealias -alias 1 -destalias tomcat -keystore ${KEYSTORE_FILE} -storepass
${STOREPASS}
```

3.17 API-Token

Zur Authentifizierung muss beim Aufruf verschiedener REST-Endpunkte ein *API-Token* (früher auch als *Open-Password* bezeichnet) angegeben werden. Bis einschließlich Version 7.40 konnte dieses Token ausschließlich per Konfigurationsoption `cmsbs.open` auf einen konkreten – oder per `cmsbs.open = {random}` auf einen zufälligen – Wert gesetzt werden.



Ab Version 7.41 sollte die Konfigurationsoption `cmsbs.open` nicht mehr verwendet werden!

Stattdessen kann nun unter *Extras / API-Token* ein neues Token generiert werden, das PBKDF2.1-verschlüsselt in der Datenbank gespeichert wird:

API-Token

Hier kann ein neues API-Token zur Verwendung als *open*-Passwort generiert werden.

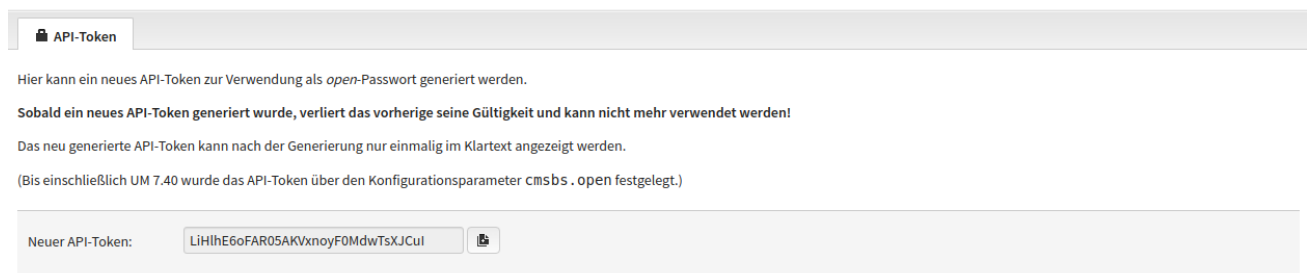
Sobald ein neues API-Token generiert wurde, verliert das vorherige seine Gültigkeit und kann nicht mehr verwendet werden!

Das neu generierte API-Token kann nach der Generierung nur einmalig im Klartext angezeigt werden.

(Bis einschließlich UM 7.40 wurde das API-Token über den Konfigurationsparameter `cmsbs.open` festgelegt.)

API-Token generieren

Das generierte Token wird dann einmalig angezeigt und kann mittels des *Kopieren*-Buttons in die Zwischenablage kopiert werden:



3.18 Virens Scanner einbinden

Zum Schutz vor Schadsoftware kann ein Virens Scanner eingebunden werden, der beim Dateiuupload in den folgenden Apps aktiv wird und die Übernahme von als böartig erkannten Dateien in den verhindert:

- Kontaktformular App
- Service Desk

3.18.1 Unterstützte Virens Scanner

Es können generell alle Virens Scanner eingebunden werden, die als Kommandozeilenprogramm auf dem gleichen System installiert sind und einen *synchronen* Scan ermöglichen. Das jeweilige Kommandozeilenprogramm muss demnach per Exit-Code kommunizieren, ob die Datei als böartig eingestuft wird oder nicht. Asynchrone Virens Scanner, die in einem Hintergrundprozess eine Queue von zu scannenden Dateien abarbeiten, sind für eine Verwendung in diesem Kontext nicht geeignet.

Neben Kommandozeilenvirens Scanner wird zurzeit auch der [Online-Virens Scanner von Cloudmersive](#) unterstützt. In diesem Fall ist ein Abonnement des entsprechenden Services erforderlich.

3.18.2 Einrichtung

Soll der Online-Virens Scanner von Cloudmersive verwendet werden, muss zunächst der API-Token in der globalen Konfigurationsdatei (`cmsbs-conf/cmsbs.properties`) angegeben werden:

```
cmsbs.virusscan.cloudmersive.apikey = 12345678-abcd-1234-5678-1234567890ab
```

(Nach dem Eintragen des API-Tokens ist ein Neustart Dessen erforderlich.)

Die weitere Einrichtung erfolgt über die Backoffice-Oberfläche unter *Extras / Apps / Virus Scan*:

Falls noch nicht vorhanden, muss hier zunächst eine Instanz der *Virus Scan*-App angelegt werden. In den Einstellungen dieser App-Instanz kann dann festgelegt werden, welcher Virens Scanner verwendet werden soll:

App-Konfiguration für Virenschanner		
App	Virenschanner	Hilfe anzeigen
VirusScan	<div><input checked="" type="radio"/> Kommandozeile: <input type="text" value="/usr/bin/clamscan --no-summary %{file}"/></div> <div><input type="radio"/> Virenschanner aus Add-On: <input type="text" value="Cloudmersive"/></div>	

In diesem Fall soll der auf vielen Linux-Systemen verfügbare **Virenschanner ClamAV** verwendet werden. Der Aufruf soll dabei mit folgender Kommandozeile erfolgen:

```
/usr/bin/clamscan --no-summary %{file}
```

Der Platzhalter `%{file}` wird beim Aufruf durch den Dateinamen ersetzt.

Wurde wie oben beschrieben das API-Token für den Virenschanner von Cloudmersive festgelegt, kann diese Scanner ausgewählt werden:

App-Konfiguration für Virenschanner		
App	Virenschanner	Hilfe anzeigen
VirusScan	<div><input type="radio"/> Kommandozeile: <input type="text" value="/usr/bin/clamscan --no-summary %{file}"/></div> <div><input checked="" type="radio"/> Virenschanner aus Add-On: <input type="text" value="Cloudmersive"/></div>	

3.18.3 Testen der Virenschannerintegration

Zum Testen der Integration kann das **EICAR-Test-File** verwendet und beispielsweise in einem Service Desk-Formular hochgeladen werden. Das Formular sollte dann wie folgt den Fund der Schadsoftware anzeigen:

Anfrageformular

Anrede *

☐ Herr ☐ Frau

Vorname *

Nachname *

E-Mail-Adresse *

Betreff

Dateiupload



No file chosen

Virus gefunden.

4 Konfiguration des Datenmodells

Der Universal Messenger wird mit einem für die einfachen Standardanforderungen vorbereiteten Datenmodell ausgeliefert, das bei der Installation automatisch angelegt wird. Das Datenmodell kann mit den hier beschriebenen Konfigurationsdateien für die individuellen Anforderungen angepasst und erweitert werden. Bitte beachten Sie, dass die möglichen Erweiterungen des Datenmodells durch die zu dem Universal Messenger erworbene Lizenz beschränkt sein können. Für weitere Informationen zu den Lizenzmerkmalen lesen Sie bitte die Produktbeschreibung.

4.1 Überblick zum Datenmodell des Universal Messenger

Zu einem Eintrag werden in dem Universal Messenger interne Attribute, vordefinierte Attribute und die in der Konfigurationsdatei definierten zusätzlichen kundenspezifische Attribute gespeichert. Die internen Attribute dienen technischen Funktionen in dem Universal Messenger und können nur gelesen, aber nicht geschrieben oder verändert werden. Die vordefinierten Attribute sind in der Standardkonfiguration des Universal Messenger vorhanden und werden für die meisten Anwendungsfälle auch unverändert eingesetzt, können aber auch in der Attributkonfiguration geändert werden. Mit kundenspezifischen zusätzlichen Attributen kann das Datenmodell des Universal Messenger für die individuellen Anforderungen erweitert werden.

Die zu jedem Eintrag gehörende OID kennzeichnet den Eintrag in jedem Fall eindeutig und kann nachträglich nicht mehr geändert werden. Die OID wird im Normalfall nur zur internen Identifikation eines Eintrags verwendet und sollte nicht mit einem Benutzernamen verwechselt werden, der als zusätzliches Attribut definiert werden kann.

4.2 Objekt-relationales Mapping

Die Speicherung der Attribute in der SQL-Datenbank unterscheidet sich grundsätzlich für die drei Kategorien Primärattribute, Sekundärattribute (Overflow-Attribute) sowie für Attribute vom Typ Tabelle:

Für jeden Eintrag legt der Universal Messenger eine neue Zeile in einer SQL-Tabelle an. In dieser Tabelle werden internen Attribute (u.a. die UID) und Primärattribute als Spalten der Tabelle gespeichert. Der Zugriff auf die Primärattribute ist besonders schnell, da hierzu lediglich ein einfacher SQL-Befehl benötigt wird. Die Anzahl der Spalten einer SQL-Tabelle und somit die Anzahl der Primärattribute sind jedoch beschränkt, so dass für weitere Attribute auf die Sekundärattribute bzw. Overflow-Attribute zurückgegriffen werden muss.

Ein weiterer Nachteil der Primärattribute ist die feste SQL-Tabellenstruktur. Beim ersten initialen Systemstart legt der Universal Messenger die Tabellenstruktur mit den in der Konfigurationsdatei `additional.attributes` definierten Primärattributen an. Wenn ein Primärattribut später hinzugefügt, geändert oder gelöscht werden soll, ist jedoch ein manueller Eingriff des Administrators per SQL notwendig. Es wird deswegen empfohlen, vor allem die häufig in Suchen und zur Identifikation verwendeten Attribute als Primärattribute anzulegen.

Weniger häufig benötigte Attribute, die zudem bei vielen Einträgen nicht gefüllt sind, sollten als Sekundärattribute bzw. Overflow-Attribute angelegt werden. Alle Attribute in der Konfigurationsdatei `additional.attributes` werden als Overflow-Attribute angelegt, wenn sie nicht speziell als Primärattribut gekennzeichnet sind. Alle Overflow-Attribute werden in einer SQL-Tabelle mit nur wenigen Spalten gespeichert. Für jeden Eintrag und für jedes Overflow-Attribut wird eine Zeile in der SQL-Tabelle angelegt. Da alle Overflow-Attribute in einer gemeinsamen Spalte gespeichert werden, sind die Overflow-Attribute in der SQL-Tabelle nicht typisiert. Ein Vorteil der Overflow-Attribute ist die Unabhängigkeit von einem festen SQL-Schema, da Änderungen an der Attributkonfiguration des Universal Messenger keine Änderung am SQL-Schema erfordern. Für den Zugriff auf die Overflow-Attribute eines Eintrags sind jedoch einige komplexere SQL-Befehle notwendig, um die Zeile aus der Primärtabelle mit den möglicherweise zahlreichen Zeilen der Overflow-Tabelle zu verbinden. Der Zugriff auf Overflow-Attribute ist also bei weitem nicht so schnell wie auf Primärattribute und kann auch nicht in vergleichbarem Maß durch zusätzliche SQL Indizes beschleunigt werden.

Die dritte Kategorie bilden die Tabellenattribute, für die von dem Universal Messenger jeweils eine eigene SQL-Tabelle angelegt wird. Der Zugriff auf die Spalten eines Tabellenattributs ist also ähnlich schnell wie der Zugriff auf Primärattribute. Ähnlich wie bei den Primärattributen wird das SQL-Schema von dem Universal Messenger beim Systemstart automatisch mit neuen SQL-Tabellen erweitert, wenn der Universal Messenger beim Einlesen der Konfigurationsdatei `additional.attributes` feststellt, dass ein neues Tabellenattribut definiert wurde. Wenn ein Tabellenattribut geändert oder gelöscht werden soll, ist jedoch ein manueller Eingriff des Administrators per SQL notwendig.

4.3 Speicherung von Null-Werten oder leeren Werten


In der Datenbank werden nur die Werte gespeichert, die von false und der leeren Zeichenkette abweichen, sonst wird das Attribut in der Datenbank ganz nicht gesetzt. Dieses Verfahren reduziert den Speicherplatzbedarf, falls viele Attribute häufig nicht gesetzt sind.

Allerdings erfordert dieses Verfahren eine besondere Berücksichtigung von Suchanfragen nach Null-Werten oder leeren Werten. Falls Sie Operatoren wie z.B. `>` oder `>=` in Abfragen einsetzen, müssen Sie bedenken, dass Einträge mit Attributen mit einem Nullwert nicht als Treffer der Abfrage gefunden werden. Das betrifft auch einen Vergleich z.B. `< 2`, da hier die Einträge gleich dem Nullwert nicht in der Treffermenge enthalten sein werden. Beim Vergleich von zwei Attributen miteinander sollten Sie beachten, dass die Abfrage nicht für die Einträge ausgeführt wird, bei denen mindestens eines der verglichenen Attribute einen Nullwert hat. Weiterhin ist ein Vergleich mit den Nullwerten nicht möglich bzw. es müssen die dafür vorgesehenen Operatoren verwendet werden.

4.4 Änderungen der Attributkonfiguration

Beim erstmaligen Start nach der Installation liest der Universal Messenger die Attributkonfiguration aus der `additional.attributes` ein und legt das Datenbankschema in der SQL-Datenbank automatisch an. Aus diesem Grund sollten Sie die Attributkonfiguration möglichst vor dem erstmaligen Start des Universal Messenger anpassen, damit das Datenbankschema automatisch angelegt werden kann.

Spätere Änderungen in der `additional.attributes` können dazu führen, dass sich das von dem Universal Messenger in der Datenbank erwartete SQL-Schema ändert. Diese Änderungen können vom Universal Messenger im Allgemeinen **nicht automatisch** durchgeführt werden, sondern es ist ein manueller Eingriff durch einen Datenbankadministrator nötig. Die manuelle Anpassung des Datenbankschemas wird durch die Datei `cmsbs-work/dbschema.sql` unterstützt: In diese Datei wird beim Start hineingeschrieben, welche SQL-Objekte der Universal Messenger erwartet. Abweichungen vom tatsächlichen Modell können damit manuell bereinigt werden.

 Wir empfehlen, das Datenmodell in der Entwicklungsphase festzulegen. In dieser Phase können die Datenbank-Objekte leicht komplett gelöscht und beim Start wieder automatisch angelegt werden. Die nötigen **DROP**-Statements stehen als auskommentierte Kopiervorlage ebenfalls in `cmsbs-work/dbschema.sql` zur Verfügung (in der richtigen Reihenfolge).

Bei der Installation des produktiven Systems wird dann die finale Attributkonfiguration eingespielt und vom System automatisch in SQL-Objekte übersetzt.

Änderung am Attribut	Ausführung	Bemerkung
Neues Tabellenattribut	automatisch	Tabelle wird beim Start vom Universal Messenger angelegt.
Änderung der gültigen Werte	automatisch	Nicht Teil der SQL-Definition, erfordert deswegen keine Änderung am Modell.
Änderung des Typs eines Primärattributes	manuell	Spaltentyp der SQL-Tabelle muss geändert werden. <code>ALTER TABLE users MODIFY COLUMN p_my_column INT;</code>
Änderung der Maximallänge eines Primärattributes	manuell	Spaltentyp der SQL-Tabelle muss geändert werden: <code>ALTER TABLE users MODIFY COLUMN email VARCHAR(140);</code>
Änderung des Typs eines Sekundärattributes	(automatisch)	Keine Änderung auf SQL-Ebene, aber vorhandene gespeicherte Werte ggf. nicht mehr sinnvoll.
Neues Sekundärattribut	automatisch	Keine Änderung an SQL-Tabellen notwendig.
Neues Primärattribut	manuell	Spalte muss mit Namenspräfix "p_" angelegt werden: <code>ALTER TABLE users ADD COLUMN p_my_column VARCHAR(100);</code>

Neue Spalte in Tabellenattribut	manuell	Spalte muss mit Namenspräfix "c_" angelegt werden: <pre>ALTER TABLE a_my_table ADD COLUMN c_my_column VARCHAR(100);</pre>
Änderung von .isPrimary nach "true"	manuell	Spalte muss angelegt werden, vorhandene Daten kopieren und aus alter Stelle löschen: <pre>ALTER TABLE users ADD COLUMN p_my_column VARCHAR(100); UPDATE users u SET p_my_column=(SELECT valcol FROM users_attr WHERE u.oid=oid AND keycol='my_column'); DELETE FROM users_attr WHERE keycol='my_column';</pre>

Alle SQL-Statements sind hier beispielhaft für MySQL angegeben. Bei anderen Datenbanken kann die Syntax abweichen.

4.5 Bereitstellung für externe Schnittstellen

Das Datenmodell kann zur Umsetzung externer Schnittstellen in das Verzeichnis <UM_HOME>/cmsbs-work/config/* exportiert werden.

Parameter	Werte	Beschreibung
cmsbs.config.externalize	false true	Generierung Datenmodell für externe Schnittstellen Voreinstellung: false

4.6 Dateien der Attributkonfiguration

Das Datenmodell des Universal Messenger wird über eine oder mehrere Konfigurationsdateien definiert:

- `cmsbs-conf/additional.attributes`: Diese Datei wird bei der Neuinstallation angelegt und enthält Voreinstellungen zu diversen Attributen und Attributgruppen. Neue Releases beinhalten ggf. überarbeitete Versionen dieser Datei. Bei einem Update wird diese Datei jedoch nicht überschrieben, so dass die Attributkonfiguration gleichsam mit der Installation eingefroren wird.
- `cmsbs-conf/conf.d/*.attributes`: Alle Dateien in diesem Verzeichnis, deren Name auf `.attributes` endet, werden zusätzlich zur o.g. `additional.attributes`-Datei geladen und können diese projektspezifisch ergänzen oder in Teilen überschreiben.



Wir empfehlen, die Datei `additional.attributes` nicht zu verändern, sondern stattdessen alle Anpassungen am Datenmodell durch eigene Dateien unter `cmsbs-conf/conf.d/` vorzunehmen.

So sind die Anpassungen gegenüber dem Standarddatenmodell leichter zu überblicken und die spätere Übernahme von Erweiterungen infolge von Updates wird erleichtert.

4.6.1 Format der Konfigurationsdateien

Alle `.attributes`-Dateien liegen in einem erweitertem Java-Properties-Format vor.

Für die Konfiguration von Listen- bzw. Array-artigen Werten wurde das Format um die folgenden Notationen erweitert:

4.6.1.1 Wert einer Liste hinzufügen

```
entrytype.values[+] = new_type

# Alternative Schreibweise:
entrytype.values[] = new_type
```

4.6.1.2 Wert aus Liste entfernen

```
entrytype.values[-] = admin
```

4.6.1.3 Liste überschreiben

```
entrytype.values[!] = customer new_type
```

Setzt die Liste `entrytype.values` auf den Wert `customer, new_type` und überschreibt damit den vorherigen Wert und auch alle inkrementellen Anpassungen, die mittels `... [-]` oder `... [+]` vorgenommen wurden.

4.6.1.4 Andere `.attributes`-Datei inkludieren

Weitere `.attributes`-Dateien können anhand ihren Pfades inkludiert werden. Der Pfad muss dabei immer absolut oder relativ zum UM-Basisverzeichnis angegeben werden; z.B.:

```
include: cmsbs-conf/project-specific.attributes
```

4.7 Interne und vordefinierte Attribute

4.7.1 Interne Attribute

Die internen Attribute dienen technische Funktionen in dem Universal Messenger und können in der Attributkonfiguration nicht geändert werden.

Attribut	Beschreibung
----------	--------------

cmsbs.numemail	Anzahl der an den Eintrag verschickten E-Mails
cmsbs.hardbounces	Anzahl der für den Eintrag erhaltenen Hardbounces
cmsbs.softbounces	Anzahl der für den Eintrag erhaltenen Softbounces
cmsbs.autoresponds	Anzahl der für den Eintrag erhaltenen Auto-Responds
cmsbs.isadmin	"Backoffice-Login": Legt fest, ob für den betreffenden Eintrag der Login ins Backoffice erlaubt sein soll. (Früher: "Administrator?")
cmsbs.creation	Erstellungsdatum als Timestamp
cmsbs.lastchanged	Änderungsdatum als Timestamp

4.7.2 Vordefinierte Attribute

Die vordefinierten Attribute sind in der Standardkonfiguration des Universal Messenger vorhanden und werden für die meisten Anwendungsfälle auch unverändert eingesetzt, können aber auch in der Attributkonfiguration geändert werden.

Attribut	Länge	Beschreibung
uid	96	interne UID
cookie	128	Cookie zur Identifikation des Eintrags (kann nur gelesen werden)
email	128	E-Mail-Adresse
mobile	128	Mobilfunknummer (internationales Format ohne führendes Plus, z.B. 491726743) ⚠️ Sollte in neuen Projekten nicht mehr verwendet werden, da die automatische Formatierung nicht für alle internationalen Mobilnummern passt und die Formatierung nur für den SMS-Versand benötigt wurde. Wenn neben der Telefonnummer <i>phone</i> ein weiteres Attribut benötigt wird, sollte im Projektrahmen besser ein eigenes STRING-Attribut vorgesehen werden.
salut	128	Anrede ('male', 'female', 'family' oder 'company' für männlich, weiblich, Familie oder Firma)
title	256	Titel
firstname	512	Vorname
lastname	32	Nachname
company	128	Firmenname
street	128	Straße und Hausnummer
region	512	Bundesland oder Region, im Ausland teilweise notwendig, z.B. in USA für Bundesstaat
zip	512	Postleitzahl
city	64	Stadt
country	512	Land

lang	512	Sprache
birthday	512	Geburtsdatum
phone	64	Telefonnummer
fax	512	Telefaxnummer
homepage	512	URL zur Website
admin_role	512	Rolle (für Administratoren)
html	1	'1', falls der Abonnent E-Mails im HTML-Format erhalten möchte, sonst '0'
password	200	Passwort (kann nur geschrieben werden)

4.7.3 Änderung von vordefinierten Attributen

Wenn nicht nur neue kundenspezifische Attribute angelegt, sondern auch vordefinierte Attribute geändert werden sollen, sollte die Standardkonfiguration des Universal Messenger per Hilfsprogramm in eine Konfigurationsdatei geschrieben werden. Aufbauend auf dieser Konfiguration können Änderungen leichter vorgenommen werden. Bitte wechseln Sie hierzu in das Verzeichnis `UM/scripts/` und führen dort folgende Kommandozeile aus:

```
./userTool.sh ../cmsbs-conf/cmsbs.properties -attribs
```

für Linux bzw. für Windows

```
.\userTool.bat ..\cmsbs-conf\cmsbs.properties -attribs
```

Dieser Kommandozeilenaufwurf schreibt die aktuelle, vollständige Konfiguration der Attribute und Attributgruppen in die Datei `cfg.log`. Diese Datei ist recht umfangreich, es müssen jedoch nur die Abschnitte in Ihre angepasste `additional.attributes` übernommen werden, die Sie ändern wollen.

4.8 Kundenspezifische Attribute

Die in der Standardkonfiguration zu jedem Eintrag gespeicherten Daten können für die Anzeige konfiguriert und um zusätzliche Attribute erweitert werden. Je nach Edition des Universal Messenger können Sie zwei oder beliebig viele zusätzliche Attribute definieren. Weitere Angaben können Sie der Produktbeschreibung entnehmen, bei Rückfragen wenden Sie sich bitte an unseren Support.

Mit dem Universal Messenger wird eine Beispielkonfiguration ausgeliefert, die eine gute Ausgangsbasis darstellt und die Sie für Ihre eigenen Anforderungen anpassen können. In dieser Beispieldatei sind auch alle vordefinierten Attribute enthalten, die zwar nicht grundsätzlich verändert, aber für die Anzeige angepasst werden können.

4.8.1 Konfiguration zusätzlicher Attribute

Die in dem Universal Messenger vordefinierten Attribute können mit zusätzlichen kundenspezifischen Attributen erweitert werden. Die Konfiguration erfolgt normalerweise in der Konfigurationsdatei `<UM_HOME>/cmsbs-conf/additional.attributes`, wobei der Name der Konfigurationsdatei mit der folgenden Option in der zentralen Konfigurationsdatei bei Bedarf beliebig gesetzt werden kann:

```
cmsbs.additional.attributes = <Dateiname mit Pfad>
```

Die Konfiguration kann zur besseren Übersichtlichkeit auf mehrere Dateien verteilt werden, die dann mit der `include`-Anweisung zusammengeführt werden. Der in der `include`-Anweisung angegebene Dateiname wird relativ zu `<UM_HOME>` ausgewertet.

```
include: <Dateiname mit Pfad>
#oder: include = <Dateiname mit Pfad>
#oder: include.<Index> = <Dateiname mit Pfad>
```

```
additional.attributes = <Liste der zusätzlichen Attributnamen>
```

Durch Leerzeichen getrennte Liste der (internen) Namen der zusätzlichen Attribute oder Angabe im mehrzeiligen Format:

```
additional.attributes.1 = <Attributname>
```

```
additional.attributes.2 = <Attributname>
```

Eine weitere Alternative ist die Angabe ohne Index:

```
additional.attributes[] = <Attributname>
```

bzw. seit Release 7.45.0:

```
additional.attributes[+] = <Attributname> Seit 7.45.0 können Attribute mit folgender Notation entfernt werden: additional.attributes[-] = <Attributname>
```

Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten. Für jedes zusätzliche Attribut müssen die folgenden Optionen angegeben werden. Die vordefinierten Attribute können teilweise auch mit diesen Optionen angepasst werden, müssen aber nicht in der Liste der Attributnamen angegeben werden.

```
<Attributname>.type = <Attributtyp>
```

Als Attributtyp können die folgenden Typen für eigene zusätzliche Attribute verwendet werden:

STRING	Zeichenkette bis max. 512 Zeichen
--------	-----------------------------------

TEXT	beliebig lange Zeichenkette, die maximale Länge wird durch die verwendete Datenbank bestimmt
HTML	beliebig lange Zeichenkette, die maximale Länge wird durch die verwendete Datenbank bestimmt, die Verwendung von HTML-Elementen kann eingeschränkt werden
INTEGER	ganze Zahl
REAL	Fließkommazahl (experimentell)
BOOLEAN	Wahrheitswert mit 'true' oder 'false'
TIMESTAMP	Datum und Uhrzeit im Format 'YYYY-MM-DD hh:mm:ss' oder 'YYYY-MM-DD 00:00:00' ohne Uhrzeit
DATE	Datum ohne Uhrzeit (entspricht intern TIMESTAMP)
PHONE	Telefonnummer
EMAIL	E-Mail-Adresse
MOBILE	Mobilfunknummer
COOKIE	eindeutiger Cookie
TABLE	Tabelle bzw. Liste von Attributen
REFERENCE	Referenz auf einen anderen Eintrag
ATTACHMENT	Datei
WIZARD	Pseudo-Attribut für die Verwendung von individuellen CSE-Wizards

Für die vordefinierten Attribute werden weitere Attributtypen verwendet, die aber nicht für zusätzliche Attribute verwendet werden können:

UID	interne UID
PASSWORD	Passwort

```
<Attributname>.isChecked = <true|false>
```

Die Verifikation des Attributwerts auf den erlaubten Wertebereich (bei Referenzattributen z.B. der als Referenz angegebenen ID) vor dem Speichern kann deaktiviert werden, so dass Referenzen auf möglicherweise (noch) nicht vorhandene Einträge gesetzt werden können. (Default: "true")

```
<Attributname>.isPrimary = true|false
```

Bei `true` wird das Attribut als Primärattribut deklariert und in der Datenbanktabelle `users` gespeichert. Das bedeutet, dass die Tabelle `users` eine eigene Spalte für dieses Attribut enthalten muss. Diese Spalte kann auch nachträglich per SQL-Kommando hinzugefügt werden. Für MySQL-Datenbanken lautet der entsprechende Aufruf:

```
ALTER TABLE users ADD p_<Attributname> VARCHAR (1);
```

```
<Attributname>.sqlName = <Spaltenname in der Datenbanktabelle "users">
```

Der SQL-Spaltenname eines Attributs kann geändert werden, wenn es sich bei dem Attribut um ein Primärattribut handelt. Der Defaultwert ist `p_<Attributname>`.

```
<Attributname>.search = <Suchoperator>
```

Gibt den Suchoperator an, welcher in der Benutzeroberfläche bei der einfachen Suche für das jeweilige Attribut verwendet wird. Mögliche Suchoperatoren sind: `equals`, `iequals`, `contains`, `icontains`, `startsWith`, `istartsWith`, `endsWith` und `iendsWith`, wobei `i` für die caseinsensitive Variante steht. Der Defaultwert ist `equals`.

```
<Attributname>.title.<Sprache> = <Text>
```

Titel des Attributs, der in der Benutzeroberfläche des Universal Messenger für die angegebene Sprache angezeigt wird.

```
<Attributname>.showClient = true|false
```

Bei `true` kann dieses Attribut über die API abgefragt werden, bei `false` bleibt es für die API unsichtbar.

```
<Attributname>.fromClient = true|false
```

Bei `true` kann dieses Attribut über die API geschrieben werden, bei `false` ist ein Schreibzugriff nicht möglich.

```
<Attributname>.fromGui = true|false
```

Bei `true` kann dieses Attribut über die Benutzeroberfläche für Administratoren geschrieben werden, bei `false` ist ein Schreibzugriff nicht möglich. Ein Attribut kann über diese Option in der Benutzeroberfläche angezeigt, aber vor einer Änderung geschützt werden.

```
<Attributname>.showFormatter = true|false
```

Bei `true` steht dieses Attribut als Personalisierungsvariable zur Verfügung.

```
<Attributname>.isUnique = true|false
```

Bei `true` muss der Wert dieses Attributs eindeutig einen Abonnenten kennzeichnen und darf nur einmal vergeben werden.

```
<Attributname>.display.mode = <default|checkbox>
```

Je nach Attributtyp und weiteren Einstellungen kann mit dieser Option die Darstellung des Attributs in der Benutzeroberfläche konfiguriert werden.

Neben dem Standardwert `default` ist derzeit die Einstellung `checkbox` implementiert, mit der eine Mehrfachauswahl als eine Liste von Checkboxes dargestellt wird. Die Einstellung `checkbox` darf nur mit der Einstellung `<Attributname>.type = STRING` zusammen verwendet werden.

```
<Attributname>.display.width = <Zahl>
```

Breite des Eingabefeldes in der Benutzeroberfläche für Administratoren in Anzahl der Zeichen. Die Breite des Eingabefeldes hat keinen Einfluss auf die maximal eingebbare Anzahl der Zeichen.

```
<Attributname>.display.height = <Zahl>
```

Höhe des Eingabefeldes in der Benutzeroberfläche für Administratoren in Anzahl der Zeilen. Die Höhe des Eingabefeldes hat keinen Einfluss auf die maximal eingebbare Anzahl der Zeilen.

```
<Attributname>.default = <Text>
```

Optional Standardwert des Attributs beim Anlegen eines neuen Abonnenten, der mit einer eigenen Eingabe überschrieben werden kann.

```
<Attributname>.resetIfChanged = <Liste der Attributnamen>
```

Falls eines der Attribute aus der durch Leerzeichen getrennten Liste der Attributnamen geändert wird, wird auch dieses Attribut auf den Standardwert zurückgesetzt. Dies kann z.B. für ein Validierungskennzeichen verwendet werden, das bei einer Änderung der Adresse des Abonnenten automatisch zurückgesetzt werden soll.

Wenn dieses Attribut über die API geschrieben wird und gleichzeitig ein Schreibzugriff auf eines der in der Liste angegebenen Attribute den Reset auslösen würde, wird der über die API geschriebene Wert dennoch übernommen.

```
<Attributname>.values.<Position> = <interner Wert>
```

oder

```
<Attributname>.values[] = <interner Wert>
```

Um die möglichen Werte des Attributs für die Attributtypen `STRING` und `INTEGER` optional auf einen festen Wertebereich zu beschränken, werden zunächst die internen Werte und ihre Position definiert. Die internen Werte dürfen keine Leerzeichen am Anfang oder Ende enthalten. In den folgenden Optionen müssen dann die Titel dieser Werte in den verschiedenen Sprachen angegeben werden. Wird die zweite Variante verwendet, ist die Position der Werte durch die Reihenfolge in der Datei bestimmt.

```
<Attributname>.value.<interner Wert>.<Sprache> = <Titel>
```

Titel des festen Wertes, der in der Benutzeroberfläche des Universal Messenger für die angegebene Sprache angezeigt wird. Falls kein Titel angegeben wird, wird der interne Wert selbst in der Benutzeroberfläche angezeigt.

```
<Attributname>.valueDistribution = group | unique | amount
```

Bestimmt, wie das Histogramm in der grafischen Segmentierung angezeigt wird.

Die möglichen Einstellungen sind:

- **group**: Attribute, die bei mehreren Einträgen identisch gesetzt werden können und den Einträgen eine bestimmte Eigenschaft zuordnen. Einträge mit dem selben Wert bilden eine Gruppe, z.B. alle Personen aus "Berlin" oder Kunden mit der Zahlungsart "Kreditkarte". Per default wird diese Einstellung für Attribute mit eingeschränktem Wertebereich bzw. vom Typ BOOLEAN gesetzt.
- **amount**: Attribute, deren Werte aufsummiert werden können, z.B. die Höhe eines Warenkorbs oder die Anzahl der Bestellungen. Per default wird diese Einstellung für Attribute vom Typ REAL oder INTEGER gesetzt, wenn die Wertebereiche nicht eingeschränkt sind.
- **unique**: Attribute, die für jeden Eintrag eindeutig sind oder vergleichbar interpretiert werden sollen, z.B. ist ein Username immer eindeutig und der Nachname einer Person ist zwar in der Regel nicht eindeutig, soll aber in den Auswertungen vergleichbar interpretiert werden, weil normalerweise alle "Meier" nicht zu einer Gruppe zusammengefasst werden sollen. Per default wird diese Einstellung für Attribute mit nicht eingeschränktem Wertebereich verwendet, für die nicht bereits eine andere Einstellung getroffen hat.

Beispiel: Attribute, bei denen davon ausgegangen werden kann, dass es endlich verschiedene Werte gibt (z.B. country, city), der Wertebereich jedoch nicht eingeschränkt ist, sollten die Eigenschaft "group" erhalten, um besser durchsucht und dargestellt werden zu können.

```
<Attributname>.includeInHistoryLog = true|false
```

Legt fest, ob der Wert dieses Attributes beim Löschen des Eintrags im History-Log archiviert werden soll.

Hinweis: [Das History-Log muss aktiviert sein](#), damit eine Archivierung beim Löschen erfolgt.

Seit Version 7.32.3 werden standardmäßig alle Attribute archiviert. Bei Attributen des Typs ATTACHMENT wird lediglich die ID des Dateiinhalts (Blob-ID) archiviert, nicht jedoch der Dateiinhalt selbst.

4.8.1.1 Freie Attribut-Konfigurations-Parameter

Zu den Attributen können zusätzliche freie Informationen gespeichert werden, die über die CSE (Core Scripting Engine) und aus der CSE und UMConfig gelesen werden können. Diese Daten können für individuelle Funktionen genutzt werden und werden von dem Universal Messenger selbst nicht verwendet.

Diese Daten werden intern als JSON gespeichert und können in der Attributkonfiguration entweder als JSON-Datei oder als einzelne Key-Value-Paare angegeben werden.

```
<Attributname>.config = <Datei im JSON-Format>
```

Eine Datei im JSON-Fomat kann z.B. in cmsbs-conf/file.json abgelegt und hier angegeben werden, um den Inhalt als freie Attribut-Konfigurations-Parameter zu dem Attribut zu speichern.

```
<Attributname>.config.<Key> = <Value>
```

In der additional.attributes-Datei können z.B. folgende Daten zu einem Attribut definiert werden:

```
<Attributname>.config.myValues[] = {"key": "a", "value": {"de": "Wert A", "en": "B"}}
<Attributname>.config.myValues[] = {"key": "b", "value": "Wert B"}
<Attributname>.config.foo = "bar"
<Attributname>.config.struct.blubb = "Noch ein Wert"
```

In der CSE können diese Daten wie folgt ausgelesen werden:

```
UM.config.getAttribute('ATTR').config ->
{
  "myValues": [
    {"key": "a", "value": {
      "de": "Wert A",
      "en": "B"
    }},
    {"key": "b", "value": "Wert B"}
  ],
  "foo": "bar",
  "struct": {
    "blubb": "Noch ein Wert"
  }
}
```

Die Konfiguration kann in Adminrollen und Entrytypes modifiziert werden. Dabei werden die Änderungen dann nach folgender Reihenfolge zusammengeführt: additional.attributes -> Adminrolle -> EntryType.

Diese Möglichkeiten gelten analog auch für Attributgruppen.

4.8.1.2 Reservierte Wörter, die nicht als Attributnamen verwendet werden dürfen

Die folgenden Schlüsselwörter dürfen weder in Groß- noch in Kleinschreibung für Attributnamen verwendet werden:

- or
- and
- like
- ilike
- defined

- undefined
- true
- false
- asc
- ascending
- desc
- descending
- having
- channel und alle Wörter, die mit channel beginnen
- permission und alle Wörter, die mit permission beginnen

4.9 Attributtypen

4.9.1 STRING

Mit dem Attributtyp STRING können Zeichenketten als Overflow-Attribut in der Standardkonfiguration bis zu einer Länge von 512 Byte gespeichert werden. Sollen längere Zeichenketten gespeichert werden, kann die Spaltenbreite der SQL-Tabelle mit der Konfigurationsoption `cmsbs.database.user.valcolSize` geändert werden. Nach einer Änderung muss das SQL-Schema manuell angepasst werden, es erfolgt keine automatische Schemaanpassung.

Wie im vorhergehenden Abschnitt beschrieben, kann der Attributtyp STRING neben der Speicherung beliebiger Zeichenketten auch für die Speicherung vordefinierter Stringkonstanten verwendet werden, indem die Auswahl möglicher Werte durch die Konfigurationsoption `<Attributname>.values` auf einen festen Wertebereich beschränkt wird.

Ohne weitere Einstellungen erlaubt der Universal Messenger die Auswahl genau eines Wertes aus dem festen Wertebereich. Soll die Auswahl mehrerer Werte im Sinne einer Mehrfachauswahl erlaubt werden, muss als Höhe des Eingabefelds ein Wert größer als eins angegeben werden, z.B.

`<Attributname>.display.height = "2"`. Der Wert gibt dabei weiterhin die Höhe des Eingabefelds vor und beschränkt nicht etwa die Anzahl der wählbaren Werte.

Die gewählten Werte werden intern als String in Form einer Pipe ("|")-separierten Liste gespeichert, wobei der String mit einer Pipe beginnt und abgeschlossen wird. Pipe-Symbole innerhalb der Werte werden mit einem einfachen Anführungszeichen escaped, das einfache Anführungszeichen selbst wird auch mit einem einfachen Anführungszeichen escaped. Die leere Liste und die Liste mit nur dem leeren Wert werden als vollständig leer betrachtet und nicht abgespeichert.

Für Abfragen auf eine Mehrfachauswahl können die für Strings geeigneten Operatoren `LIKE` und `ILIKE` verwendet werden. Eine Abfrage, ob ein Wert in der Mehrfachauswahl gesetzt ist, könnte funktion `LIKE '%|Manager|%'` lauten. Da diese Abfrage sehr häufig benötigt wird, kann sie auch durch funktion `= 'Manager'` abgekürzt werden.

4.9.2 TEXT

Mit dem Attributtyp TEXT können sehr lange Zeichenketten in dem Universal Messenger gespeichert werden. Die maximale Länge der Zeichenkette wird durch die verwendete Datenbank bestimmt, siehe dazu die Tabelle der Datenbanken.

Folgende Einschränkungen sind dabei zu beachten:

- TEXT-Attribute müssen entweder Primärattribute oder Spalten in einem Tabellenattribut sein. Sie können nicht als Sekundär- bzw. Overflow-Attribute verwendet werden.
- Die Abfragesprache des Universal Messenger erlaubt für TEXT-Attribute nur die Operatoren `defined` und `undefined`.
- Die Speicherung sehr langer Werte (mehr als ein paar hundert Kilobyte) kann die Performance beeinträchtigen.
- Die Maximallänge eines TEXT-Attributes hängt von der verwendeten Datenbank ab (siehe unten).

TEXT-Attribute können in der Benutzeroberfläche per Textarea bearbeitet werden.

Datenbank	verwendeter Datentyp	max. Länge
Oracle	NCLOB	ca. 2 Mrd. Zeichen
MySQL	LONGTEXT	ca. 2 Mrd. Zeichen
MSSQL 2000	NTEXT	ca. 1 Mio. Zeichen
MSSQL ab 2005	NTEXT	ca. 2 Mrd. Zeichen
DB2	DBCLOB	ca. 1 Mrd. Zeichen
Postgres	TEXT	"unbegrenzt"

Die angegebenen Maximallängen gelten jeweils für die Datenbanken an sich. Da der Universal Messenger bei der Verarbeitung von TEXT-Attributen die Werte als Strings im Speicher behandelt und kein Streaming verwendet, wird die praktisch nutzbare Maximallänge in der Praxis eher durch den verfügbaren Heap-Speicher der Java-VM begrenzt als durch die Datenbank.

Zum Anlegen eines neuen Primärattributes `textfield` in einer bereits vorhandenen Users-Tabelle nutzen Sie das jeweilige SQL-Statement:

Datenbank	SQL
Oracle	<code>ALTER TABLE users ADD p_textfield NCLOB;</code>
MySQL	<code>ALTER TABLE users ADD p_textfield LONGTEXT</code>
MSSQL	<code>ALTER TABLE users ADD p_textfield NTEXT;</code>
DB2	<code>ALTER TABLE users ADD p_textfield DBCLOB;</code>
Postgres	<code>ALTER TABLE users ADD p_textfield TEXT;</code>

4.9.3 HTML

Der Attributtyp HTML leitet sich vom Typ TEXT ab. Die Einschränkungen sind dabei gleich und die maximale Länge ist ebenfalls von der verwendeten Datenbank abhängig (siehe TEXT).

Durch einen integrierten Filter (jTidy) wird sichergestellt, dass in HTML-Attributen stets nur valides XHTML gespeichert wird. `<Attributname>.valueFilter = HtmlAttributeFilter({whiteList: {strong: [], em: [], a: ['href']}})`

Mit dieser Einstellung kann genau festgelegt werden, welche HTML-Elemente mit welchen Attributen verwendet werden dürfen. Alle anderen Elemente und Attribute werden vor dem Speichern entfernt. Standardmäßig sind folgende Elemente zulässig:

```
<b>, <p>, <i>, <strong>, <em>, <s>, <ol>, <ul>, <li>, <a href="...">, <br/>
```

Die CSE-Klasse `HtmlAttributeFilter` kann bei Bedarf auch als Basis für projektspezifische HTML-Filter verwendet werden. `<Attributname>.editor.mode = inline`

Soll zur Bearbeitung des Attributs in der Benutzeroberfläche des Universal Messenger keine normale Textarea, sondern die Inline-Version des [CK-Editor](#) verwendet werden (empfohlen), muss

`<Attributname>.editor.mode` auf `inline` gesetzt werden. `<Attributname>.editor.config =`

```
{removeButtons: 'Copy,Paste,Cut,PasteText,Underline'}
```

Mit dieser Option können Sie den [CK Editor konfigurieren](#), beispielsweise unnötige Buttons nicht anzeigen.

Beispiel einer Konfiguration

```
htmltext.type = HTML
htmltext.title.de = "HTML-Feld"
htmltext.title.en = "HTML field"
htmltext.isPrimary = true

# optionale Angaben
htmltext.editor.mode = inline
htmltext.editor.config = {removeButtons: 'Copy,Paste,Cut,PasteText,Underline'}
htmltext.valueFilter = HtmlAttributeFilter({whiteList: {strong: [], em: [], a: ['href']}})
```

4.9.4 TIMESTAMP und DATE

Die beiden Attributtypen `TIMESTAMP` und `DATE` werden zum Speichern von sekundengenauen Zeitstempeln verwendet. Wobei beim Typ `DATE` als Zeit "00:00:00" intern gespeichert wird, bei der Anzeige im Universal Messenger aber nur das Datum zu sehen ist. Das Format für `TIMESTAMP` ist 'YYYY-MM-DD|hh:mm:ss', das für `DATE` ist 'YYYY-MM-DD|00:00:00' (ohne Uhrzeit).

Beim Schreiben des Wertes kann die Variable `{now}` für die aktuelle Uhrzeit bzw. das aktuelle Datum (des Servers) verwendet werden, sie wird beim Speichern gesetzt.

```
e.set('dateAttribute', '{now}');
```

Die Variable kann in der CSE verwendet werden.

4.9.5 EMAIL

Der Attributtyp EMAIL wird für die Speicherung von E-Mail-Adressen verwendet. Mit einem Attribut vom Typ EMAIL kann ein Eintrag eindeutig identifiziert werden.

Die Standardattributkonfiguration enthält bereits ein EMAIL-Attribut mit dem Namen "email". Dieses Attribut wird standardmäßig als Empfängeradresse beim E-Mail-Versand genutzt.

```
<Attributname>.isUnique = true|false
```

Gibt an, ob der Wert des Attributes global eindeutig sein soll. Für das Attribut " email " ist das standardmäßig der Fall.

Beim Setzen des Wertes eines EMAIL-Attributes werden Nicht-ASCII-Zeichen (z.B. deutsche Umlaute) automatisch im Local- bzw. Domainteil nach [RFC 3490](#) ("Punycode") konvertiert.

4.9.6 COOKIE

Der Attributtyp COOKIE wird für die eindeutige Identifikation von Einträgen verwendet. Mit einem Attribut vom Typ COOKIE kann ein Eintrag eindeutig identifiziert werden, anders als bei der UID sind die Werte aber nicht erratbar.

Die Standardattributkonfiguration enthält bereits ein COOKIE-Attribut mit dem Namen "cookie".

Zu einem Attribut vom Typ COOKIE können die Zeichen definiert werden, aus denen der Identifikationscode automatisch generiert werden soll. Im Normalfall soll der Identifikationscode eindeutig sein, was durch die Option `isUnique` angegeben wird. Falls die Eindeutigkeit durch einen Vergleich mit den vorhandenen Werten bei der Erstellung explizit geprüft werden soll, muss zusätzlich die Option `isChecked` angegeben werden.

```
<Attributname>.alphabet = <Zeichenauswahl>
```

Angabe der Zeichen (ohne Leerstellen als Zeichenkette angeben), aus denen der Identifikationscode automatisch generiert werden soll.

```
<Attributname>.display.width = <Länge>
```

Länge des automatisch generierten Identifikationscodes.

```
<Attributname>.isUnique = true|false
```

Es soll ein eindeutiger Identifikationscode generiert werden.

```
<Attributname>.isChecked = true|false
```

Die Eindeutigkeit des Identifikationscodes soll bei der Erstellung explizit geprüft und sichergestellt werden.

```
<Attributname>.allowEmpty = true|false
```

Soll ein leeres Custom-Cookie-Feld beim Speichern nicht automatisch gefüllt werden, muss `allowEmpty` auf `true` gesetzt werden. Beliebig viele Werte des Attributs können leer sein, auch wenn `isUnique` auf `true` gesetzt ist.

4.9.7 PASSWORD

Der Attributtyp `PASSWORD` wird für Passwörter benutzt und bietet die Möglichkeiten, sie als Einweghashwert (d.h. verschlüsselt) zu speichern.

Attribute des Typs `PASSWORD` (also auch das Standardattribut `password`) haben die Option `hashAlgorithm`. Diese Option kann die Werte `""`, `MD5`, `SHA1`, `SSHA` oder `PBKDF2_1` annehmen und gibt den Hashing-Algorithmus an, der bei der Speicherung der Passwörter verwendet werden soll.

Der Universal Messenger speichert die Benutzerpasswörter standardmäßig mit `PBKDF2_1` (in früheren Versionen mit `SHA1`) verhasht in der Datenbank. Das Updaten einer bestehenden Installation von z.B. `SHA1` auf `PBKDF2_1` ist nicht ohne Verlust aller Passwörter möglich.

Bei der Option `PBKDF2_1` werden Passwörter mit `PBKDF2WithHmacSHA512` / 64-Bit random seed / 512-Bit Schlüssellänge verhasht gespeichert. Die Anzahl der Iterationen zur Passwortverhashung kann in der Konfigurationsdatei `cmsbs.properties` mit der Option `cmsbs.password.hashIterations` angegeben werden.

Die Option `encoding` gibt an, welches Charset beim Hashen von Passwörtern verwendet wird. Der Default-Wert ist `UTF-8`.

Die Option `allowEmpty` gibt an, ob leere Passwörter zugelassen sind. (Leeres Passwort bedeutet: In der Datenbank steht ein Leerstring und ein Login ist nicht möglich.)

Klartext (keine Verschlüsselung)

```
password.hashAlgorithm = ""
```

MD5

```
password.encoding = "UTF-8"  
password.hashAlgorithm = "MD5"  
password.dbSize = 32  
password.allowEmpty = true
```

SHA1 (Standardkonfiguration)

```
password.encoding = "UTF-8"  
password.hashAlgorithm = "SHA1"  
password.dbSize = 40  
password.allowEmpty = true
```

SSHA

```
password.encoding = "UTF-8"  
password.hashAlgorithm = "SSHA"  
password.dbSize = 32  
password.allowEmpty = true
```

Mit SSHA gehashte Passwörter sind kompatibel zu LDAP (4 Byte random Salt).

PBKDF2_1

```
password.encoding = "UTF-8"  
password.hashAlgorithm = "PBKDF2_1"  
password.dbSize = 200  
password.allowEmpty = true
```

Der Vergleich eines eingegebenen Passworts mit dem in der Datenbank gespeichert Passwort ist nicht mehr per Abfragesprache möglich. Folgendes Query wird keine Treffer liefern:

uid="admin" and password="admin"

Stattdessen kann in der CSE die Methode `testPassword()` der Entry-Klasse verwendet werden.

4.9.7.1 Passwortqualität

Seit Release 7.49.0 können Mindestanforderungen an die Passwortqualität festgelegt werden, die jeweils beim Setzen eines neuen Wertes erzwungen werden.

Der numerische Wert `minimumEntropy` gibt dabei an, wie zufällig bzw. schwer erratbar das Passwort mindestens sein soll. Höhere Werte stehen dabei für "bessere" Passwörter.

Wird für `minimumEntropy` ein Wert größer 0 angegeben, so wird ein neu eingegebenes Passwort u.a. gegen eine Liste von besonders häufig verwendeten Passwörtern geprüft. Ebenso wird es auf leicht zu erratende Wiederholungen, Datums- und Jahreszahlen und ähnliches geprüft.

Mit `minimumLength` kann außerdem eine bestimmte Mindestlänge erzwungen werden.

Bei einer Neuinstallation werden die folgenden Einstellungen in `cmsbs-conf/additional.attributes` eingetragen:

```
password.minimumEntropy = 32
password.minimumLength = 8
```

Ein Wert von 32 für `minimumEntropy` resultiert beispielsweise in der folgenden Bewertung einiger exemplarischer Passwörter:

- OK:
 - 72hshf78
 - Ja8iujaYq
 - dude-purify-eggplant
 - hug-chest-deftly
- zu schwach:
 - qwerty.123
 - 1234567890
 - Ja8iu222
 - laracroft1977
 - aba-aba-aba-aba-aba-aba-aba-aba

4.9.8 TABLE

Für die Nutzung von Attributen vom Typ TABLE ("Tabellenattribute") wird eine zusätzliche Lizenz benötigt, die je nach gewählter Edition im Lieferumfang des Universal Messenger enthalten ist. Weitere Angaben können Sie der Produktbeschreibung entnehmen, bei Rückfragen wenden Sie sich bitte an unseren Support.

Mit einem Attribut vom Typ TABLE können in einem Attribut Tabellen bzw. Listen gespeichert werden. Als Spalten der Tabelle können alle möglichen Attribute konfiguriert werden, so dass in einer Tabelle z.B. pro Zeile die Bestellungen des Kunden bestehend aus Datum, Artikelanzahl und Betrag gespeichert werden können.

Attribute vom Typ TABLE werden von dem Universal Messenger in einem klassischen relationalen Datenbankschema gespeichert, so dass auf diese Attribute besonders gut mit externen SQL-Tools zugegriffen werden kann. Aus dem gleichen Grund sind allerdings Änderungen an der Konfiguration eines Attributs vom Typ TABLE nicht ohne einen manuellen Eingriff per SQL möglich, da von dem Universal Messenger für jedes Attribut vom Typ TABLE eine neue SQL-Tabelle angelegt wird und in dieser bei einer Änderung der Konfiguration die einmalig angelegten Spalten nicht automatisch geändert werden können.

Jede Zeile der Tabelle ist durch eine eindeutige Zeilennummer gekennzeichnet, die auch beim Löschen einzelner Zeilen aus der Tabelle beibehalten wird. Damit die Zuordnung der Zeilennummern erhalten bleibt, dürfen aber keine Zeilen eingefügt oder verschoben werden, ggf. muss dies durch die Konfigurationsoptionen `canInsertRow` und `canMoveRow` ausgeschlossen werden.

```
<Attributname>.members = <Liste der Spaltennamen>
```

oder

```
<Attributname>.members[] = <Spaltenname>
```

Durch Leerzeichen getrennten Liste der (internen) Namen der Spalten, die für das Attribut vom Typ TABLE angelegt werden sollen. Wird die zweite Variante verwendet, ist die Position der Spalten durch die Reihenfolge in der Datei bestimmt. Für jede Spalte werden die Attribute mit den bereits beschriebenen Konfigurationsmöglichkeiten definiert.

```
<Attributname>-><Spaltenname>.type = <Attributtyp>
<Attributname>-><Spaltenname>.title.<Sprache> = <Text>
<Attributname>-><Spaltenname>.showClient = true|false
```

Die Konfiguration der Spalten der Tabelle entspricht den normalen Attributen, ebenso können die Rechte und Rollen auf die Spalten einer Tabelle übernommen werden.

```
<Attributname>-><Spaltenname>.display.mode = <default|popup>
```

Bei sehr großen Tabellen ist es sinnvoll diese über ein PopUp-Fenster zu öffnen, da die Ladezeit beim Ansehen und Bearbeiten eines Eintrags sonst unnötig lange dauern kann. Wird der Wert `popup` verwendet, wird statt der Tabelle selbst ein Link zur Tabelle und deren Anzahl der Zeilen dargestellt.

```
<Attributname>-><Spaltenname>.dbSize = <Anzahl in Bytes>
```

Mit dieser Option kann die jeweilige Spaltenbreite in der Tabelle definiert werden, die von dem Universal Messenger für jedes Attribut vom Typ TABLE neu angelegt wird. Die Breite der Spalte bestimmt die Länge der Werte, die maximal gespeichert werden können. Falls diese Option nicht angegeben wird, wird die Standardeinstellung aus `cmsbs.database.user.valcolSize` übernommen.

```

<Attributname>.canInsertRow = true|false
<Attributname>.canMoveRow = true|false
<Attributname>.canDeleteRow = true|false
<Attributname>.canAppendRow = true|false

```

Mit diesen Konfigurationsoptionen können die Bearbeitungsfunktionen für Tabellen in der Benutzeroberfläche des Universal Messenger eingeschränkt werden. Diese Einstellungen gelten global und können nicht für einzelne Rollen geändert werden.

4.9.9 REFERENCE

Der Attributtyp REFERENCE kann verwendet werden, wenn in einem Attribut eine Referenz auf einen anderen Eintrag gespeichert werden soll.

In der Datei `additional.attributes` können für ein REFERENCE-Attribut die folgenden zusätzlichen Eigenschaften definiert werden:

```
<Attributname>.reference.foreignKey = <Attributname>
```

Name des Attributes, dessen Wert als Referenz gespeichert werden soll (optional, Default: "uid")

```
<Attributname>.reference.selector = dropdown|text|autocompleter
```

Art der Auswahl ("dropdown", "text" oder "autocompleter"). Bitte beachten Sie, dass die Auswahl per Dropdownliste nur verwendet werden sollte, wenn die Anzahl der möglichen Referenzpartner relativ klein ist.

```
<Attributname>.reference.selectorQuery = <Query>
```

Query, welche die Menge der möglichen Einträge einschränkt, die referenziert werden können, z.B. "entrytype = 'customer'"

```
<Attributname>.reference.selectorSort = <Attributname>
```

Name des Attributes, nach dem die Einträge in der Dropdownbox sortiert werden sollen oder displayValue. (Default: "displayValue")

```
<Attributname>.reference.displayValue = <Anzeigeformat>
```

Formatierungsanweisung zur Anzeige in der Dropdownbox, z.B. "{lastname}, {firstname} ({email})", auch mehrsprachig möglich. Die Formatierung kann nur als String-Wert nicht als HTML-Fragment angegeben werden.

```
<Attributname>.reference.searchAttributes = <Liste der Attributnamen>
```

Attribute, die vom Autocompleter durchsucht werden sollen, z.B. "city lastname firstname". (Default: "lastname firstname email")

```
<Attributname>.isChecked = <true|false>
```

Die Verifikation des Attributwerts auf den erlaubten Wertebereich (bei Referenzattributen z.B. der als Referenz angegebenen ID) vor dem Speichern kann deaktiviert werden, so dass Referenzen auf möglicherweise (noch) nicht vorhandene Einträge gesetzt werden können. (Default: "true")

Beispiel

Der folgende Ausschnitt aus der additional.attributes-Datei definiert ein Referenzattribut ref, das über die uid eine Referenz auf einen Eintrag vom Entrytype standort speichert. Die möglichen Einträge werden auf der Edit-Seite in einer Dropdownliste zur Auswahl angeboten. Es werden jeweils Stadt und Straße angezeigt.

```
referenzAttribut.type = REFERENCE
referenzAttribut.title = "Eine Referenz auf einen Standort-Eintrag"
#referenzAttribut.reference.foreignKey = uid
referenzAttribut.reference.selectorQuery = "entrytype='standort' "
referenzAttribut.reference.selector = "dropdown"
referenzAttribut.reference.displayValue = "{city}, {street}"
referenzAttribut.allowEmpty = true
```

Anzeige in der Trefferliste

Referenzattribute können, wie alle anderen Attribute auch, in die Trefferliste der Suche aufgenommen werden, siehe [Suchfunktion](#).

Soll jedoch nicht nur die UID bzw. OID des referenzierten Eintrags in der Trefferliste angezeigt werden, kann das Dereferenzierungskonstrukt follow der Personalisierungssprache verwendet werden. follow erwartet zwei Parameter: Den Namen des Referenzattributes und ein Personalisierungstemplate, das im Kontext des jeweils referenzierten Eintrags ausgeführt werden soll.

Das folgende Beispiel sorgt in der additional.attributes-Datei dafür, dass in der Spalte "Ref" jeweils Nach- und Vorname des im Attribut "referenzAttribut" referenzierten Eintrags ausgegeben werden.

```
area.ListUsers.columns = "Salut Firstname Lastname ZIP City Email Ref"
area.ListUsers.column.Ref.title = "Referenz"
area.ListUsers.column.Ref.display = "{follow|referenzAttribut|{lastname}, {firstname}}"
```

4.9.10 ATTACHMENT

Zum Speichern von Dateien gibt es den Attributtyp ATTACHMENT.

Folgende Einschränkungen sind dabei zu beachten:

- ATTACHMENT-Attribute müssen entweder Primärattribute oder Spalten in einem Tabellenattribut sein. Sie können nicht als Sekundär- bzw. Overflow-Attribute verwendet werden.
- Die Abfragesprache des Universal Messenger erlaubt für ATTACHMENT-Attribute nur die Operatoren `defined` und `undefined`.
- Die Maximallänge eines ATTACHMENT-Attributes hängt von der verwendeten Datenbank ab (siehe unten).

Datenbank	Max. Größe
Oracle	ca. 2 GB
MySQL	ca. 2 GB; der gesamte Inhalt wird vom JDBC-Treiber im Heap-Speicher der VM zwischengespeichert.
MSSQL 2000	ca. 2 GB
MSSQL ab 2005	ca. 2 GB
DB2	maximal 1 GB
Postgres	ca. 2 GB; der gesamte Inhalt wird vom JDBC-Treiber im Heap-Speicher der VM zwischengespeichert.

Intern werden die Dateien in einer separaten Tabelle gespeichert und als Wert des Attributs wird eine GUID hinterlegt. Dies gilt auch, wenn das ATTACHMENT-Attribut in einem Tabellenattribut gespeichert werden soll. Die Blob-Tabelle wird von dem Universal Messenger automatisch angelegt.

Zum Anlegen eines neuen Primärattributes `avatar` in einer bereits vorhandenen Users-Tabelle nutzen Sie das jeweilige SQL-Statement:

Datenbank	SQL
Oracle	<code>ALTER TABLE users ADD p_avatar VARCHAR2(64);</code>
Andere Datenbanken	<code>ALTER TABLE users ADD p_avatar VARCHAR(64);</code>

`<Attributname>.sqlTableName = <Tabellenname>`

Der Tabellenname bzw. die Blob-Tabelle kann optional je Attribut mit der Option `sqlTableName` geändert werden. Abhängig von der verwendeten Datenbank wird dies notwendig, wenn der Tabellenname relativ kurz sein muss und dabei sprechender sein soll als das vom Universal Messenger generierte Kürzel (z.B.: `b_avatar__1bf4e72e`).



Da Dateien, die innerhalb einer Transaktion hochgeladen werden, bereits vor dem eigentlichen Speichern des Eintrags im System gespeichert werden, wird empfohlen einen Job vom Typ "Garbage Collection" anzulegen. Mehr dazu finden Sie im Handbuch zur Benutzeroberfläche unter "Jobs".

```
<Attributname>.hashAlgorithm = <SHA-256|MD5|SHA1>
```

Zu jeder Datei kann eine Checksumme gespeichert werden, deren Hashing-Algorithmus mit der Option `hashAlgorithm` festgelegt werden kann. Dies kann beispielsweise genutzt werden, um zu prüfen, ob dieselbe Datei bereits gespeichert wurde. Ist der Wert leer (Standardkonfiguration), wird keine Checksumme gespeichert.

Beispiel einer Konfiguration

```
avatar.type = ATTACHMENT
avatar.title = "Avatar"
avatar.isPrimary = true

# optionale Angaben
avatar.hashAlgorithm = SHA-256
avatar.sqlTableName = "b_avatar_attachment_tabellenname"
avatar.sqlName = "p_avatar"
```

4.9.11 WIZARD

Bei diesem Attributtyp handelt es sich um eine Art "Pseudoattribut". Er kann verwendet werden, wenn mehrere Attribute oder sehr große Tabellen für den Nutzer vereinfacht in einem Wizard dargestellt werden sollen, um die Bearbeitung intuitiver zu machen. Um einen solchen Wizard schreiben zu können, benötigen Sie die Core Scripting Engine des Universal Messenger und das entsprechende Know-How. Attribute, die über den Wizard angezeigt und bearbeitet werden sollen, sollten in einer separaten Attributgruppe von der Anzeige und der Bearbeitung ausgeschlossen werden.

```
contact_wizard.type = WIZARD
contact_wizard.title = Bearbeitungswizard
contact_wizard.wizard_controller = de.pinuts.cmsbs.pluginintest.EntryAttributeWizard
```

4.10 Gruppierung der Attribute

Für die Gruppierung von Attributen wird eine zusätzliche Lizenz benötigt, die je nach gewählter Edition im Lieferumfang des Universal Messenger enthalten ist. Weitere Angaben können Sie der Produktbeschreibung entnehmen, bei Rückfragen wenden Sie sich bitte an unseren Support.

Im zweiten Abschnitt der Konfigurationsdatei `cmsbs-conf/additional.attributes` bzw. in der in den Konfigurationseinstellungen definierten Datei können die Attribute zur Darstellung in Attributgruppen zusammengefasst und die Sichtbarkeit definiert werden. Durch Leerzeichen getrennte Liste der (internen) Namen der Attributgruppen.

```
attribute.groups = <Liste der Attributgruppen>
```

oder:

```
attribute.groups[] = <Attributgruppe>
```

Wird die zweite Variante verwendet, ist die Position der Attributgruppen durch die Reihenfolge in der Datei bestimmt. Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten. Für jede Attributgruppe müssen die folgenden Optionen angegeben werden.

```
grp.<Attributgruppe>.title.<Sprache> = <Text>
```

Titel der Attributgruppe, der in der Benutzeroberfläche des Universal Messenger für die angegebene Sprache angezeigt wird.

```
grp.<Attributgruppe>.show = <true|false|always>
```

In der Benutzeroberfläche für Administratoren können die Attributgruppen auf- und zugeklappt werden, so dass alle Eingabefelder angezeigt werden oder nur der Titel der Attributgruppe angezeigt wird. Mit dieser Konfigurationseinstellung können Sie vorgeben, ob die Attributgruppe beim Anlegen oder Bearbeiten eines Abonnenten auf- oder zugeklappt sein soll.

Mit der Einstellung `always` können Sie angeben, dass die Attributgruppe für alle Administratoren unabhängig von den gesetzten Rechten und Rollen immer angezeigt wird. Diese Einstellung überschreibt die individuelle Konfiguration der Attributgruppen.

```
grp.<Attributgruppe>.showForNew = <true|false>
```

Mit dieser Konfigurationseinstellung können Sie für das Anlegen eines neuen Abonnenten einen anderen Zustand vorgeben, als mit der vorhergehend beschriebenen Einstellung festgelegt wurde. Ist diese Konfigurationseinstellung nicht angegeben, wird die Einstellung von `grp.<Attributgruppe>.show` übernommen.

```
grp.<Attributgruppe>.showIf = <Abfrage>
```

Mit dieser Konfigurationseinstellung kann die Darstellung der Attributgruppe in Abhängigkeit von einer Abfrage für den aktuell angezeigten Abonnenten erfolgen. Die Attributgruppe wird aufgeklappt, wenn die Abfrage für den aktuellen Abonnenten zutrifft.

Diese Einstellung überschreibt einen mit `grp.<Attributgruppe>.show` vorgegebenen Standardwert, sie kann nicht gleichzeitig mit der Einstellung `hideIf` verwendet werden. Diese Funktion ist in dem Universal Messenger nur verfügbar, wenn in Ihrer Lizenzdatei das Feature `AttributeGroups_Flexible` freigegeben ist.

```
grp.<Attributgruppe>.hideIf = <Abfrage>
```

Mit dieser Konfigurationseinstellung kann die Darstellung der Attributgruppe in Abhängigkeit von einer Abfrage für den aktuell angezeigten Abonnenten erfolgen. Die Attributgruppe wird zugeklappt, wenn die Abfrage für den aktuellen Abonnenten zutrifft.

Diese Einstellung überschreibt einen mit `grp.<Attributgruppe>.show` vorgegebenen Standardwert, sie kann nicht gleichzeitig mit der Einstellung `showIf` verwendet werden. Diese Funktion ist in dem Universal Messenger nur verfügbar, wenn in Ihrer Lizenzdatei das Feature `AttributeGroups_Flexible` freigegeben ist.

```
grp.<Attributgruppe>.ignore = true|false
```

Mit dieser Konfigurationseinstellung können Sie vorgeben, ob die Attributgruppe vollständig unsichtbar sein soll, um z.B. das Bearbeiten interner Attribute zu verhindern.

Attribute, die *ignoriert* und nicht nur *versteckt* werden, werden nicht validiert.

```
grp.<Attributgruppe>.ignoreUnlessEntryType = <Entry-Types>
```

Wird diese Konfigurationseinstellung gesetzt, so wird die betreffende Attributgruppe für alle Entry-Types *ignoriert*, die hier nicht namentlich genannt sind.

```
grp.<Attributgruppe>.members.<Position> = <Attributname>
```

oder

```
grp.<Attributgruppe>.members[] = <Attributname>
```

Name des Attributs, das an der angegebenen Position in der Attributgruppe angezeigt werden soll. Wird die zweite Variante verwendet, ist die Position der Attribute durch die Reihenfolge in der Datei bestimmt.

```
grp.<Attributgruppe>.icon = <HTML-Fragment>
```

Bei Verwendung der iconbasierten Attributgruppendarstellung gibt diese Konfigurationseinstellung ein HTML-Fragment an, welches zur Darstellung des Icons in der Eintragsansicht- und -Bearbeitungsseite verwendet werden soll. Als Icon stehen u.a. die [Google Material Icons](#) zur Verfügung. Das Symbol mit dem Einkaufswagen ("shopping cart") kann beispielsweise durch dieses HTML-Fragment dargestellt werden: `<i class="material-icons"></i>`

Siehe auch [Rechte und Rollen](#).

4.10.1 Änderung der Gruppierung von vordefinierten Attributen

Wenn nicht nur neu angelegte kundenspezifische Attribute gruppiert werden sollen, sondern auch die Gruppierung vordefinierter Attribute geändert werden soll, sollte die Standardkonfiguration des Universal Messenger per Hilfsprogramm in eine Konfigurationsdatei geschrieben werden. Aufbauend auf dieser Konfiguration können Änderungen leichter vorgenommen werden. Bitte wechseln Sie hierzu in das Verzeichnis `scripts` und führen dort folgende Kommandozeile aus:

```
./userTool.sh ../cmsbs-conf/cmsbs.properties -attrs
```

für Linux bzw. für Windows

```
.\userTool.bat ..\cmsbs-conf\cmsbs.properties -attrs >userTool.log 2>&1
```

Dieser Kommandozeilenaufruf schreibt die aktuelle, vollständige Konfiguration der Attribute und Attributgruppen in die Datei `cfg.log`. Diese Datei ist recht umfangreich, es müssen jedoch nur die Abschnitte in Ihre angepasste `additional.attributes` übernommen werden, die Sie ändern wollen.

Übernehmen Sie aus dem unteren Drittel den Abschnitt ab

```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # #   Group definitions                               # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # #
```

in Ihre `additional.attributes`. Dort sind die Gruppen und die Zuordnung der Attribute zu den Gruppen definiert. Die Definition der Attribute braucht nicht mit übernommen zu werden. Nach einem Neustart ist die neue Gruppierung aktiv, Fehler sind ggf. im Logfile des Tomcat zu finden.

4.11 Datentypen / Entry-Types

In vielen Anwendungsszenarien wird der Universal Messenger nicht nur zur Speicherung gleichartiger Einträge (z.B. Abonnenten) verwendet, sondern es sollen verschiedene Eintragstypen gespeichert werden, z.B. Kunden und interne Newsletterabonnenten. In der objektorientierten Programmierung würde man hierfür verschiedene Objektklassen anlegen, um zu den Eintragstypen verschiedene Attribute speichern zu können. Der Universal Messenger bietet mit den Entry-Types (bzw. synonym verwendet mit "Eintragstypen" und "Datentypen") eine vergleichbare Funktion, um diese Anforderungen realisieren zu können.

 Änderungen mit Release 7.45.0

Ab Release 7.45.0 wird der Universal Messenger bei einer Neuinstallation standardmäßig mit aktivierten Entry-Types ausgeliefert.

Siehe dazu auch [Release-Notes 7.45.0](#)

Dieser Abschnitt beschreibt, wie die Attributkonfiguration so gestaltet werden kann, dass für unterschiedliche Eintragstypen (*Entry-Types*) unterschiedliche Attribute und Attributgruppen zugeordnet werden können. Für jeden definierten Eintragstyp wird das *Einträge*-Menü automatisch um einen Menüeintrag zur Anzeigen aller Einträge des Typs und einen Menüeintrag zum Anlegen eines neuen Eintrags des Typs erweitert.

In dem folgenden Beispiel werden die Schritte zur Konfiguration und Verwendung der Entry-Types beschrieben:

4.11.1 Verwendung von Entry-Types aktivieren

Die Zuordnung eines Eintrags zu einem bestimmten Entry-Types erfolgt in dem Universal Messenger anhand eines Auswahlattributs, dessen Name in der `cmsbs.properties` -Datei angegeben werden muss, um die Verwendung von Entry-Types zu aktivieren:

```
cmsbs.entrytypes.attribute = "entrytype"
```

4.11.2 Auswahlattribut konfigurieren

In diesem Beispiel soll das Auswahlattribut "entrytype" heißen. Dieses Attribut und insbesondere auch der Wertebereich dieses Attributes muss in der `additional.attributes`-Datei definiert werden. Als Grundlage kann dazu die mitgelieferte Datei `cmsbs-conf/demo-entrytype.attributes` dienen:

- Beispieldatei `cmsbs-conf/demo-entrytype.attributes` (Quelltext siehe unten) nach `cmsbs-conf/entrytype.attributes` kopieren
- Attribut `entrytype` in `additional.attributes` eintragen:

```
additional.attributes[] = entrytype
```

```
include: cmsbs-conf/entrytype.attributes
```

Im Folgenden wird der Inhalt der erwähnten Beispielkonfiguration ausgegeben:

cmsbs-conf/demo-entrytype.attributes

```
entrytype.type = STRING
entrytype.title.de = "Eintragstyp"
entrytype.title.en = "Entry type"
entrytype.dbSize = 1
entrytype.isPrimary = true
entrytype.default = "c"
entrytype.values.1 = ""
entrytype.values.2 = "a"
entrytype.values.3 = "b"
entrytype.values.4 = "c"

entrytype.value. = "unbekannt"
entrytype.value.a = "EntryType A"
entrytype.value.b = "EntryType B"
entrytype.value.c.de = "EntryType C (Kunde)"
entrytype.value.c.en = "EntryType C (Customer)"
```

In der Beispielkonfiguration wird das Auswahlattribut `entrytype` als Primärattribut deklariert:

```
entrytype.isPrimary = true
```

Das bedeutet, dass die Datenbanktabelle `users` eine eigene Spalte für dieses Attribut enthalten muss. Diese Spalte kann auch nachträglich per SQL-Kommando hinzugefügt werden. Für MySQL-Datenbanken lautet der entsprechende Aufruf:

```
ALTER TABLE users ADD p_entrytype VARCHAR(1);
```

4.11.3 Definition der Entry-Types

Für jeden Entry-Type (entspricht der Wertemenge des Auswahlattributes `entrytype`) muss eine neue Konfigurationsdatei `cmsbs-conf/entryTypes/type_<NAME>.properties` angelegt werden. Für `<NAME>` ist jeweils der interne Wert des `entrytype`-Attributes einzusetzen.

Das Verzeichnis `entryTypes` kann relativ zum Verzeichnis `cmsbs-conf` mit der Option `cmsbs.entrytypes.dir` in den `cmsbs.properties` geändert werden.

```
cmsbs.entrytypes.dir = <Pfad relativ zu cmsbs-conf>
```

Die Struktur dieser `type_*.properties`-Dateien entspricht weitgehend der der `additional.attributes`-Datei. Die Attributkonfiguration für einen bestimmten Entry-Type basiert immer auf der `additional.attributes`-Datei. Entry-Type-spezifische Anpassungen können dann in der jeweiligen `type_*.properties`-Datei vorgenommen werden und überlagern die gemeinsamen Basiseinstellungen. Folgende Entry-Type-spezifische Modifikationen der Basisattributkonfiguration sind möglich:

4.11.3.1 Attributgruppen auswählen

Es können gezielt nur bestimmte Attributgruppen gewählt werden, die übrigen werden dann in der GUI nicht angezeigt. Soll eine Gruppe, die in der `additional.attributes`-Datei bereits definiert wurde, übernommen werden, so muss ihrem Namen ein `"*"` vorangestellt werden:

```
attribute.groups = "*address *contact"
```

4.11.3.2 Attributgruppen hinzufügen

Neue Attributgruppen können hinzugefügt werden:

```
attribute.groups = "*address *contact neue_gruppe"
grp.neue_gruppe.title = "Meta-Daten"
grp.neue_gruppe.members.1 = entrytype
grp.neue_gruppe.members.2 = cmsbs.creation
```

4.11.3.3 Attributgruppen umbenennen

Die Anzeigetitel von Attributgruppen kann geändert werden:

```
grp.address.title.de = "Wohnadresse"
```

4.11.3.4 Attribute zuordnen

Die Verteilung der Attribute auf Attributgruppen kann beliebig geändert werden:

```
grp.address.members.1 = street
grp.address.members.2 = zip
grp.address.members.3 = city
```

4.11.3.5 Attribute umbenennen

Der Anzeigetitel von Attributen kann geändert werden:

```
attr.street.title = "Straße und Hausnummer"
```

4.11.3.6 Defaultwert anpassen

Der Defaultwert von Attributen kann geändert werden:

```
attr.country.default = "de"
```

4.11.3.7 Wertebereich anpassen

Der Wertebereich von Auswahlattributen kann angepasst werden:

```
attr.salut.values = "m f"
```

4.11.3.8 Bearbeitung zulassen / unterbinden

Die Bearbeitung von Attributen über die GUI kann zugelassen oder unterbunden werden:

```
attr.password.fromGui = false
```

4.11.4 Weitere Hinweise

Neue Attribute können hier jedoch **nicht** definiert werden. Die Gesamtmenge aller Attribute muss also bereits in der `additional.attributes`-Datei festgelegt sein.

4.11.5 Entry-Types und Adminrollen

Entry-Types können auch zusammen mit Adminrollen verwendet werden. In diesem Fall ergibt sich die Ausprägung der verschiedenen Entry-Types aus einer Überlagerung der einzelnen Konfigurationsdateien in folgender Reihenfolge:

1. `additional.attributes`
2. `adminRoles/role_[[ROLENAME]].properties`

3. `entryTypes/type_[[NAME]].properties`

Somit kann beispielsweise für eine einzelne Adminrollen festgelegt werden, welche Entry-Types verfügbar sein sollen. Beispiel `adminRole/role_erfasser.properties`:

```
attr.entrytype.values = "a b"
```

Damit dürfen Benutzer der Adminrolle `erfasser` nur Einträge der Typen "a" und "b" sehen und bearbeiten.

4.12 Composite Unique Key

Seit 7.45.0 kann in der Attributkonfiguration festgelegt werden, dass zwei oder mehr ID-Attribute nicht mehr nur jedes für sich, sondern alle in Kombination eindeutig (*unique*) sein müssen.

Somit lässt es sich einstellen, dass beispielsweise das *email*- zusammen mit dem *entrytype*-Attribut eindeutig sein soll, es aber mehrere Einträge mit der gleichen E-Mail-Adresse geben darf, sofern sie jeweils unterschiedliche Werte für *entrytype* haben.

In `cmsbs-conf/additional.attributes` kann das wie folgt konfiguriert werden:

```
email.isUnique = false
                # email allein muss nicht mehr eindeutig sein,
isUnique = email entrytype
                # ... aber die Kombination aus email und entrytype.
```

Wie üblich sind *null*-Werte immer erlaubt, so dass im Beispiel sowohl ein leeres *email*- als auch ein leeres *entrytype*-Attribut die weitere Eindeutigkeitsprüfung für den jeweiligen Eintrag gleichsam außer Kraft setzt.

Bei einer UM-Neuinstallation wird der Composite Unique Key wie oben gezeigt festgelegt.

Seit dem Universal Messenger 7.48.0 kann diese Eindeutigkeitsprüfung gezielt für bestimmte Entry-Types deaktiviert werden. Beispielsweise für den Entry-Type `sd_ticket`:

```
isUnique = email entrytype
isUnique.ignoreForEntryTypes[] = sd_ticket
```

Damit muss die Kombination aus *email* und *entrytype* im Allgemeinen eindeutig sein, für alle Einträge des Typs `sd_ticket` gilt diese Einschränkung jedoch nicht.

4.13 Einwilligung und Widerruf

Datenfelder für Einwilligungen speichern mittels eines Zeitstempels, wann der Nutzer die Einwilligung erteilt hat. Ist der Zeitstempel nicht gesetzt, wurde die Einwilligung nicht erteilt bzw. widerrufen. Zusätzlich zum Zeitstempel kann ein Feld für die IP-Adresse, von der aus die Einwilligung aktualisiert wurde, konfiguriert werden.

Einwilligungen können in den Formularassistenten der Apps als Checkbox in ein Formular eingefügt werden. Die Anwendung wandelt die Formularwerte und Datenbankwerte entsprechend um. Das Datenfeld wird automatisch durch die Anwendung aktualisiert, wenn eine Änderung für die Einwilligung vorgenommen werden muss.

In der Segmentierung kann mit der Erweiterung `hasConsent($consentname, $form_id)` auf Einwilligungen zugegriffen werden. Die Einwilligungen werden in der Segmentierung in der Benutzeroberfläche im Abschnitt "Abonentendaten" bereitgestellt.

4.13.1 Standard-Attributkonfiguration

In der Standard-Attributkonfiguration des Universal Messenger sind mehrere Datenfelder für Einwilligungen vorgesehen, die mit Standardfunktionen des Universal Messenger verbunden sind bzw. für diese Standardfunktionen benötigt werden.

Attribut	Beschreibung	Bemerkung
<code>gtc_date</code> , <code>gtc_ip</code>	Das Feld wird allgemein genutzt, um die Zustimmung zu Geschäftsbedingungen, Nutzungsbedingungen und / oder Datenschutz zu dokumentieren.	Aus historischen Gründen wird das Feld <code>gtc</code> auch über den Bezeichner <code>permission</code> genutzt.
<code>personal_tracking_date</code> , <code>personal_tracking_ip</code>	Das Feld wird genutzt, um die Zustimmung zum personenbezogenen Tracking in der Customer Analytics App zu dokumentieren.	Aus historischen Gründen wird das Feld <code>personal_tracking</code> auch über den Bezeichner <code>permissionTracking</code> genutzt.

Für das Newsletter-Tracking werden die folgenden Attribute verwendet (siehe auch das Handbuch zur Benutzeroberfläche im Abschnitt Newsletter-Tracking):

Konfigurationsoption	Default bei Update	Default bei Neuinstallation	Beschreibung
<code>cmsbs.tracker.tracking_personal.attribute</code>	<code>behavior_date</code>	<code>consent_nl_tracking_personal</code>	Name des Consent-Attribut für personen-bezog Tracking

<code>cmsbs.tracker.tracking_anonymous.attribute</code>		<code>consent_nl_tracking_anonymous</code>	Name des Consent-Attribut für anonymes Tracking
<code>cmsbs.tracker.tracking_optout.attribute</code>	<code>anonymous</code>	<code>consent_nl_tracking_optout</code>	Name des Consent-Attribut für explizites Tracking Opt-Out

4.13.2 Eigene Einwilligungen

Darüber hinaus können in die Attributkonfiguration des Universal Messenger zusätzliche Datenfelder für Einwilligungen aufgenommen werden. Die Datenfelder beginnen mit dem Präfix "consent_". Das optionale Feld für die Speicherung einer IP-Adresse heißt gleich zum Datenfeld Einwilligung und wird mit dem Präfix „consent_“ und dem Suffix „_ip“ erweitert.

Die Aktualisierung des Datenfeldes für die IP-Adresse erfolgt durch die Anwendung zusammen mit dem Datumsfeld.

Die projektspezifischen Datenfelder für Einwilligungen werden automatisch in den Formularassistenten in den Apps aufgenommen. Die Einwilligungen werden als Checkbox angezeigt.

Auszug aus der Konfiguration `$UM_HOME/cmsbs-conf/entryTypes/type_user.properties`

```
attribute.groups[] = *consents
```

Auszug aus der Konfiguration `$UM_HOME/cmsbs-conf/conf.d/consent.attributes`

```

additional.attributes[] = consent_contact_phone
additional.attributes[] = consent_contact_phone_ip
additional.attributes[] = consent_contact_whatsapp
additional.attributes[] = consent_contact_whatsapp_ip
additional.attributes[] = consent_datasharing_ag
additional.attributes[] = consent_datasharing_ag_ip
consent_contact_phone.type = TIMESTAMP
consent_contact_phone.title = Kontaktaufnahme per Telefon
consent_contact_phone_ip.title "Kontaktaufnahme per Telefon IP"
consent_contact_whatsapp.type = TIMESTAMP
consent_contact_whatsapp.title = Kontaktaufnahme per WhatsApp
consent_contact_whatsapp_ip.title "Kontaktaufnahme per WhatsApp IP"
consent_datasharing_ag.type = TIMESTAMP
consent_datasharing_ag.title = Datenweitergabe an Unternehmen der Company AG
consent_datasharing_ag_ip.title "Datenweitergabe an Unternehmen der Company AG IP"
attribute.groups[] = consents
grp.consents.title.de = "Einwilligungen"
grp.consents.title.en = "Consents"
grp.consents.members[] = consent_contact_phone
grp.consents.members[] = consent_contact_phone_ip
grp.consents.members[] = consent_contact_whatsapp
grp.consents.members[] = consent_contact_whatsapp_ip
grp.consents.members[] = consent_datasharing_ag
grp.consents.members[] = consent_datasharing_ag_ip

```

4.13.3 Historien-Tabelle consent_history

Alle Änderungen an Datenfeldern für Einwilligungen werden dauerhaft in das Log `consent_history` aufgenommen. Das gilt auch für Änderungen an den Einwilligungen eines Eintrags in der Benutzeroberfläche des Universal Messenger. Das Log speichert somit auch den Zeitpunkt eines Widerrufs. Das Log wird tabellarisch in der Eintragsübersicht im Abschnitt "Abonnentendaten" ausgegeben.

Der Inhalt des Logs wird in das History-Log des Universal Messenger übernommen, wenn der Eintrag gelöscht wird. Die Einstellungen des History-Logs bestimmen die Vorhaltezeit dieser Angaben.

Der Eintrag ins Log kann Referenzen zu App-Instanzen aufnehmen, in denen die Einwilligung erteilt bzw. der Widerruf vorgenommen wurde.

Datenfeld	Datentyp	Beschreibung	Beispiel	Bemerkung
form_name	STRING	Name der App, in der die Einwilligung geschrieben wurde	Registrieren	

form_id	STRING	ID der App, in der die Einwilligung geschrieben wurde	eqa57c1j-efe5-4091-8c4f-5cad6a21a6ed	Kann in der Segmentierung genutzt werden <code>hasConsent("permission", "eqa57c1j-efe5-4091-8c4f-5cad6a21a6ed")</code>
form_type	STRING	Typ der App, in der die Einwilligung geschrieben wurde	de.pinuts.cmsbs.contactform	
date	TIMESTAMP	Zeitstempel der Änderung	Fr, 29. Jun. 2018, 12:23:45	
consentname	STRING	Name des Einwilligungsfeldes	permission	Kann in der Segmentierung genutzt werden <code>hasConsent("permission")</code>
consented	STRING	Ja (Einwilligung) oder Nein (Einwilligung nicht erteilt bzw. widerrufen)	Ja	
ip	STRING	IP-Adresse, von der aus die Änderung vorgenommen wurde	85.67.73.105	
firstname	STRING	Vorname des Eintrags / Nutzers	John	
lastname	STRING	Nachname des Eintrags / Nutzers	Smith	
email	STRING	E-Mail des Eintrags / Nutzers	j.smith@example.com	
text	TEXT	Bezeichner und Beschreibungen der Einwilligungen aus dem Formular in einer JSON-Struktur	<pre>{ "heading": "permission", "label": "Ja, ich akzeptiere die AGB der Company AG." }</pre>	

Auszug aus der Konfiguration `$UM_HOME/cmsbs-conf/additional.properties`

```
additional.attributes[] = consent_history
include: cmsbs-conf/cse/api/plugins/de.pinuts.consent/consent-history.attributes
...
# Eintragsübersicht Abschnitt "Abonentendaten"
grp.abo.members[] = consent_history
```

Auszug aus der Konfiguration

\$UM_HOME/cmsbs-conf/cse/api/plugins/de.pinuts.consent/consent-history.attributes

```
consent_history.type = "TABLE"
consent_history.title.de = "Einwilligung Historie"
consent_history.fromGui = "false"
consent_history.members = "form_name form_id form_type date consentname consented ip firstname
lastname email text"
consent_history.includeInHistoryLog = "true"
consent_history->date.type = "TIMESTAMP"
consent_history->text.type = "TEXT"
consent_history->form_name.title.de = "App-Name"
consent_history->form_id.title.de = "App-ID"
consent_history->form_type.title.de = "App-Type"
consent_history->date.title.de = "Datum"
consent_history->consentname.title.de = "Name"
consent_history->consented.title.de = "Eingewilligt?"
consent_history->consented.dbSize = "12"
consent_history->consented.values[] = ""
consent_history->consented.values[] = "yes"
consent_history->consented.values[] = "no"
consent_history->consented.value.yes.de = "Ja"
consent_history->consented.value.no.de = "Nein"
consent_history->ip.title.de = "IP"
consent_history->firstname.title.de = "Vorname"
consent_history->lastname.title.de = "Nachname"
consent_history->email.title.de = "E-Mail"
consent_history->text.title.de = "Text"
```

4.14 Konfigurationsfragmente unter conf.d

Im Verzeichnis `cmsbs-conf/conf.d` können spezifische Teile der Konfiguration für den Universal Messenger abgelegt werden. Diese werden wie folgt behandelt:

`*.properties`

Die Inhalte von properties-Dateien werden an das Ende der Konfigurationsdatei `cmsbs.properties` geladen.

`*.attributes`

Die Inhalte von attributes-Dateien werden an das Ende der Attribut-Konfigurationsdatei `additional.attributes` geladen.

`*.js`

JavaScript-Dateien werden vor der CSE-API am Anfang in jeden CSE-Scope geladen.

`*.xml`

Die Einstellungen des Universal Messenger, die über die Benutzeroberfläche vorgenommen werden (z.B. das Anlegen von Listen, Segmenten und Mailingvorlagen), können aktuell über einen Export-Job (Menüpunkt Extras > Neuer Job > Jobtyp: Konfiguration exportieren) exportiert und analog dazu wieder importiert werden.

XML-Dateien, die beim Starten des Universal Messenger geladen werden sollen, sollten also in diesem Format im `conf.d`-Verzeichnis abgelegt werden. Fehlermeldungen beim Laden werden in das `cse.log` geschrieben.

Einstellungen, die bereits im dem Universal Messenger vorgenommen wurden, werden nicht überschrieben.
`*.overwrite.xml`

Wie `*.xml`, allerdings werden bereits vorhandene Einstellungen (Listen, Segmente, Mailingvorlagen usw.) bei jedem Neustart überschrieben.

Alle Dateien werden jeweils in ihrer alphanumerischen Reihenfolge geladen.

Der Pfad des `conf.d`-Verzeichnis kann in der Konfigurationsdatei `cmsbs.properties` geändert werden:

```
cmsbs.directory.conf.d = <Pfad>
```

5 Konfiguration der Benutzeroberfläche

Der Universal Messenger bietet zahlreiche Möglichkeiten, um die Benutzeroberfläche für den jeweiligen Anwendungsfall und für verschiedene Nutzergruppen anzupassen. Den Benutzern können vorher definierte Adminrollen zugeordnet werden und die in der Benutzeroberfläche zur Verfügung stehenden Funktionen und die Rechte der Benutzer können über die Konfiguration der Rechte und Rollen eingeschränkt werden.

Mit der Anbindung des Universal Messenger an einen bestehenden Verzeichnisdienst wie z.B. LDAP können im Unternehmen bereits bestehende Mitarbeiterverzeichnisse für das Login zur Benutzeroberfläche des Universal Messenger genutzt werden und es kann durch zusätzliche Konfigurationseinstellungen ein Single Sign-On realisiert werden, so dass die Eingabe des Passworts entfällt, wenn der Mitarbeiter bereits an seinem Arbeitsplatz angemeldet ist.

5.1 Allgemeine Einstellungen

5.1.1 Allgemeine Einstellungen

```
cmsbs.gui.session.timeout = <Anzahl der Minuten>
```

Länge der Session der Benutzeroberfläche in Minuten, nach denen bei Inaktivität eine neue Anmeldung notwendig wird. In der Standardkonfiguration ist die Sessiondauer auf 30 Minuten eingestellt.

```
cmsbs.gui.showStackTraces = true|false
```

Gibt an, ob Stacktraces auf Fehlerseiten ausgegeben werden dürfen, per Default true.

```
cmsbs.gui.showTemplatingErrors = true|false
```

Gibt an, ob Template-Fehler bei CSE-basierten Seiten mit Stacktrace und Code-Fragmenten angezeigt werden soll. Per Default true.

```
cmsbs.jsp.title = <Text>
```

Der Text, der als Überschrift auf den Seiten des Universal Messenger erscheinen soll.

```
cmsbs.jsp.showContact = true|false
```

Gibt an, ob die Kontaktadresse zum Hersteller der Software auf der Login-Seite und der Extras-Seite in der Benutzeroberfläche angezeigt werden soll.

```
cmsbs.jsp.logo = <Pfad>
```

Gibt den Pfad auf dem Webserver zu dem Logo an, das auf den Seiten des Universal Messenger erscheinen soll. Die Grafik sollte 44 Pixel hoch sein.


```
cmsbs.jsp.logoLogin = <Pfad>
```

Gibt den Pfad auf dem Webserver zu dem Logo an, das auf der Login-Seite des Universal Messenger angezeigt werden soll.

```
cmsbs.gui.allowLogout = true|false
```

Gibt an, ob der Logout-Button angezeigt werden soll.

```
cmsbs.gui.newsarchive.minContacts = <leerzeichengetrennte Liste von Zahlen>
```

Wenn diese Variable gesetzt ist, wird das Newsletterarchiv um einen Filter ergänzt, der Newsletter mit kleineren Zielsegmenten ausblendet. Die Variable kann leerzeichengetrennt einen oder mehrere Zahlenwerte enthalten, die im Filter jeweils als Untergrenze der Zielsegmentgröße auswählbar sein sollen. Die erste Zahl wird als Vorauswahl verwendet.

Beispiel: `cmsbs.gui.newsarchive.minContacts = "10 0 5 100"`

Der Filter wird die Einträge "beliebig" (Wert "0"), "mindestens 5" (Wert "5"), "mindestens 10" (Wert :10") und "mindestens 100" (Wert "100") enthalten. "mindestens 10" wird vorausgewählt.

```
cmsbs.gui.newsarchive.pageSize = <maximale Anzahl von Newslettern pro Seite>
```

Gibt die maximale Anzahl von Newslettern pro Seite im Newsletterarchiv an. Die Standardeinstellung ist 20.

```
cmsbs.gui.ListUsers.keepSearchesTime = <Sekunden>
```

Gibt die Sekunden an, die eine Suche in einer Session gespeichert wird. 0 steht für endlos und ist die Standardeinstellung.

```
cmsbs.gui.ListUsers.keepSearches = <Anzahl der offenen Suchen>
```

Gibt die Anzahl der offenen Suchen an. Die Standardeinstellung ist 32.

```
cmsbs.gui.ListUsers.allowScrolling = true|false
```

Gibt an, ob die Trefferliste scrollbar ist (`true`) oder Seiten erstellt werden (`false`, Standardeinstellung).

```
cmsbs.gui.ListChannels.separatePublicAndPrivate = true|false
```

Gibt an, ob private und öffentliche (V)Channels in zwei einzelnen Tabellen angezeigt werden sollen. Die Standardeinstellung ist `false`.

```
cmsbs.gui.ListChannels.minGroupSize = <Anzahl>
```

Gibt an, ab wann *Listen* und *Segmente* gruppiert werden, wenn die beginnenden Wörter gleich sind. Die Standardeinstellung ist 3

```
cmsbs.gui.ListNotifications.minGroupSize = <Anzahl>
```

Gibt an, ab wann *Mailingvorlagen* gruppiert werden, wenn die beginnenden Wörter gleich sind. Die Standardeinstellung ist 3

```
cmsbs.gui.VisualQuery.charts = true|false
```

Bei sehr großen Datenmengen, Channels und VChannels sollte diese Option auf `false` gesetzt werden, um Wartezeiten zu verringern. Bei der grafischen Suche werden dann keine Grafiken mehr erzeugt.

```
cmsbs.gui.VisualQuery.autocompleter = true|false
```

Bei sehr großen Datenmengen sollte diese Option auf `false` gesetzt werden, um Wartezeiten zu verringern. Bei der grafischen Suche werden dann die meisten Autocompleter deaktiviert, um die Datenbank nicht zu überlasten.

```
cmsbs.plugininstance.usage.entries = <Anzahl>
```

Die Verwendung der Formulare von Apps – z.B. Newsletter-Anmeldeformulare, Kontaktformulare usw. – kann von dem Universal Messenger protokolliert werden, so dass im Backoffice nachvollzogen werden kann, welche App-Instanzen wann und auf welcher Seite aufgerufen bzw. abgeschickt wurden.

Diese Funktion muss durch Setzen der oben genannten Konfigurationsoption aktiviert werden. Die Zahl gibt dabei an, wie viele Aufrufe pro App-Instanz gespeichert werden sollen. Der Wert 0 deaktiviert die Protokollierung (=Default).

5.1.2 Bearbeitung von Listen ("Channels")

5.1.2.1 Custom-Attribute

Listen können um *Custom-Attribute* erweitert werden, die über die Backoffice-Oberfläche bearbeitet und per CSE- oder REST-API abgerufen werden können.

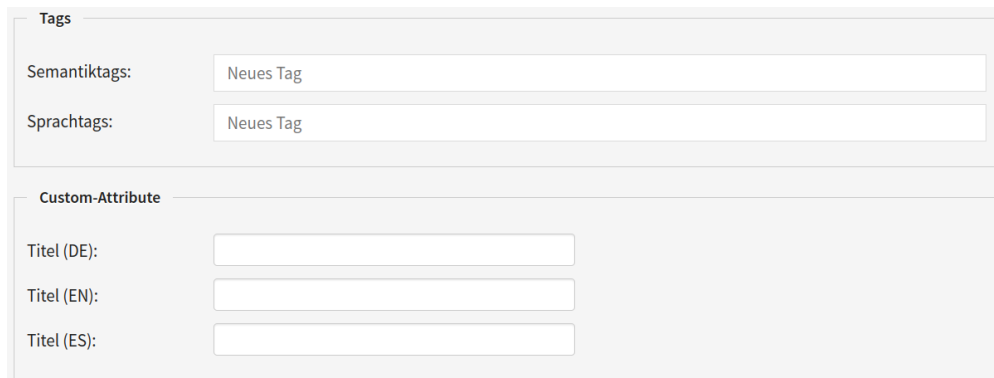
Die Namen der gewünschten Custom-Attribute werden in `cmsbs.gui.EditChannel.customAttributes` aufgelistet. Die zugehörigen ausgeschriebenen Titel dazu können jeweils in

`cmsbs.gui.EditChannel.customAttribute.[ATTR_NAME].title` angegeben werden.

Beispiel:

```
cmsbs.gui.EditChannel.customAttributes = title_de title_en title_es
cmsbs.gui.EditChannel.customAttribute.title_de.title = Titel (DE)
cmsbs.gui.EditChannel.customAttribute.title_en.title = Titel (EN)
cmsbs.gui.EditChannel.customAttribute.title_es.title = Titel (ES)
```

Bei der Bearbeitung einer Liste können die Werte Custom-Attribute modifiziert werden:



5.2 Symbole für Einträge

In dem Universal Messenger können Symbole (Icons) definiert werden, die Einträge mit bestimmten Eigenschaften kennzeichnen. Wenn die Attribute des Eintrags zur konfigurierten Abfrage passen, wird das zugeordnete Icon angezeigt. Sie können z.B. alle Abonnenten kennzeichnen, die den Pressechannel abonniert haben oder die Abonnenten, die den Double Opt-In Anmeldevorgang abgeschlossen haben.

Die Symbole werden in der Benutzeroberfläche in der Trefferliste zur Suche nach Einträgen und bei der Anzeige eines Eintrags ausgegeben.

Zusammen mit dem Universal Messenger werden einige vordefinierte Icons ausgeliefert, die mit dem Browser unter der URL `http://server:port/cmsbs/userIcons/` eingesehen werden können. Diese Icons können auf Anfrage um eigene Symbole erweitert werden, bitte senden Sie uns dazu freigestellte Grafiken im GIF-Format in einer Auflösung von 16x16-Pixel.

Die Konfiguration der Icons erfolgt über die zentrale Konfigurationsdatei

```
<UM_HOME>/cmsbs-conf/cmsbs.properties.
```

```
custom.userIcons.url = <Pfad>
```

Pfad relativ zur URL des Universal Messenger `http://server:port/cmsbs/` oder vollständige URL zu dem Verzeichnis mit den Grafikdateien der Icons, wobei beide Angaben mit einem Schrägstrich abschließen müssen. In der Standardkonfiguration ist diese Einstellung auf `userIcons/` gesetzt.

```
custom.userIcons = <Liste der Symbolnamen>
```

Durch Leerzeichen getrennte Liste der (internen) Namen der Symbole (Icons). Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten. Für jedes Symbol müssen die folgenden Optionen angegeben werden.

```
custom.userIcon.<Symbolname> = <Dateiname>
```

Name der Grafikdatei für die Anzeige des Icons.

```
custom.userIcon.<Symbolname>.title.<Sprache> = <Text>
```

Titel des Symbols, der beim Überfahren des Icons mit der Maus als Tool Tip angezeigt wird. Falls kein Titel angegeben wird, wird der Symbolname als Tool Tip ausgegeben.

```
custom.userIcon.<Symbolname>.query.<Sprache> = <Abfrage>
```

Das Symbol wird in Abhängigkeit von der hier definierten Abfrage für den aktuell angezeigten Eintrag dargestellt.

5.3 Überschrift für Einträge

Die Überschrift für Einträge im Ansicht- bzw. Bearbeitungsmodus kann über ein Template gesteuert werden. Dazu kann in der `additional.attributes`-Datei, einer Admin-Rollen-Definition oder einer Entry-Type-Definition die Einstellung `area.ShowUser.title` verändert werden, z.B.:

```
# Für alle Sprachen
area.ShowUser.title.display = "{firstname} {lastname} ({uid})"

# Nur für deutschsprachige Admins
area.ShowUser.title.display.de = "{firstname} {lastname} ({uid}) de"
```

Standardmäßig wird das folgende Template für alle Admin-Rollen und Entry-Types verwendet:

```
area.ShowUser.title.display = "{nice:salut} {firstname} {lastname}"
```

5.4 Suchfunktion

5.4.1 Suchformular

```
cmsbs.gui.ListUsers.extra = <Attributname>
```

Name eines Attributs, für das im Suchformular zusätzlich zu den vorhandenen Feldern ein Eingabefeld angezeigt wird. Es wird nach Übereinstimmung gesucht und kein Jokerzeichen voran- und nachgestellt.

5.4.2 Trefferliste

```
cmsbs.gui.ListUsers.HitsPerPage = <Anzahl>
```

Anzahl der gleichzeitig dargestellten Treffer auf der Seite zur Suche nach Einträgen, die Standardeinstellung sind 20 Treffer.

5.4.3 Definition der Spalten

Das Format der Trefferliste bei der Suche nach Einträgen in der Benutzeroberfläche kann konfiguriert und um zusätzliche Benutzerattribute erweitert werden. Die Ausgabe der Benutzerattribute und die Formatierung wird mit Personalisierungsvariablen und Formatierungsanweisungen definiert. Besondere Hinweise bei der Verwendung eines Referenzattributs finden Sie in der [Beschreibung zum Attributtyp REFERENCE](#).

In der Standardkonfiguration des Universal Messenger werden folgende vordefinierte Spalten in der Trefferliste ausgegeben:

Spaltenname	Formatierungsanweisung	Beschreibung
Icons	{UserIcons:*}	alle passenden Symbole
Fullname	{lastname}, {firstname}	voller Name
Company	{company}	Firma
Email	{email}	verlinkte E-Mail-Adresse
Mobile	{nice:mobile}	formatierte Mobilfunknummer

Die Konfiguration der Spalten der Trefferliste erfolgt in der Konfigurationsdatei

<UM_HOME>/cmsbs-conf/additional.attributes bzw. in der in den Konfigurationseinstellungen definierten Datei.

```
area.ListUsers.columns = <Liste der Spaltennamen>
```

oder

```
area.ListUsers.columns[] = <Spaltenname>
```

Durch Leerzeichen getrennte Liste der (internen) Spaltennamen. Wird die zweite Variante verwendet, ist die Position der Spalten durch die Reihenfolge in der Datei bestimmt. Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten.

Mit einem Stern vor dem Namen der Attributgruppe gekennzeichnete Spaltennamen werden aus der Standardkonfiguration übernommen, so dass sie nicht erneut konfiguriert werden müssen. Einzelne Optionen können hier jedoch überschrieben werden, so dass die Standardkonfiguration übernommen aber z.B. der Titel der Spalte geändert wird.

```
area.ListUsers.column.<Spaltenname>.title.<Sprache> = <Text>
```

Titel der Spalte, der in der Benutzeroberfläche des Universal Messenger für die angegebene Sprache angezeigt wird.

```
area.ListUsers.column.<Spaltenname>.display.<Sprache> = <Text>
```

Personalisiertes und optional durch eine Formatierungsanweisung bearbeitetes Attribut zu dem angezeigten Eintrag. Der Text muss als HTML-Format ausgegeben werden und wird direkt zur Anzeige in der Trefferliste an den Browser gesendet.

```
area.ListUsers.column.<Spaltenname>.tooltip.<Sprache> = <Text>
```

Personalisiertes und optional durch eine Formatierungsanweisung bearbeitetes Attribut zu dem angezeigten Eintrag, das beim Überfahren der Tabellenzelle mit der Maus als Tool Tip angezeigt wird. Der Text muss als HTML-Format ausgegeben werden und wird direkt zur Anzeige in der Trefferliste an den Browser gesendet.

```
area.ListUsers.column.<Spaltenname>.nowrap = true|false
```

Gibt an, ob der Inhalt der Tabellenzelle beim Überschreiten der Breite umgebrochen werden soll. Im Normalfall wird mit einem Umbruch ein besseres optisches Resultat erzeugt, Mobilfunknummern oder ähnliche Attribute sollten jedoch nicht umgebrochen werden.

5.5 Login

5.5.1 User-Authentifizierung

Der Universal Messenger kann Backoffice-User auf verschiedenen Wegen authentifizieren. Dazu wird die Einstellung `cmsbs.gui.login` verwendet, welche den Login-Modus festlegt.

5.5.1.1 Login-Modus "default" – seit Release 7.46

```
cmsbs.gui.login = default
```

Seit Release 7.46 wird bei neuen Installationen der Login-Modus `default` aktiviert. In diesem Modus stehen die folgenden Quellen zur User-Authentifizierung zur Verfügung:

1. Single Sign-On via OpenID Connect
2. Passwort-basierter Login gegen LDAP bzw. Active Directory
3. Passwort-basierter Login gegen die interne User-Verwaltung
4. Passwort-basierter Login gegen in der Konfigurationsdatei festgelegte Userdaten.

Jede dieser vier Quellen kann einzeln per Konfigurationsoption aktiviert oder deaktiviert werden. Versucht ein Nutzer, sich im Backoffice anzumelden, werden die aktivierten Login-Methoden in der hier dargestellten Reihenfolge angesprochen. Sobald der Login mit einer der Methoden gelingt, wird die Nutzersitzung entsprechend gestartet und der Login ist abgeschlossen. Schlägt auch die letzte der aktivierten Methoden fehl, ist der Login insgesamt fehlgeschlagen.

Die Konfiguration für den Login-Modus `default` erfolgt in der Datei `cmsbs-conf/conf.d/guilogin.js`, die vom Installer initial angelegt wird. Anders als in einigen der anderen Konfigurationsdateien, liegt diese Datei im JavaScript-Format vor und muss entsprechend der JavaScript-Syntax bearbeitet werden.

cmsbs-conf/conf.d/guilogin.js

```

var GUILOGIN_CONFIG = {
  ldap: {
    // 1) Login against LDAP / ADS
    enabled: false,
    debug: false,
    url: "ldap://ldap-host:389",
    bindDnPrefix: "WINDOWS_DOMAIN\\",
    adminBindDn: "WINDOWS_DOMAIN\\Administrator",
    adminPassword: "xxx",
    userSearchFilter: "sAMAccountName=${login}",
    userSearchBase: "ou=User,ou=org-unit,dc=domain,dc=local",
    userSearchTree: true,
    // loadAttributes: ['memberOf', 'sn', 'givenName', 'mail'], // Uncomment for SLAPD
    shadowUser: {
      loginAttribute: "login_name",
      entrytype: 'admin',
      umAttributes: {
        email: '${mail}',
        firstname: '${givenName}',
        lastname: '${sn}',
        street: '${streetAddress}',
        city: '${l}',
      }
    },
  },
  mapping: {
    memberOfAttribute: "memberOf",
    groupToAdminRole: {
      "cn=Admins,ou=Group,ou=org-unit,dc=domain,dc=local": "admin",
    },
    groupToTenant: {},
  },
},
internal: {
  // 2) Internal user management
  enabled: true,
},
config: {
  // 3) Accept credentials from config file: cmsbs.gui.user.*
  enabled: true,
},
oidc: {
  // Single-Sign-On via OpenID Connect
  enabled: false,
  guiBaseUrl: 'http://localhost:8080/cmsbs/',
  requiredRoles: [ 'um-user' ],
  defaultAdminRole: 'editor',
  shadowUser: {
    // loginAttribute: "login_name",
    entrytype: 'admin',
    umAttributes: {
      email: '${email}',
      firstname: '${given_name}',
      lastname: '${family_name}',
    }
  }
}

```

```

    },
    mapping: {
      roleToAdminRole: {
        "um-admin": "admin",
      },
      roleToTenant: {},
    },
  },
  guiUser: {
    displayName: "{firstname} {lastname} ({nice:admin_role})",
  },
};

```

LDAP / Active Directory

Ist diese Quelle aktiviert (`ldap: {enabled: true}`), können sich Nutzer gegenüber LDAP bzw. ADS authentifizieren. Dazu sind eine Reihe von Konfigurationsparameter anzugeben, welche den Zugang zum LDAP-System definieren; siehe Konfigurationsbeispiel oben.

Versucht ein User, sich am Backoffice anzumelden, wird im ersten Schritt ein *LDAP-Bind* mit dem angegebenen Loginnamen / Passwort an LDAP/ADS versucht. Falls vorhanden, wird der Loginname noch um das Präfix aus `ldap.bindDnPrefix` ergänzt. Bei ADS sollte dies der Name der Domain mit abschließendem Backslash sein.

War der Bind erfolgreich, erfolgt im zweiten Schritt ein LDAP-Bind mit dem konfigurierten technischen ("Admin")-Zugang aus `ldap.bindDnPrefix` bzw. `ldap.adminPassword`. Der Userdatensatz wird über den Suchausdruck aus `ldap.userSearchFilter` unter Berücksichtigung der Suchbasis `ldap.userSearchBase` gesucht.

Beim ersten Login des betreffenden Users wird im UM ein *ShadowUser*-Eintrag mit folgenden Attributvorbelegungen angelegt:

- `login_name` (bzw. UM-Login-Attribute): Loginname
- `cmsbs.isadmin`: `true`
- `lang`: *Defaultsprache*
- `email`: E-Mail-Adresse, siehe `ldap.shadowUser.umAttributes.email`
- `firstname`: Vorname, siehe `ldap.shadowUser.umAttributes.firstname`
- `lastname`: Nachname, siehe `ldap.shadowUser.umAttributes.lastname`
- ggf. weitere Attribute, die in `ldap.shadowUser.umAttributes` auf LDAP-Attribute abgebildet werden
- `admin_role`: Admin-Rolle abgeleitet aus LDAP-Gruppenzugehörigkeiten und `ldap.mapping.groupToAdminRole`

War der ShadowUser-Eintrag bereits vorhanden, werden die oben genannten UM-Attribute *nicht* überschrieben. Sprich: Künftig können und müssen alle Anpassungen an den Einstellungen des Users im UM erfolgen. LDAP dient danach nur noch zur Überprüfung der Credentials.

Anschließend wird die User-Session mit den oben genannten Attributwerten gestartet.

Soll kein ShadowUser-Datensatz im UM angelegt und verwendet werden, kann diese Funktion deaktiviert werden, indem `ldap.shadowUser` auf `undefined` gesetzt wird:

```
var GUILOGIN_CONFIG = {
  ldap: {
    // ...
    shadowUser: undefined,
    // ...
  }
}
```

Interne Nutzerverwaltung

Ist diese Quelle aktiviert (`internal: {enabled: true}`), können sich Nutzer mit passenden *Backoffice-User*-Einträgen im Backoffice einloggen. Alle Einträge, bei denen die Checkbox *Administrator?* gesetzt ist, können sich mit ihrem Loginnamen und dem Passwort anmelden.

Das Login zur Benutzeroberfläche kann über ein beliebiges Attribut und das Passwort erfolgen, das Attribut muss jedoch mit der Kennzeichnung `isUnique` als eindeutig definiert sein. Im Normalfall wird das Attribut `login_name` (vor Release 7.45: `uid`) für die Anmeldung verwendet. Mit `cmsbs.gui.loginAttr = <Attributname>` kann das Login-Attribut geändert werden.



Im Bereich [Attributtypen](#) erfahren Sie mehr über Passwörter und deren Verschlüsselung.

Nutzerdaten aus Konfigurationsdatei

Ist diese Quelle aktiviert (`config: {enabled: true}`), können sich Nutzer mit passenden fest in der Konfigurationsdatei hinterlegten Credentials einloggen. Beispiel:

cmsbs-conf/cmsbs.properties

```
# Format: "[login_name]:[password]:[display name]:[lang]:[admin_role]"
cmsbs.gui.user.1 = "admin:adminpw:Admin User:de:"
cmsbs.gui.user.2 = "ed:ed-pw:Editor:en:editor"
```

OpenID Connect

Ist diese Quelle aktiviert (`oidc: {enabled: true}`), können sich Nutzer per SSO per OpenID Connect authentifizieren. Dazu sind eine Reihe von Konfigurationsparametern anzugeben, welche die Verknüpfung mit dem verwendeten Identity-Provider definieren; siehe Konfigurationsbeispiel oben. Wird [Keycloak](#) als Identity-Provider verwendet, kann der Großteil der erforderlichen Parameter in Form einer JSON-Datei aus Keycloak exportiert und unter `cmsbs-conf/conf.d/keycloak.json` abgelegt werden.

Versucht ein User, sich am Backoffice anzumelden, erfolgt im ersten Schritt ein clientseitiger Redirect auf die Loginseite des Identity-Providers. Nach erfolgreichem Login erfolgt wiederum ein clientseitiger Redirect zurück auf die in `oidc.guiBaseUrl` angegebene URL des UM.

Dort wird noch geprüft, ob der User alle der in `oidc.requiredRoles` aufgezählten OIDC-Rollen besitzt.

Beim ersten Login des betreffenden Users wird im UM ein *ShadowUser*-Eintrag mit folgenden Attributvorbelegungen angelegt:

- `login_name` (bzw. UM-Login-Attribute): Loginname
- `cmsbs.isadmin`: `true`
- `lang`: *Defaultsprache*
- `email`: E-Mail-Adresse, siehe `oidc.shadowUser.umAttributes.email`
- `firstname`: Vorname, siehe `oidc.shadowUser.umAttributes.firstname`
- `lastname`: Nachname, siehe `oidc.shadowUser.umAttributes.lastname`
- ggf. weitere Attribute, die in `oidc.shadowUser.umAttributes` auf OIDC-Attribute abgebildet werden
- `admin_role`: Admin-Rolle abgeleitet aus den OIDC-Rollen des Users und `oidc.mapping.roleToAdminRole`

War der ShadowUser-Eintrag bereits vorhanden, werden die oben genannten UM-Attribute *nicht* überschrieben. Sprich: Künftig können und müssen alle Anpassungen an den Einstellungen des Users im UM erfolgen. OIDC dient danach nur noch zur Überprüfung der Credentials.

Anschließend wird die User-Session mit den oben genannten Attributwerten gestartet.

Soll kein ShadowUser-Datensatz im UM angelegt und verwendet werden, kann diese Funktion deaktiviert werden, indem `oidc.shadowUser` auf `undefined` gesetzt wird:

```
var GUILOGIN_CONFIG = {
  oidc: {
    // ...
    shadowUser: undefined,
    // ...
  }
}
```

5.5.1.2 Projektspezifische Authentifizierungsmethode

```
cmsbs.gui.login = cse
```

Bei diesem Login-Modus werden alle für das Login erforderlichen Schritte durch einen CSE-Callback vorgenommen, d.h. durch kundenspezifische Skripte. Dazu wird die Klasse `ApplicationCallback` überschrieben und einige Methoden implementiert (siehe CSE API-Dokumentation). Mit dieser Technik lassen sich praktisch alle Authentifizierungsmechanismen ankoppeln (SSO, komplizierte LDAPs, beliebige Datenbanken), da in der CSE diverse Bibliotheken zur Verfügung stehen (z.B. LDAP und JDBC).

5.5.1.3 Backoffice-Oberfläche ohne User-Authentifizierung

Durch Auswahl des Login-Modus `none` wird die Authentifizierung für die Backoffice-Oberfläche deaktiviert:

```
cmsbs.gui.login = none
```

Es findet keine Authentifizierung statt, ein Benutzer ist nach dem Aufruf der URL direkt angemeldet. Diese Einstellung dient nur für Entwicklungssysteme und ist nicht für den produktiven Betrieb geeignet.

```
cmsbs.gui.login.uid = <uid>
```

```
cmsbs.gui.login.role = <Name der Adminrolle>
```

```
cmsbs.gui.login.lang = de|en|fr
```

Diese drei Optionen können optional zum `cmsbs.gui.login = none` verändert werden. Die Standardwerte entsprechen `cmsbs.gui.login.uid = "admin"`, `cmsbs.gui.login.role = ""` und `cmsbs.gui.login.lang = "de"`. Ohne Authentifizierung wird man dem definierten Eintrag, der definierten Adminrolle und/oder der definierten Sprachversion zugeordnet.

5.5.1.4 Legacy-Modi

Die Login-Modi `config` und `internal` bleiben bis auf Weiteres verfügbar, sollten jedoch in neuen Installationen nicht mehr verwendet werden.

Nur Login gegen Daten aus Konfigurationsdatei

```
cmsbs.gui.login = config
```

Die Accounts der Administratoren sind fest in der Konfigurationsdatei eingetragen. Wird während der Installation der Standard-Administrator (Benutzer "admin") eingerichtet, wird diese Methode gewählt und Username/Passwort (sowie Name, Sprache und Rolle) in der Einstellung `cmsbs.gui.user.1` festgelegt.

Nur Login gegen interne Benutzerverwaltung

```
cmsbs.gui.login = internal
```

Die Accounts der Administratoren werden wie die Abonnenten auch direkt in der Datenbank des Systems gespeichert und erscheinen dort als Einträge. Alle Einträge, bei denen die Checkbox *Administrator?* gesetzt ist, können sich mit ihrem Loginnamen und dem Passwort anmelden.

Das Login zur Benutzeroberfläche kann über ein beliebiges Attribut und das Passwort erfolgen, das Attribut muss jedoch mit der Kennzeichnung `isUnique` als eindeutig definiert sein. Im Normalfall wird das Attribut `login_name` für die Anmeldung verwendet. Mit `cmsbs.gui.loginAttr = <Attributname>` kann das Login-Attribut geändert werden.



Im Bereich [Attributtypen](#) erfahren Sie mehr über Passwörter und deren Verschlüsselung.

5.5.2 Weitere Optionen

```
cmsbs.gui.login.redirectPrefix = "http://www.example.org http://example.org"
```

URL-Präfixe der verwendeten Custom-GUIs. Custom-GUIs sind kundenspezifische Anwendungen, die direkt mit dem Universal Messenger verbunden sind und zur Authentifizierung die Login-Seite des Universal Messenger nutzen.

```
cmsbs.gui.logout.redirectUri = "Login.jsp"
```

Relative URL der Seite, auf die nach erfolgtem Logout gesprungen werden soll. Bei Verwendung eines Single-Sign-On-Mechanismus ist es sinnvoll, diesen Wert auf "Logout.jsp" zu setzen. Damit wird verhindert, dass der Benutzer nach erfolgtem Logout wieder automatisch eingeloggt wird. Die Seite "Logout.jsp" enthält einen Link, der wiederum auf die Login-Seite führt.

5.5.3 Anlegen von Backoffice-Usern

Während der Installation kann ein fester Benutzer (z.B. "admin") in die Konfigurationsdatei geschrieben werden. Der Universal Messenger ist in diesem Fall direkt nach der Installation über die Backoffice-Oberfläche zu erreichen.

Weitere User können über die Backoffice-Oberfläche angelegt werden. Hierzu legen Sie im Universal Messenger einen neuen Eintrag des Eintragstyps *Backoffice-User* an, markieren die Checkbox beim Attribut *Backoffice-Login / Administrator?* (interner Name des Attributs ist `cmsbs.isadmin`) und vergeben einen Loginnamen (`login_name`) und ein Passwort (`password`).

Neue Backoffice-User können ggf. auch per Shell-Kommando angelegt werden. Wechseln Sie dazu in das Verzeichnis `scripts` und führen Sie das Skript `userTool` mit folgender Anweisung aus:

```
./userTool.sh ../cmsbs-conf/cmsbs.properties -admin
```

bzw.

```
.\userTool.bat ..\cmsbs-conf\cmsbs.properties -admin
```

Während der Ausführung werden Sie gebeten, einige Angaben zum Zugang (z.B. Benutzernamen und Passwort) zu machen. Wenn Sie die Eingaben lediglich mit ENTER bestätigen, wird der Standardzugang mit dem Benutzernamen `admin` und dem Passwort `admin` eingerichtet.

5.6 Rechte und Rollen

5.6.1 Rechte und Rollen

5.6.1.1 Einleitung

Rollen definieren die Zugriffsrechte der ihnen zugeordneten Benutzer innerhalb des Universal Messenger. Über eine Rolle können Bereiche der Benutzeroberfläche ausgeblendet bzw. angepasst werden und es kann der Zugriff auf bestimmte Eintragsgruppen oder Attribute/Attributgruppen beschränkt werden.

Für die Nutzung von Rollen wird eine zusätzliche Lizenz benötigt, die je nach gewählter Edition im Lieferumfang des Universal Messenger enthalten ist. Weitere Angaben können Sie der Produktbeschreibung entnehmen, bei Rückfragen wenden Sie sich bitte an unseren Support.

Ist in der zentralen Konfigurationsdatei `cmsbs-conf/cmsbs.properties` die Benutzerverwaltung auf `internal` umgestellt worden, so können über die grafische Benutzeroberfläche neue Einträge angelegt werden. Jeder Abonnent ist ein solcher Eintrag. Ein Eintrag bzw. Benutzer wird über das Attribut `cmsbs.isadmin` zum Login an der Benutzeroberfläche zugelassen. Die Rechte des Benutzers können über die mit dem Attribut `admin_role` angegebene Rolle beschränkt werden. Ist das Attribut `admin_role` nicht gesetzt bzw. hat es den leeren Wert, erhält der Benutzer uneingeschränkte globale Rechte. Rollen können z.B. den Zugang zu bestimmten Menüpunkten sperren oder sie können verwendet werden, um nur bestimmte Jobs oder Einträge, die einem bestimmten Channel zugewiesen sind, darzustellen.

Ein angemeldeter Benutzer hat immer genau eine Rolle. Rollen können nicht zusammengefasst werden.

5.6.1.2 Konfigurationsdateien

```
cmsbs.adminRoles.dir = <Verzeichnisname>
```

Zu jeder Rolle wird eine Konfigurationsdatei angelegt, die im Normalfall

`cmsbs-conf/adminRoles/role_{Rollennamen}.properties` heißt. Das Verzeichnis für die Konfigurationsdateien der Rollen kann über diese Konfigurationsoption geändert werden. Das Verzeichnis kann absolut oder relativ zur Konfigurationsdatei `cmsbs.properties` angegeben werden.

5.6.1.3 Einrichten einer Rolle

Für das Attribut `admin_role` muss in der Konfigurationsdatei `cmsbs-conf/additional.attributes` ein fester Wertebereich definiert werden, der auch den leeren Wert erlauben muss. Für jede Rolle wird ein neuer Wert definiert, der dem internen Namen der Rolle entspricht.

Zu jeder Rolle wird eine Konfigurationsdatei angelegt, die im Normalfall

`cmsbs-conf/adminRoles/role_{Rollennamen}.properties` heißt.



In `cmsbs-conf/adminRoles/__prototype.properties` wird immer eine Kopiervorlage für eine Rollendefinition abgelegt, die alle Optionen umfasst und sich leicht anpassen lässt.

5.6.1.4 Funktionen der Benutzeroberfläche

In der Konfigurationsdatei der Rolle werden zunächst die für den Benutzer sichtbaren Seiten in der Benutzeroberfläche des Universal Messenger definiert. Die Seiten sind zum Teil über einen eigenen Reiter im Hauptmenü erreichbar oder können nur aus einer anderen Seite aufgerufen werden, z.B. das Anzeigen eines Eintrags aus der vorangehenden Trefferliste.

Die angegebenen Seiten mit eigenem Reiter werden in der durch die Position bestimmten Reihenfolge im Hauptmenü dargestellt. Sollen keine Reiter sichtbar sein, muss dies mit `gui.areas = ""` angegeben werden.

```
gui.areas.<Position> = <Seite>
```

oder

```
gui.areas[] = <Seite>
```

Zu einigen Seiten kann das Verhalten untergeordneter Funktionen über diese Konfigurationsoption definiert werden.

```
area.<Seite>.<Funktion> = true|false
```

Die möglichen Seiten und deren Funktionen sind in der folgenden Tabelle aufgeführt:

Seite	Funktionen	Beschreibung
ListChannels		Channel-Liste anzeigen und Einträge nach Channels suchen

	Statistics	Statistik zu den Channels und allen Einträgen
EditChannels		Channels anlegen, bearbeiten und löschen
	Create / Delete	Channels anlegen / löschen
	EditTags	Semantik- bzw. Sprachtags festlegen
ListVChannels		VChannel-Liste anzeigen
	Statistics	Statistik zu den virtuellen Channels und allen Einträgen
EditVChannels		Virtuelle Channels anlegen, bearbeiten und löschen
	Create / Delete	Virtuelle Channels anlegen / löschen (Anlegen erfordert zusätzlich die Seite <code>VisualQuery</code>)
	EditTags	Semantik- bzw. Sprachtags festlegen
ListUsers		Einträge suchen
	Query	Suchformular mit freiem Suchausdruck
	Import	Einträge importieren
	Export	Einträge exportieren
	columns	Tabellenspalten der Ergebnisse
VisualQuery		Grafische Suchabfrage
ShowUser		Einträge anzeigen
	Channels	Abonnierte Channels auf der Detail-Seite anzeigen
	Channels.EntryType.\$entryType	Abonnierte Channels eines bestimmten EntryTypes auf der Detail-Seite anzeigen
	VChannels	Abonnierte virtuelle Channels auf der Detail-Seite anzeigen
	VChannels.EntryType.\$entryType	Abonnierte virtuelle Channels eines bestimmten EntryTypes auf der Detail-Seite anzeigen
	Tracking	Tracking auf der Detail-Seite anzeigen
	Tracking.EntryType.\$entryType	Tracking eines bestimmten EntryTypes auf der Detail-Seite anzeigen
	icons	Projektspezifische Icons für den aktuellen Eintrag anzeigen. Siehe auch Symbole für Einträge
	Style	"IconBar": Iconverzierte Attributgruppendarstellung "": Textuelle Attributgruppendarstellung (Standard)
EditUser		Einträge bearbeiten und löschen
	Channels	Einträge bearbeiten und Channels setzen
	Channels.EntryType.\$entryType	Einträge eines bestimmten EntryTypes bearbeiten und Channels setzen

	Notify.Standard	Einträge bearbeiten und vordefinierte Benachrichtigungen versenden
	Notify.Custom	Einträge bearbeiten und benutzerdefinierte Benachrichtigungen versenden
	Notify.EntryType.\$entryType	Einträge eines bestimmten EntryTypes bearbeiten und Benachrichtigungen versenden
	Delete	Einträge löschen
	EntryType.\$entryType	Einträge eines bestimmten EntryTypes bearbeiten
	StealLock	Bearbeitungssperre eines anderen Benutzers aufheben und Bearbeitung übernehmen
CreateUser		Einträge anlegen
	\$entryType	Einträge eines bestimmten EntryTypes anlegen
SendNewsletter		Newsletter versenden
	EditTags	Newsletter versenden und Tags setzen
NewsletterQueue		Newsletter Versandwarteschlange aufrufen
	reschedule	Versandzeitpunkt kann in der Warteschlange geändert werden
	control	Versandfortschritt bearbeiten
NewsletterArchive		Newsletter-Archiv aufrufen
	TrackingFilter	Die Empfängerliste am archivierten Newsletter kann bzgl. Tracking-Informationen gefiltert werden
	TrackingFilter.toChannel	Die gefilterte Liste kann in Channels eingefügt bzw. daraus entfernt werden
	Delete	Newsletter löschen
	Edit	Newsletter nachträglich bearbeiten (z.B. die Schlagwörter / Tags)
	tags	Nur Newsletter mit bestimmten Tags anzeigen
NewsletterGroups		Newsletter-Gruppen aufrufen
Statistics		Statistiken aufrufen
	Total	Statistik mit Übersicht zu den Einträgen anzeigen
	Channels	Statistik mit Channels anzeigen
	VChannels	Statistik mit virtuellen Channels anzeigen
	CustomStats	Statistik mit benutzerdefinierten Statistiken anzeigen
Dashboard		Dashboard anzeigen
PluginInstances		Apps anzeigen
	Create / Edit / Delete	App-Instanzen anlegen / bearbeiten / löschen

	EditTags	Semantik- bzw. Sprachtags festlegen
Notifications		Benachrichtigungen im Menüpunkt Extras ansehen, versenden, bearbeiten, anlegen und löschen
	Create / Delete	Benachrichtigungen anlegen / löschen
	EditTags	Semantik- bzw. Sprachtags festlegen
Jobs		Jobs im Menüpunkt Extras ansehen, aufrufen, bearbeiten, anlegen und löschen
	Create / Delete	Jobs anlegen / löschen
	Edit	Existierenden Job bearbeiten
	EditTags	Jobs anlegen und Tags setzen
	EditCallback	Jobs anlegen und Callbacks setzen
ListModules		Module aufrufen
	\$module	Ein bestimmtes Modul nutzen können
SystemDashboard		System Dashboard aufrufen
MailRelays		Übersicht über alle Mail-Relays aufrufen
ApiToken		Neues API-Token generieren (<i>Extras / API-Token</i>)
LogFile		Log-Datei anzeigen
Help		Online-Hilfe und Dokumentation aufrufen

Seiten, die als `AddOn` mitgeliefert werden:

Seite	Beschreibung
<code>AddOn.de.pinuts.cmsbs.mailbacklog</code>	Mail-Back-Log im Menüpunkt Extras anzeigen



Bitte beachten Sie, dass die mit `ListChannels.Statistics` oder `ListVChannels.Statistics` aktivierte Statistik alle Einträge umfasst, auch wenn der Benutzer durch andere Beschränkungen keinen Zugriff auf diese Einträge hat. Für den Benutzer wird also ersichtlich, wie viele Einträge er nicht sehen kann.

Da der Default-Wert für Funktionen `true` ist, werden in der Konfiguration die Funktionen, die die angemeldete Rolle nicht nutzen soll, mit `false` abgeschaltet.

Beispiel

```
# Der Rolle verbieten, VChannels anzulegen
area.EditVChannels.Create = false
# Der Rolle verbieten, VChannels zu löschen
area.EditVChannels.Delete = false
# Der Rolle verbieten, Semantik- bzw. Sprachtags selbst festzulegen
area.EditVChannels.EditTags = false

# Der Rolle verbieten das MailBackLog zu sehen
gui.areas[] = AddOn.de.pinuts.cmsbs.mailbacklog = false
```

```
gui.area.default = <Menüpunkt>
```

Menüpunkt, der nach dem Login direkt angesprungen wird. Dieser Menüpunkt muss in der Liste der für den Benutzer sichtbaren Menüpunkte enthalten sein. Diese Option kann sowohl in der globalen `additional.attributes`-Datei als auch in den einzelnen Adminrollen überschrieben werden.

Gültige Werte für *Menüpunkt* sind hier die Seitennamen aus der Tabelle oben und folgende Schreibweise für die Verwendung eines Wizards als Startseite:

```
gui.area.default = /Custom/com.example.sample.MyWizard
```

5.6.1.5 Zugriff auf Apps

Die Berechtigungen zum Anlegen von App-Instanzen in der App-Übersicht werden ebenfalls über die Konfigurationsoption `gui.areas[]` vorgenommen. Dies geschieht über das Präfix `AddOn` zusammen mit dem Paketnamen der Apps selbst, z.B:

```
# Adminrolle darf App-Liste sehen:
gui.areas[] = PluginInstances

# Adminrolle darf grundsätzlich App-Instanzen anlegen:
area.PluginInstances.Create = true

# Adminrolle darf Instanzen der folgenden Apps anlegen:
gui.areas[] = AddOn.de.pinuts.cmsbs.newsletter
gui.areas[] = AddOn.de.pinuts.cmsbs.contactform
gui.areas[] = AddOn.de.pinuts.servicedesk
```

Je nach App können optional Features freigeschaltet bzw. unterbunden werden. Welche Features dies im Einzelnen genau sind, kann der Einstellung "guiFeatures" aus der Descriptor-Datei `plugin.desc.json`, die im Verzeichnis der Apps liegt, entnommen werden.

```
# Content-Tags der Customer Analytics App bearbeiten
area.AddOn.de.pinuts.cmsbs.analytics.ContentTags = true
```

Welche App-Instanzen sichtbar sind und bearbeitet werden können, hängt wie bei Segmenten und Listen von der Kombination der Sem-Tags von App-Instanz und Backoffice-User ab.

5.6.1.6 Zugriff auf Einträge

In der Konfigurationsdatei können die für den Benutzer sichtbaren Einträge über eine Abfrage eingeschränkt werden. Von dieser Abfrage sind die Menüpunkte zur Suche nach Einträgen und zur Bearbeitung von Einträgen betroffen.

Bitte beachten Sie den Zusammenhang zwischen der Zugriffsbeschränkung auf einzelne Einträge, die durch eine Abfrage auf Attribute definiert wird, und die Zugriffsbeschränkung auf die in dieser Abfrage enthaltenen Attribute. Unter Umständen kann ein Benutzer neue Einträge anlegen, für die nach dem Speichern keine Zugriffsberechtigung mehr besteht.

```
user.query = <Abfrage>
```

Es sind nur die Einträge zur Anzeige und Bearbeitung sichtbar, für die die angegebene Abfrage zutrifft.

5.6.1.7 Zugriff auf Attribute und Attributgruppen

In der Konfigurationsdatei können die für den Benutzer sichtbaren Attribute und Attributgruppen definiert werden. Die Attributgruppen können unabhängig von den global definierten Attributgruppen zusammengestellt werden, so dass eine für die jeweilige Rolle günstige Anordnung und Zusammenfassung der Attribute möglich ist. Es ist anders als in der globalen Konfiguration nicht notwendig, alle Attribute zu erfassen.

In dem Formular zur Suche nach Einträgen werden nur die für die Rolle konfigurierten Attribute zur Auswahl angezeigt. Bei der freien Eingabe eines Suchausdrucks können jedoch derzeit auch die nicht konfigurierten Attribute verwendet werden. Über speziell zusammengestellte Suchausdrücke könnte ein Benutzer indirekt Kenntnisse über eigentlich nicht konfigurierte Attribute erlangen, dies ist in sicherheitskritischen Anwendungsfällen bitte zu beachten!

Attribute

Die Konfiguration der Attribute erfolgt mit den Optionen, wie sie auch für die globale Konfiguration verwendet werden. Die Optionen werden allerdings mit einem `attr.` eingeleitet und es können nur die hier aufgeführten Optionen verwendet werden, da das grundsätzliche Verhalten der Attribute nicht für eine Rolle geändert werden kann.

```

attr.<Attributname>.title.<Sprache> = <Text>
attr.<Attributname>.default = <Text>
attr.<Attributname>.values.<Position> = <interner Wert>
attr.<Attributname>.value.<interner Wert>.<Sprache>.title = <Text>
attr.<Attributname>.fromGui = true | false

grp.<Attributgruppenname>.hideInAdminGui = true | false

```

Durch die Angabe eines von der globalen Konfiguration abweichenden Standardwerts `default` kann beim Anlegen eines neuen Eintrags sichergestellt werden, dass dieser Eintrag auch von der jeweiligen Rolle gesehen werden kann.

Die mit der Option `values` angegebenen Werte können die in der globalen Konfiguration erlaubten Werte weiter einschränken, es dürfen jedoch keine neuen Werte aufgenommen werden.

Attributgruppen

Die Konfiguration der Attributgruppen erfolgt mit den Optionen, wie sie auch für die globale Konfiguration verwendet werden.

```
attribute.groups = <Liste der Attributgruppen>
```

oder

```
attribute.groups[] = <Attributgruppe>
```

Durch Leerzeichen getrennte Liste der (internen) Namen der Attributgruppen. Wird die zweite Variante verwendet, ist die Position der Attributgruppe durch die Reihenfolge in der Datei bestimmt. Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten. Sollen keine Attribute angezeigt werden, muss eine leere Liste angegeben werden.

Mit einem Stern vor dem Namen der Attributgruppe gekennzeichnete Gruppen werden aus der globalen Konfiguration übernommen, so dass sie nicht erneut konfiguriert werden müssen.

Für die mit einem Stern gekennzeichneten und aus der globalen Konfiguration übernommenen Attributgruppen können einzelne Optionen (`title`, `show`, `showIf`, `hide`, `hideIf` und `ignore`) überschrieben oder die in der Gruppe enthaltenen Attribute geändert werden, so dass die Basiskonfiguration übernommen wird, aber z.B. der Titel der Attributgruppe geändert wird.

5.6.1.8 Zugriff auf bestimmte Listen, Segmente, Jobs und Mailingvorlagen einschränken

Wird der Universal Messenger von unterschiedlichen Benutzergruppen oder Fachabteilungen (Mandanten) verwendet, so ist es meist nicht ausreichend, den Zugriff auf einzelne Bereiche mittels der Rollen zu beschränken. Mandanten hätten damit weiterhin die Möglichkeit, alle Listen, Segmente, Jobs oder Mailingvorlagen zu sehen. Stattdessen ist gewünscht, nur einige der Listen, Segmente, Jobs usw. darzustellen, z.B. abhängig vom Standort des Mandanten. Hierzu können Semantiktags verwendet werden. Semantiktags können innerhalb der Benutzeroberfläche direkt eingegeben werden:

Im ersten Beispiel soll ein Benutzer nur die Elemente (Listen, Segmente, Jobs usw.) sehen dürfen, bei denen das semantische Tag "anzeigen" verwendet wurde. Elemente, bei denen keine semantischen Tags gesetzt wurden, werden nicht angezeigt.

Die Rollenkonfiguration wird ergänzt um `default.semTags = "anzeigen"`

Im zweiten Beispiel sollen dagegen alle Elemente angezeigt werden, die kein semantisches Tag gesetzt haben. Die zu versteckenden Elemente sind mit dem semantischen Tag "nicht_anzeigen" versehen worden.

Die Rollenkonfiguration wird ergänzt um `default.semTags = "<>"`

Damit Benutzer nicht Tags verändern können, kann die Bearbeitung dieser global deaktiviert werden.

Die Rollenkonfiguration wird ergänzt um `area.*.EditTags = false`

5.6.1.9 Symbole für Einträge

In der Konfigurationsdatei der Rolle können die Symbole individuell konfiguriert werden, die in der Benutzeroberfläche in der Trefferliste zur Suche nach Einträgen und bei der Anzeige eines Eintrags ausgegeben werden sollen. Zur Definition können die in der globalen Konfiguration erlaubten Einstellungen verwendet werden.

5.6.1.10 Trefferliste der Suchfunktion

In der Konfigurationsdatei der Rolle kann das Format der Trefferliste für die Benutzer individuell konfiguriert werden. Zur Definition können die in der globalen Konfiguration erlaubten Einstellungen verwendet werden.

5.6.1.11 Aktivieren der Beispielrollen

der Universal Messenger wird mit drei Beispiel-Rollen ausgeliefert, die sich im Verzeichnis `cmsbs-conf/adminRoles` befinden.

- `role_address.properties` : Eingeschränkt auf die Pflege von Einträgen und Jobs
- `role_controller.properties` : Eingeschränkt auf die Statistiken und Pflege von Channeln
- `role_redakteur.properties` : Eingeschränkt auf die Pflege von Einträgen mit E-Mail-Adressen

Die Datei `cmsbs-conf/demo-additional.attributes` enthält (auskommentierte)

Konfigurationsanweisungen, welche in die Datei `additional.attributes` übernommen werden müssen:

```
#admin_role.values.0 = ""
#admin_role.values.1 = address
#admin_role.values.2 = redakteur
#admin_role.values.3 = controller
```

Die Werte (rechts der Gleichheitszeichen) müssen den Dateinamen im Verzeichnis `cmsbs-conf/adminRoles` entsprechen (z.B. `role_redakteur.properties`).

Nach einem Neustart des Tomcat wird die Konfiguration eingelesen und die Rollen können zugewiesen werden.

5.6.2 Verwendung von Absender- und Reply-To-Adressen einschränken

Wie viele Absender- bzw. Reply-To-Adressen verwendet werden dürfen, ist in der Regel durch die Lizenz limitiert. In den beiden Dateien `UM/cmsbs-conf/cmsbs.sender.csv` bzw. `cmsbs.replyto.csv` steht, welche Adressen zurzeit erlaubt sind.

Die Verwendung dieser Adressen kann optional basierend auf der jeweiligen Admin-Rolle eines Benutzers auf eine Untermenge der generell erlaubten Adressen beschränkt werden. Dabei werden die Namen der Admin-Rollen, die eine bestimmte Adresse verwenden dürfen, in die dritte Spalte der CSV-Datei geschrieben:

```
"Vertrieb";vertrieb@example.org;|controller|
"Newsletter";newsletter@example.org;|controller|redakteur|
;info@example.org
```

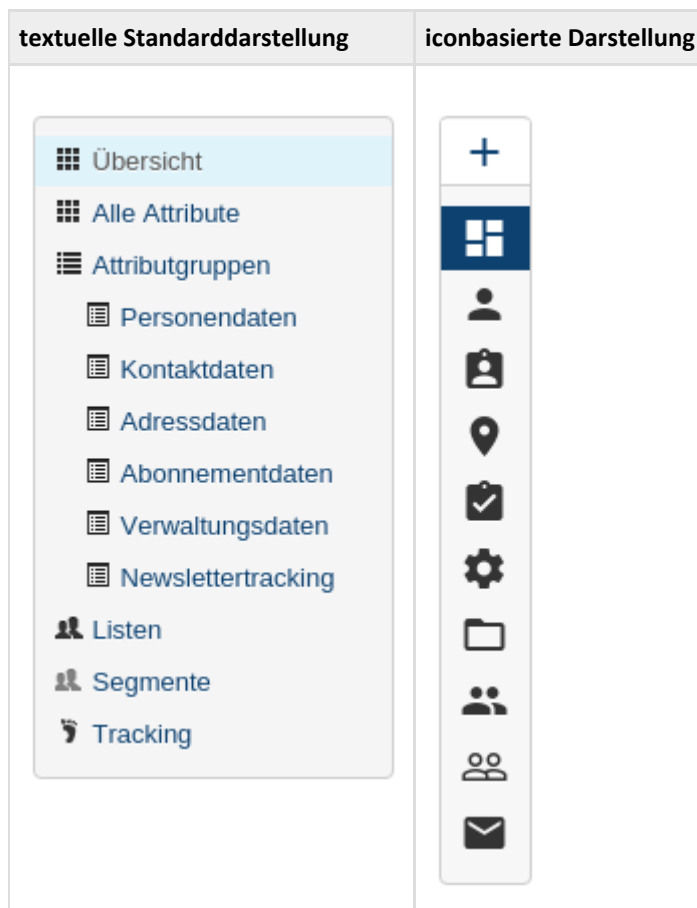
Es gelten folgende Formatvorgaben:

- Soweit nicht anders konfiguriert, ist das Encoding immer latin1.
- Die einzelnen Spalten werden durch Semikolon getrennt,
- die Spaltenwerte können optional in Anführungszeichen eingeschlossen werden,
- die Liste der Admin-Rollen in der dritten Spalte muss im Multi-Select-Format (siehe unten) angegeben werden.
- Bleibt die dritte Spalte leer oder fehlt sie ganz, kann die jeweilige Adresse von allen Admin-Rollen verwendet werden.

(Multi-Select-Format: die einzelnen Werte werden durch Pipe-Zeichen "|" getrennt. Außerdem steht vorn und hinten auch immer ein Pipe-Zeichen.)

5.6.3 Darstellung der Attributgruppenliste in der Eintragsansicht- und -Bearbeitungsseite

Die Darstellung der Attributgruppenliste in der Eintragsansichts- und Eintragsbearbeitungsseite kann textuell (Standard) oder iconbasiert erfolgen:



Um von der Standarddarstellung auf die iconbasierte Darstellung umzustellen, muss die Zeile

```
area.ShowUser.Style = IconBar
```

in die

- `additional.attributes-Datei` (Umstellung global für alle Einträge)
- oder in die jeweilige `entryTypes/type_*.properties-Datei` (für einen bestimmten Entry-Type)
- oder in die jeweilige `adminRoles/role_*.properties-Datei` (für eine bestimmte Adminrolle)

eingetragen werden.

Zur Festlegung der zu verwendenden Icons siehe [Gruppierung der Attribute](#) / `grp.<Attributgruppe>.icon`.

5.7 Mandantenfähigkeit

Die Benutzeroberfläche des Universal Messenger kann ab der Customer Interaction Edition so konfiguriert werden, dass sie für mehrere getrennte Mandanten verwendet werden kann.

Das Konzept der Mandantenfähigkeit basiert auf der Einrichtung eines speziellen "tenant"-Attributes, das den Nutzern des Universal Messenger zugewiesen werden kann. Über dieses Attribut können Bereiche der Benutzeroberfläche ausgeblendet bzw. angepasst werden und es kann der Zugriff auf bestimmte Einträge oder Attribute/Attributgruppen beschränkt werden. Auch nahezu alle anderen Objekte in dem Universal Messenger können über die sogenannten Semantiktags einem Mandanten zugeordnet werden.

Ein häufiger Anwendungsfall der Mandantenfähigkeit ist die Schaffung getrennter *Datentöpfe* für z.B. Tochtergesellschaften. Ohne Mandanten wird die E-Mail Adresse der Einträge häufig als eindeutiges Kennzeichen definiert, so dass es im System nur einen Eintrag pro E-Mail Adresse geben darf. In der Regel ist diese Eindeutigkeit notwendig, da die E-Mail Adresse zur Identifikation der Einträge verwendet wird. Mit dem Einsatz der Mandantenfähigkeit wird die E-Mail Adresse häufig als eindeutiges Kennzeichen pro Mandant definiert, d.h. die Identifikation eines Eintrags erfolgt durch die Kombination aus E-Mail Adresse und tenant-Attribut. In dieser Kombination kann es dann pro Mandant einen Eintrag pro E-Mail Adresse geben.

In der Praxis wird dies verwendet, um z.B. Tochtergesellschaften einen eigenen Datenbestand zu geben. Innerhalb einer Tochtergesellschaft (also innerhalb eines Mandanten) bleibt die E-Mail Adresse eindeutig, systemweit gibt es aber mehrere Einträge pro E-Mail Adresse.

5.7.1 Zusammenhang zwischen Rechte&Rollen und Mandantenfähigkeit

Die Einschränkung per Rechten&Rollen auf der einen Seite und die Mandantenfähigkeit auf der anderen Seite sind zwei Konzepte, die ineinander greifen.

Mit den Rechten&Rollen kann man erreichen, dass Nutzer in dem Universal Messenger nur freigeschaltete Menüpunkte aufrufen dürfen und auch nur die Einträge sehen, die ihnen zugeordnet sind, z.B. alle Einträge einer speziellen Liste. Die Rechte&Rollen sind gut geeignet, wenn z.B. Ländergesellschaften nur die Einträge sehen sollen, die auf ihrer Länderliste sind. Mit den Rechten&Rollen kann also die Sicht auf die Daten in dem Universal Messenger eingeschränkt werden. Man kann die Sicht auch auf bestimmte Attribute einschränken, so dass ein Administrator nur die Basisdaten der Einträge sehen darf, während ein anderer Administrator auch bspw. die Bestellhistorie sehen darf. Beide Administratoren sehen von dem selben Eintrag also unterschiedliche Bereiche.

Wenn ein Kunde in der Länderliste von Italien und gleichzeitig in der Länderliste von Frankreich ist, würden beide Landesgesellschaften den selben Kunden sehen.

Mit der Mandantenfähigkeit können hingegen mehrere Datentöpfe logisch voneinander getrennt werden. In diesem Fall hätten die Landesgesellschaften aus Italien und aus Frankreich jeweils ihren eigenen Datentopf und würden nicht mehr auf den selben Kunden sehen, sondern der Kunde wäre dann ggf. wirklich zweimal im System, pro Landesgesellschaft also jeweils ein getrennter Kundendatensatz.

Beim Einsatz dieser Konzepte muss abgewogen werden, ob es nur um Zugriffsrechte geht (Rechte&Rollen) oder ob die Kundendaten in separaten Datensätzen gehalten werden sollen (Mandantenfähigkeit), wobei innerhalb eines Mandanten die Zugriffsrechte wiederum durch Rechte&Rollen definiert werden können.

Zur Entscheidung kann die Sicht des Kunden helfen: Wenn ein Kunde den Newsletter abbestellen und seinen Datensatz löschen will, geht er wahrscheinlich davon aus, dass er für beide Länder gelöscht wird, wenn man zwischen den Ländern auf der Website per Sprachumschaltung schnell wechseln kann. Für den Kunden ist es trotz der verschiedenen Sprachen ein Unternehmen. Anders mag es aussehen, wenn sich der Kunde für einen Newsletter bei einer Tochtergesellschaft angemeldet hat, die vielleicht auch einen anderen Namen trägt. Dann sind es für den Kunden zwei Unternehmen und er würde sich wundern, wenn ihn die Newsletterabmeldung bei einer Tochtergesellschaft auch gleichzeitig noch von dem anderen Unternehmen abmelden würde.

5.7.2 Einrichtung

Bei einer Standardinstallation sind zur Einrichtung die in den folgenden Abschnitten beschriebenen Schritte durchzuführen.

5.7.2.1 Attribut "tenant" einrichten

In der `additional.attributes`-Datei wird ganz unten das neue Attribut angelegt:

UM/cmsbs-conf/additional.attributes

```
additional.attributes[]      = tenant
tenant.title.de             = Mandant
tenant.title.en             = Tenant
tenant.dbSize               = 64
tenant.isPrimary            = true

tenant.values[]             = m1
tenant.value.m1             = Mandant 1
tenant.values[]             = m2
tenant.value.m2             = Mandant 2
tenant.values[]             = " "
tenant.value.               = "kein Mandant"

grp.std_personal.members[] = tenant
```

Für das neue Primärattribut muss die Datenbanktabelle "users" um die entsprechende Spalte erweitert werden (siehe dazu u.a [Kundenspezifische Attribute](#)). Für MySQL-Datenbanken lautet der entsprechende Aufruf:

```
ALTER TABLE users ADD p_tenant VARCHAR(64);
```

Der Wertebereich (hier im Beispiel "m1" und "m2") muss dabei projektspezifisch angepasst werden. Für jeden Mandanten muss es hier einen entsprechenden Wert (=Mandantenkürzel) geben.

5.7.2.2 E-Mail-Adresse nicht mehr eindeutig

In den meisten Fällen muss die globale Eindeutigkeit des "email"-Attributes (und ggf. auch des "mobile"-Attributes) aufgehoben werden, damit ein und dieselbe E-Mail-Adresse in mehreren Mandanten verwendet werden kann.

Dazu wird folgende Zeile an die `additional.attributes`-Datei angefügt:

```
email.isUnique          = false
mobile.isUnique         = false
```

5.7.2.3 Admin-Rollen

Nun muss mindestens die folgende Admin-Rolle definiert werden: "tenant_user"

UM/cmsbs-conf/additional.attributes

```
admin_role.values[]      = ""
admin_role.value.        = Super-Admin
admin_role.values[]      = tenant_user
admin_role.value.tenant_user = Mandanten-Benutzer
```

UM/cmsbs-conf/adminRoles/role_tenant_user.properties

```
area.*.EditTags          = false
```

Diese Einstellungen verhindern, dass ein Mandanten-Benutzer selbst die Semantiktags bearbeiten und damit die Rechteverwaltung aushebeln kann.

Benutzer der Admin-Rolle "tenant_user" können damit nur Einträge ihres Mandanten sehen, die selbst nicht *Administrator* sind.

Die Benutzer der Admin-Rolle "tenant_user" mit gesetztem *Administrator?*-Flag (sprich: die Benutzer, die auf die GUI zugreifen können) können nur durch einen Benutzer mit der Admin-Rolle "" (Super-User) verwaltet werden.

5.7.2.4 Login-Modus "default"

Der Login-Modus muss auf den in Release 7.46 eingeführten Wert `default` gesetzt und die Konfiguration in `cmsbs-conf/conf.d/guilogin.js` entsprechend angepasst werden. Siehe dazu [Login](#).

Nun kann die Mandantenfähigkeit in `cmsbs-conf/conf.d/guilogin.js` aktiviert und konfiguriert werden:

cmsbs-conf/conf.d/guilogin.js

```
var GUILOGIN_CONFIG = {
  // ...
  multiTenancy: {
    enabled: true,

    tenantAttribute: 'tenant',
    permissions: {
      setSemTagsFor: {
        newsletters: true,
        channels: true,
        jobs: true,
        notifications: true,
        vchannels: true,
        customStats: true,
        pluginInstances: true,
        newsletterGroups: true,
      },
      modifyUserQuery: true,
      restrictSenderAddresses: false,
      restrictReplyToAddresses: false,
    }
  }
};
```

Über die Optionen unterhalb von `multiTenancy.permissions` kann detailliert eingestellt werden, welche Rechtebeschränkungen beim Login eines Backoffice-Users entsprechend seines `tenant`-Attributes eingestellt werden sollen.

Danach muss der Universal Messenger neugestartet werden.

5.7.2.5 Listen, Segmente, Mailingvorlagen, Jobs und Apps anpassen

Über die Benutzeroberfläche sollten nun alle Listen, Segmente, Mailingvorlagen, Jobs und Apps durchgegangen werden. Für jedes dieser Objekte kann nun über die Semantiktags festgelegt werden, für welche Mandanten es künftig sichtbar sein soll. Dabei muss das Mandantenkürzel jedes Mandanten, der das Objekt später sehen können soll, als jeweils ein Semantiktag angegeben werden.

Objekte ganz ohne Semantiktag sind nur noch für den "Super"-Mandanten (tenant="") sichtbar.

Beispiel einer Liste, die für die Mandanten "m1" und "m2" freigegeben werden soll:

Liste bearbeiten: Monatlicher Newsletter

Interner Name:

Titel:

Beschreibung:

Öffentlich? ☒ Ja ☐ Nein

Semantiktags:

Sprachtags:

5.7.2.6 Newsletterversand

Newsletter werden ähnlich wie Listen, Segmente usw. per *Tag* einem oder mehreren Mandanten zugeordnet. Bereits versendete Newsletter, die keine Mandantenkürzel in ihren jeweiligen Tags haben, sind künftig für keinen Mandanten sichtbar.

Newsletter, die nach der Umstellung durch einen Mandanten über die Oberfläche versendet werden, bekommen automatisch ein entsprechendes Tag, so dass sie für denjenigen Mandanten sichtbar bleiben.

Wenn Newsletter auch per Eventfile versendet werden, müssen dort ggf. entsprechende Tags mit Mandantenkürzeln ergänzt werden, z.B.:

event.xml

```
<event>
  <destination>...</destination>
  <data>...</data>
  <tag>m2</tag>
</event>
```

5.7.2.7 Newsletterformulare / Newsletter App

Falls das Newsletter App zur Erstellung von Newsletter-Anmeldeformularen verwendet wird, müssen hier alle Instanzen durchgegangen werden, um jeweils unter *Einstellungen / Mandantenkennung für neue Einträge* den passenden Mandanten auszuwählen. Z.B.:

Add-On-Konfiguration

Add-On: [Vorschau anzeigen](#) Sprache wechseln: Deutsch ▼

Einstellungen [Hilfe anzeigen](#)

Newsletter Einstellungen

- ☐ Widget darf nur auf bestimmten URLs erscheinen
- ☒ Mandantenkennung für neue Einträge
 - m2 ▼
 - Mandantenkennung für neue Einträge. [mehr...](#)
- CTP:
Customer touch point (CTP) [mehr...](#)
- ☒ Erweiterte Label-Beschriftungen verwenden
Aktiviert im Formularassistenten zu jedem Formularfeld zwei weitere Beschreibungsfelder, die für Informations- und Hilfetexte im Formular angezeigt werden können. [mehr...](#)
- ☐ Formularspezifische CSS Klasse
Setzt die angegebene CSS-Klasse im HTML-Quellcode im äußersten div-Container.

5.7.2.8 Import von Einträgen

Da die E-Mail-Adresse nicht mehr als global eindeutiges Identifikationsmerkmal eines Eintrags verwendet werden kann, müssen u.U. auch Import-Jobs angepasst werden.

Der Import-Wizard erlaubt gleich im ersten Schritt die Festlegung des Zielmandanten für den Importvorgang:

5.7.2.9 Newsletter-Tracking

Die URLs für das Newsletter-Tracking (Trackerpixel und Link-Redirecter) können bei Bedarf auch mandantenspezifisch angepasst werden. Bisher ist dies jedoch nur möglich, wenn der Newsletterversand per Eventfile und nicht aus der Oberfläche heraus gestartet wird.

Zunächst wird in der `cmsbs.properties`-Datei eine neue globale *SpecialVar* eingeführt:

UM/cmsbs-conf/cmsbs.properties

```
# Globales Tracking, nicht Mandantenspezifisch
custom.special = "trackUrl"
custom.special.trackUrl = "http://www.mydomain.com/p"

cmsbs.tracker.pixel = 
cmsbs.tracker.link =
{special:trackUrl}/t/nl?t={msgid}&d={trackedurl}&h={trackedurl:sha1}&i={trackedurlid}
```

Diese *SpecialVar* soll die URL des zu verwendenden REST-Proxy enthalten. `cmsbs.tracker.pixel` und `cmsbs.tracker.link` werden so gesetzt, dass sie die neue Variable verwenden, um die jeweilige Tracking-URL zu bilden.

Soll nun ein Newsletter mit abweichender Tracking-URL versendet werden, kann dies durch Setzen der *SpecialVar* `trackUrl` im Eventfile erreicht werden:

event.xml

```

<event>
  <destination>...</destination>
  <data>...</data>
  <global special="trackUrl">http://www.mandant1.com/p</global>
  <tag>m1</tag>
</event>

```

5.7.2.10 Verwendung von Absender- und Reply-To-Adressen einschränken

Wie viele Absender- bzw. Reply-To-Adressen verwendet werden dürfen, ist in der Regel durch die Lizenz limitiert. In den beiden Dateien `UM/cmsbs-conf/cmsbs.sender.csv` bzw. `cmsbs.replyto.csv` steht, welche Adressen zurzeit erlaubt sind.

Die Verwendung dieser Adressen kann optional basierend auf dem Mandanten eines Benutzers auf eine Untermenge der generell erlaubten Adressen beschränkt werden. Dabei werden die Kürzel der Mandanten, die eine bestimmte Adresse verwenden dürfen, in die vierte Spalte der CSV-Datei geschrieben:

```

"Vertrieb";vertrieb@example.org;;|m1|m2|
"Newsletter";newsletter@example.org;;|m1|
;info@example.org

```

Es gelten folgende Formatvorgaben:

- Soweit nicht anders konfiguriert, ist das Encoding immer latin1.
- Die einzelnen Spalten werden durch Semikolon getrennt,
- die Spaltenwerte können optional in Anführungszeichen eingeschlossen werden,
- (die Liste der Admin-Rollen in der dritten Spalte muss im Multi-Select-Format (siehe unten) angegeben werden.)
- die Liste der Mandantenkürzel in der vierten Spalte muss im Multi-Select-Format (siehe unten) angegeben werden.
- Bleibt die vierte Spalte leer oder fehlt sie ganz, kann die jeweilige Adresse von allen Mandanten verwendet werden.

(Multi-Select-Format: die einzelnen Werte werden durch Pipe-Zeichen "|" getrennt. Außerdem steht vorn und hinten auch immer ein Pipe-Zeichen.)

Siehe auch [Rechte und Rollen](#) / *Verwendung von Absender- und Reply-To-Adressen einschränken*.

6 Personalisierung

Zur Personalisierung des Newsletters und anderer Texte stehen dem Redakteur Variablen zur Verfügung, die innerhalb geschweiften Klammern in einen Text eingefügt werden können, z.B. kann "Hallo {firstname} {lastname}," zur Erstellung einer einfachen Anrede verwendet werden.

6.1 Verwendbare Variablen

Alle in dem Universal Messenger zu einem Eintrag vorhandenen Attribute können für die Personalisierung verwendet werden. Um den Wert eines Attributs in den Text einzufügen, wird der Name des Attributs in geschweifte Klammern gesetzt. Der Text wird automatisch passend zur Ausgabe kodiert, so dass das Zeichen & bei der Ausgabe im HTML-Newsletter automatisch zu & kodiert wird. Das Kodieren kann durch ein dem Attributnamen vorangestelltes Ausrufezeichen innerhalb der geschweiften Klammer unterbunden werden.

Folgende spezielle Variablen stehen zur Verfügung:

- {channels} fügt die Titel der vom Eintrag abonnierten Channels ein, die Titel werden durch Kommata getrennt ausgegeben.
- {msgid} fügt die Nachrichten-ID ein, die jeden Newsletterversand eindeutig kennzeichnet und für das Tracking verwendet werden kann.
- {msgname} fügt den beim Newsletterversand angegebenen Betreff in der nicht-personalisierten Form ein und kann für das Tracking verwendet werden.
- {trackerpixel} fügt ein Zählpixel gemäß dem in der Konfigurationsoption `cmsbs.tracker.pixel` angegebenen Formatierungsstring in die E-Mail ein. Diese Variable kann nur in einer E-Mail verwendet werden.

Variablen werden in geschweifte Klammern geschrieben.

Beispiel: {company} fügt den Firmennamen ein, {!email} fügt die E-Mail-Adresse ohne Umkodierung zur Anzeige ein.

Benutzerattribute vom Typ TABLE

Bei der Verwendung eines Benutzerattributs vom Typ TABLE muss in der Regel die Zeilennummer der Tabelle angegeben werden, die für die Personalisierung eingefügt werden soll. Mit {table[1]->row} wird der Inhalt der Spalte row der ersten Zeile des Tabellenattributs table eingefügt.

Mit der Angabe von {table->row} wird immer die letzte Zeile der Tabelle angesprochen.

Formatierungsanweisungen

Mit den Formatierungsanweisungen kann die Ausgabe der Personalisierungsvariablen beeinflusst werden.

Formatangabe	Beschreibung
{max:Zahl:{Attribut}}	begrenzt die Ausgabe auf die mit Zahl angegebene Länge
{urlencode:{Attribut}}	kodiert den Inhalt der Variable für die Übergabe in einer URL als GET-Parameter
{urlencode:charset:{Attribut}}	zusätzliche Angabe des Encodings
{nice:Attribut}	formatiert das Attribut nach den auch für die Benutzeroberfläche geltenden Regeln, z.B. bei salut zu den Titeln Herr/Frau
{nice:Attribut:Pattern}	formatiert das Attribut vom Typ TIMESTAMP nach dem angegebenen Pattern im SimpleDateFormat
{nice:birthday:dd.MM.yyyy}	formatiert das Attribut <i>birthday</i> in dem angegebenen (deutschen) Datumsformat

Bei der Formatierungsanweisungen `max` und `urlencode` kann nach dem Doppelpunkt eine beliebige Zeichenkette folgen, die feste Texte und auch Personalisierungsvariablen enthält. Die Personalisierungsvariablen müssen deswegen in geschweiften Klammern angegeben werden, damit sie von dem Universal Messenger erkannt werden.

Mit `charset` kann bei `urlencode` der Zeichensatz für das Encodings angegeben werden. Im Normalfall wird die Standardeinstellung verwendet, für spezielle Anwendungen kann ein Zeichensatz wie z.B. `utf-8` oder `latin9` angegeben werden.

Bei der Formatierungsanweisung `nice` folgt nach dem Doppelpunkt ausschließlich ein Attributname, der deswegen ohne geschweifte Klammern angegeben werden muss.

Der Aufbau des Patterns bei der Formatierung eines Attributs vom Typ `TIMESTAMP` mit `{nice:Attribut:Pattern}` entspricht den Möglichkeiten des `SimpleDateFormat`, die Details entnehmen Sie bitte der Dokumentation des JDK unter <http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html>.

Aktuelles Datum und Uhrzeit

Mit der speziellen Personalisierungsvariable `{now}` können das zum Versandzeitpunkt aktuelle Datum und die aktuelle Uhrzeit in verschiedenen Formaten eingefügt werden. Bei einem Newsletterversand wird als Versandzeitpunkt der Beginn der Aussendung gespeichert, so dass auch bei einem längeren Versand für alle Abonnenten die gleiche Uhrzeit ausgegeben wird.

Formatangabe	Beschreibung
{now}	in den Spracheigenschaften des JDK definiertes langes Format, z.B. Fr 16.Apr.2004, 14:46:55 Uhr
{now:date}	in den Spracheigenschaften des JDK definiertes kurzes Datum, z.B. 16.04.04
{now:time}	in den Spracheigenschaften des JDK definierte kurze Uhrzeit, z.B. 14:46

<code>{now:yyyy-MM-ddHH\:mm\:ss,SSS}</code>	benutzerdefiniertes Format mit Formatierungszeichen, z.B. 2008-04-16 14:46:55,033
---	---

Bei der Angabe eines benutzerdefinierten Formats müssen Doppelpunkte durch einen vorangestellten Backslash \ gekennzeichnet werden. Statt des Doppelpunkts kann ein beliebiges anderes Zeichen als Trennzeichen verwendet werden. Es stehen die [hier](#) beschriebenen Formatierungszeichen zur Verfügung.

E-Mail-Adressen

Nach RFC822 können E-Mail-Adressen neben der eigentlichen E-Mail-Adresse des Empfängers einen Klartextnamen enthalten. Die spezielle Formatierungsanweisung `makeEmailAddress` kann zum leichteren Einfügen dieser standardkonformen E-Mail-Adressen verwendet werden.

Formatangabe	Beschreibung
<code>{makeEmailAddress:Name:eMail-Adresse}</code>	RFC822 konforme E-Mail-Adresse
<code>{makeEmailAddress:Name:eMail-Adresse:charset}</code>	zusätzliche Angabe des Encodings

Ein Beispiel für eine konforme Adresse gemäß RFC822 ist:

Hans Meier <hans@meier.com>

Von `makeEmailAddress` wird der Teil zur Angabe des Empfängernamens in das Quoted Printable Encoding umgesetzt, das für den Mailversand vorgeschrieben ist. Mit `charset` kann der Zeichensatz für dieses Encoding angegeben werden. Im Normalfall wird die Standardeinstellung verwendet, für spezielle Anwendungen kann ein Zeichensatz wie z.B. `utf-8` oder `latin9` angegeben werden.

Symbole für Einträge

Die spezielle Formatierungsanweisung `UserIcons` kann nur zur Konfiguration der Benutzeroberfläche des Universal Messenger verwendet werden und steht zur Personalisierung eines Newsletters nicht zur Verfügung.

Mit der Formatierungsanweisung `UserIcons` können zu Einträgen Symbole (Icons) angezeigt werden, wenn eine vordefinierte Abfrage für den aktuell angezeigten Eintrag erfüllt ist, siehe [Symbole für Einträge](#).

Formatangabe	Beschreibung
<code>{UserIcons:*}</code>	alle zum Eintrag passenden Symbole
<code>{UserIcons:Symbol1:Symbol2}</code>	nur die genannten zum Eintrag passenden Symbole

6.2 Kontrollstrukturen

Mit den Kontrollstrukturen ist eine sehr flexible Personalisierung möglich, für die Sie beliebige Attribute auswerten können. Kontrollstrukturen bestehen aus Abfragen und zugeordneten Textabschnitten, die unter bestimmten Bedingungen eingefügt werden.

In den Kontrollstrukturen werden in den Beispielen senkrechte Striche bzw. Pipes "|" als Trennzeichen zur Trennung von Abfragen und Textabschnitten verwendet. Als Trennzeichen kann anstelle der Pipe ein beliebiges Zeichen verwendet werden, da immer automatisch das erste Zeichen nach der switch- bzw. if-Anweisung als Trennzeichen interpretiert wird. Das gewählte Trennzeichen kann in den Textabschnitten der Kontrollstrukturen wegen seiner Sonderfunktion dann nicht mehr normal verwendet werden. Falls dieses Trennzeichen in einem Textabschnitt dennoch normal ausgegeben werden soll, muss es durch einen vorangestellten Backslash \ gekennzeichnet werden.

Leerzeichen an Anfang und Ende der Textabschnitte werden nicht ausgegeben, so dass die Kontrollstrukturen an diesen Stellen frei formatiert werden können. Die if-Anweisung kann in einer Variante ohne das automatische Entfernen der Leerzeichen verwendet werden, indem die Schlüsselworte `if`, `elif` und `else` mit einem großen Anfangsbuchstaben geschrieben werden.

In den Textabschnitten können wiederum Personalisierungsvariablen und Kontrollstrukturen verwendet werden, die rekursiv ausgewertet werden. Bei geschachtelten Kontrollstrukturen müssen die Pipes der inneren Kontrollstruktur mit einem vorangestellten Backslash \ angegeben werden, damit diese richtig zugeordnet werden können. Für jede Ebene der Verschachtelung muss ein weiterer Backslash der Pipe vorangestellt werden.

Bei mehreren Verschachtelungsebenen empfehlen wir, verschiedene Sonderzeichen als Trennzeichen zu nutzen. Die Verwendung der UTF8-Sonderzeichen ^{1 2 3 4} hat sich hier als praktische Alternative zur Pipe erwiesen, da so die verschiedenen Ebenen leichter zu erkennen sind.



Da es bei Verwendung von Sonderzeichen immer wieder zu Problemen kommt, achten Sie bitte darauf, dass Sie einheitlich UTF-8 als Zeichenkodierung verwenden. Mit UTF-8 können Sie bei den Trennzeichen auch auf eine größere Anzahl an Sonderzeichen zurückgreifen, falls notwendig.



Fehler in den Kontrollstrukturen werden direkt in den ausgegebenen Text eingefügt, so dass Sie die Kontrollstrukturen durch den Versand einer Vorschau unbedingt testen sollten.

6.2.1 switch-Anweisung

Attributen mit einem festen Wertebereich kann einzelnen Werten über eine switch-Anweisung eine Zeichenkette zur Ausgabe zugeordnet werden. Diese Anweisung wird häufig zusammen mit der Anrede eingesetzt, z.B.:

```
{switch|salut
  |male |Sehr geehrter Herr {lastname},
  |female |Sehr geehrte Frau {lastname},
  |family |Sehr geehrte Familie {lastname},
  |company|Sehr geehrte Damen und Herren,
  |default|Sehr geehrte Damen und Herren,
}
```

Bis zur ersten Pipe muss die switch-Anweisung direkt der geschweiften Klammer folgen, die weitere Formatierung kann frei gewählt werden. Auf die switch-Anweisung folgt zunächst das Attribut, das ausgewertet werden soll. Danach folgt jeweils durch Pipes getrennt eine Liste der Werte und der für die Ausgabe zugeordneten Zeichenketten. (Das Komma im Beispiel gehört zur Zeichenkette und ist für die Syntax der switch-Anweisung nicht notwendig.) Falls der Inhalt des Attributs keinem der angegebenen Werte entspricht, wird die Zeichenkette nach `default` ausgegeben.

6.2.2 if-Anweisung

Mit der if-Anweisung können die zur Abfrage auf Abonentendaten verfügbaren Operatoren für die Personalisierung der Newsletter und Benachrichtigungen verwendet werden, siehe [Abfragesprache](#). Es können die aus Programmiersprachen üblichen if/else-Kontrollstrukturen gebildet werden, z.B.:

```
{if|orders == 10|
  Sie haben 10-mal bestellt.
|elif|orders > 10| Sie haben mehr als 10-mal bestellt.
|else| Sie haben weniger als 10-mal bestellt.
}
```

Bis zur ersten Pipe muss die if-Anweisung direkt der geschweiften Klammer folgen, die weitere Formatierung kann frei gewählt werden. Der der if-Anweisung und der Abfrage folgende Textabschnitt wird eingefügt, wenn die Abfrage `true` liefert. Über optional folgende elif-Anweisungen können weitere Abfragen und Textabschnitte eingefügt werden. Die else-Anweisung wird ausgeführt, wenn keine der vorangehenden Abfragen `true` lieferte.

Die if-Anweisung kann in einer Variante ohne das automatische Entfernen der Leerzeichen am Anfang und Ende der Textabschnitte verwendet werden, indem die Schlüsselworte `If`, `Elif` und `Else` mit einem großen Anfangsbuchstaben geschrieben werden.

6.2.3 foreach-Anweisung

Mit der `foreach`-Anweisung können alle Zeilen eines Attributs vom Typ `TABLE` durchlaufen und für die Personalisierung verwendet werden, z.B.:

```
{foreach|tabellenattribut|
  Die Tabelle enthält keine Einträge.
  |
  Jetzt folgt die Liste der Zeilen.
  |
  Diese Zeile enthält {tabellenattribut->spaltenname}.
  |
  Das Ende der Tabelle ist erreicht.
}
```

Bis zur ersten Pipe muss die `foreach`-Anweisung direkt der geschweiften Klammer folgen, die weitere Formatierung kann frei gewählt werden. Der der `foreach`-Anweisung und dem Namen des Tabellenattributs folgende Textabschnitt wird eingefügt, wenn das Tabellenattribut leer ist. Nach der nächsten Pipe folgt der initial ausgegebene Text, danach die Ausgabe für jede Zeile des Tabellenattributs und nach der letzten Pipe der abschließend ausgegebene Text.

6.2.4 ifcse-Anweisung

Mit `{ifcse}` können Textblöcke in Abhängigkeit von einer in CSE-Code formulierten Bedingung ein- oder ausgeblendet werden. Diese Bedingung muss in Form einer Methode des `RenderEntryCallback` vorliegen. Beim Aufruf dieser Methode können beliebige statische Argumente übergeben werden. Der jeweilige Eintrag im Universal Messenger ist für die Methode ebenfalls erreichbar. Die Syntax der `{ifcse}`-Anweisung entspricht der des gewöhnlichen `{if}` mit der Ausnahme, dass die Bedingung in Form eines CSE-Methodenaufrufs anstatt einer Query im Universal Messenger angegeben wird.

Das folgende Beispiel ruft die Funktion `RenderEntryCallback.prototype.firstnameStartsWith()` auf, um festzustellen, ob der Vorname des Empfängers eines Newsletters mit einem bestimmten Buchstaben beginnt.

```
Vorname beginnt mit {ifcse|firstnameStartsWith('a') | A|elif|firstnameStartsWith('b') | B
|elif|firstnameStartsWith('c') | C
|elif|firstnameStartsWith('d') | D
|elif|firstnameStartsWith('e') | E
|elif|firstnameStartsWith('f') | F
|else|anderem Buchstaben}
```

Die Implementierung dieser Funktion muss im Callback-Scope verfügbar gemacht werden. Sie sollte also im Verzeichnis `.../cmsbs-conf/cse/callback/` abgelegt werden.

```
RenderEntryCallback.prototype.firstnameStartsWith = function(firstLetter)
{
    var firstname = this.entry.get('firstname');

    return (firstname.length > 0 && firstname[0].toLowerCase() == firstLetter);
};
```

Wie im Beispiel zu sehen ist, kann die Methode sowohl auf die beim Aufruf übergebenen Argumente (Anfangsbuchstabe) als auch über `this.entry` auf den Eintrag im Universal Messenger des Empfängers zugreifen.

6.2.4.1 ifcse|specialVarIs

Mit `ifcse|specialVarIs` kann der Inhalt einer SpecialVar mit der angegebenen Zeichenketten verglichen werden.

Das Beispiel zeigt das Datenfeld `salut` für die Anrede und `lastname` für den Nachnamen eines Veranstaltungsteilnehmers:

```
{ifcse|specialVarIs("salut", "female")|Sehr geehrte Frau {special:lastname},
|elif|specialVarIs("salut", "male")|Sehr geehrter Herr {special:lastname},
|elif|specialVarIs("salut", "family")|Liebe Familie {special:lastname},
|else|Sehr geehrte Damen und Herren,
}
```

Die korrespondierende CSE-Methode ist `RenderEntryCallback.prototype.specialVarIs`.

6.2.4.2 ifcse|specialVarContains

Mit `ifcse|specialVarContains` kann geprüft werden, ob der Inhalt einer SpecialVar einen bestimmten Wert enthält.

Das Beispiel zeigt das Datenfeld `colors` mit einen Multiselect :

```
{ifcse|specialVarContains("colors", "red")|Sie haben Rot gewählt.}
{ifcse|specialVarContains("colors", "blue")|Sie haben Blau gewählt.}
{ifcse|specialVarContains("colors", "yellow")|Sie haben Gelb gewählt.}
```

Die korrespondierende CSE-Methode ist `RenderEntryCallback.prototype.specialVarContains`.

6.2.4.3 ifcse|specialVarIsSet

Mit `ifcse|specialVarIsSet` kann geprüft werden, ob der Inhalt einer SpecialVar nicht-leer ist;
Beispiel:`{ifcse|specialVarIsSet("name")| Value if specialVar is not empty}`



Die Datenfelder, die in einem Kontaktformular als Checkbox genutzt werden, müssen über den zusätzlich bereitgestellten "nice"-Wert abgefragt werden.`{ifcse|specialVarIsSet("cf_topic05_nice")| Value if specialVar is not empty}`

Die korrespondierende CSE-Methode ist `RenderEntryCallback.prototype.specialVarIsSet`.

6.2.4.4 ifcse|specialVarIsEmpty

Mit `ifcse|specialVarIsEmpty` kann geprüft werden, ob der Inhalt einer SpecialVar leer ist;

Beispiel:`{ifcse|specialVarIsEmpty("name")| No value}`



Die Datenfelder, die in einem Kontaktformular als Checkbox genutzt werden, müssen über den zusätzlich bereitgestellten "nice"-Wert abgefragt werden.`{ifcse|specialVarIsEmpty("cf_topic05_nice")| No value}`

Die korrespondierende CSE-Methode ist `RenderEntryCallback.prototype.specialVarIsEmpty`.

6.2.5 withcse-Anweisung

Mit `{withcse}` kann ein beliebig personalisierter Textblock an eine CSE-Funktion übergeben werden. Die CSE-Funktion kann diesen Textblock verarbeiten, ggf. modifizieren und entscheiden, ob er schließlich in den Text übernommen werden soll oder nicht.

Im folgenden Beispiel wird die CSE-Methode `RenderEntryCallback.prototype.encloseIn()` aufgerufen. Als Textblock wird die E-Mail-Adresse des Eintrags im Universal Messenger übergeben. Die Funktion soll den übergebenen Textblock in Klammern einschließen. Dazu werden die gewünschten Klammerzeichen als zwei statische Zeichenketten übergeben.

```
{withcse|encloseIn('[', '']|{email}}
```

Die Implementierung dieser Funktion muss im Callback-Scope verfügbar gemacht werden. Sie sollte also im Verzeichnis `.../cmsbs-conf/cse/callback/` abgelegt werden.

```
RenderEntryCallback.prototype.encloseIn = function(left, right)
{
  this.value = left + this.value + right;

  return true;
};
```

`encloseIn()` kann hier über `this.value` lesend und schreibend auf den Textblock zugreifen. Liefert sie am Ende `true` zurück, wird der letzte Wert von `this.value` in den Ausgabertext eingefügt. Bei `false` wird nichts eingefügt.

6.2.5.1 `_pre()`-Methode

Optional kann eine Methode mit gleichem Namen erweitert mit `"_pre"` als Namenssuffix implementiert werden. Diese Methode wird dann noch vor der Personalisierung des Textblocks aufgerufen.

Das folgende Beispiel zeigt einen Anwendungsfall, bei dem anhand des per `{forceSend}` bzw. `{suppressSend}` beeinflussbaren `doSend`-Flags entschieden wird, ob der übergebene Textblock semantisch leer ist oder ob er angezeigt werden soll. Die `netEmpty_pre()`-Funktion wird hier benötigt, um einen Stack aufzubauen, der zum Zwischenspeichern dieses `doSend`-Flags dient.

```
/**
 * Ueberschriften einblenden, wenn es darin Artikel gibt (schachtelbar).
 * {withcse:notEmpty():...}
 */
RenderEntryCallback.prototype.netEmpty_pre = function()
{
    if (!this.doSendStack)
        this.doSendStack = [];

    // aktuellen zustand sichern
    this.doSendStack.push(this.templateContext.doSend);
    this.templateContext.doSend = false;

    return true;
};

/**
 * Stack anschauen und globales doSend richtig setzen.
 */
RenderEntryCallback.prototype.netEmpty = function()
{
    var doSendFragment = this.templateContext.doSend;
    var doSendGlobally = this.doSendStack.pop();

    if (!doSendFragment) {
        this.value = "<!-- Fragment suppressed -->";
    }

    this.templateContext.doSend = (doSendGlobally || doSendFragment);

    return true;
};
```


6.2.6 Kontrollstrukturen für Split-Test-Newsletterversand

Bei einem Split-Test-Newsletterversand werden mehrere unterschiedliche Varianten eines Newsletters an eine Untermenge eines Zielsegments versendet. Nach Auswertung des Klick- und Öffnungsverhaltens wird dann eine Gewinnervariante ausgewählt, die im Anschluss für den Versand an den restlichen (Groß-)Teil der Empfänger verwendet wird. (Siehe dazu auch [Split-Testing](#))

Die in diesem Abschnitt beschriebenen Kontrollstrukturen ermöglichen das gezielte Ein- bzw. Ausblenden einzelner Abschnitte des Newsletterinhalts für einzelne Split-Test-Varianten.

Ein Split-Test umfasst immer zwei oder mehr Varianten, die jeweils mit 1 beginnend durchnummeriert werden.

6.2.6.1 splitTestSwitch-Anweisung

Diese Anweisung ermöglicht es, einen Abschnitt des Inhalts für jede Variante unterschiedlich zu gestalten.

Das folgende Beispiel erzeugt fünf unterschiedliche personalisierte Anreden:

```
{splitTestSwitch|Lieber {firstname},|Hallo {firstname},|Guten Tag, {firstname},|Servus {firstname},|Hallo,}
```

6.2.6.2 splitTestIf-Anweisung

Diese Anweisung blendet einen Abschnitt nur für eine oder mehrere benannte Varianten ein. In allen anderen Varianten wird der Abschnitt weggelassen.

```
{splitTestIf|1,3,6|Das kommt nur in die Varianten 1, 3 und 6}
{splitTestIf|2|Und das kommt nur in die Variante 2}
```

6.3 Verwendbare Funktionen

Inhalt einer personalisierten URL einfügen

Mit Hilfe der Personalisierungsfunktion `includeUrl` ist es möglich, den Inhalt einer personalisierten URL in den Newslettertext zu integrieren.

```
{includeUrl|URL|ENCODING}
```

Diese Personalisierungsfunktion bewirkt, dass während des Newsletterversands für jeden einzelnen Empfänger die angegebene URL aufgerufen und der Inhalt ausgelesen wird. Die URL kann auch wiederum Personalisierungsvariablen enthalten, so dass es zum Beispiel möglich ist, die UID oder die E-Mail-Adresse als Parameter zu übergeben. Der ausgelesene Inhalt wird ebenfalls personalisiert und in den Newslettertext eingefügt. Sollte dieser Inhalt ein öffnendes und ein schließendes `BODY`-Tag enthalten, so wird nur der Inhalt zwischen diesen beiden Tags in den Newsletter eingefügt.

Für die URL sind die Schemata "file:" und "http(s):" zulässig.

Die Angabe des Encoding ist optional. Fehlt die Angabe, wird das Encoding aus dem Content-Type-HTML-Header der abgerufenen Datei verwendet. Fehlt die Angabe an dieser Stelle ebenfalls, wird das Encoding aus dem HTTP-Content-Type-Header verwendet.

Beispiel 1: `{includeUrl|http://localhost:8080/service.jsp?t={msgid}&uid={uid}}`

Ruft für jeden Empfänger die Datei `http://localhost:8080/service.jsp` per HTTP ab und übergibt dabei sowohl das DeliveryTicket (msgid) als auch die UID des Empfängers als GET-Parameter.

Beispiel 2: `{includeUrl|file:/home/um/personal/{lastname}-{firstname}.html|UTF-8}`

Lädt den einzufügenden Textbaustein jeweils aus einer Datei, deren Name sich aus Nach- und Vorname des Empfängers zusammensetzt. Als Encoding wird UTF-8 verwendet.

Hinweise:

- Diese Funktion dient nur zum Einfügen von Plaintext- oder HTML-Fragmenten in den Newslettertext.
- Das per `includeUrl` eingefügte Textfragment darf fast alle verfügbaren Personalisierungsfunktionen enthalten.
- Die Personalisierungsfunktion `attach` kann im eingefügten Textfragment nicht verwendet werden.
- Grafiken und Stylesheets, die im eingefügten Textfragment referenziert werden, werden nicht automatisch in die E-Mail eingebettet.

Fehlerbehandlung: Sollte es beim Abrufen der angegebenen URL zu einem Fehler kommen, wird der betroffene Empfänger beim Versand zunächst übersprungen. HTTP-Status-Codes ab 400 gelten in diesem Zusammenhang ebenso als Fehler wie Probleme auf Netzwerkebene. Am Ende kann der Versand ggf. manuell im Newsletterarchiv fortgesetzt werden, um den Newsletter an die bisher übersprungenen Empfänger erneut versuchen zu senden.

Newsletter-Versand unterdrücken bzw. erzwingen

Der Newsletter wird personalisiert versendet, d.h. dem Empfänger werden nur die Themen angezeigt, für die dieser sich angemeldet hat. Wenn keiner der im Newsletter enthaltenen Inhalte gemäß Personalisierungsfunktionen für den User relevant ist, würde das dazu führen, dass ein leerer Newsletter an ihn versendet werden würde. Mit den im Folgenden beschriebenen Funktionen kann der Versand solcher leerer Newsletter vermieden werden.

Im Newslettertemplate werden dafür zwei Personalisierungsfunktionen genutzt: `suppressSend` und `forceSend`.

Zunächst wird am Anfang des Rumpfes das Schlüsselwort `{suppressSend}` abgelegt. Dadurch wird der Versand zunächst unterdrückt. In allen Textelementen, die den Newsletter "nicht mehr leer" machen, wird dann `{forceSend}` mit abgelegt. Dadurch wird der Versand veranlasst, sobald durch die Personalisierung ein echter Inhalt in den Rumpf gelangt.

Beispiel für das Newslettertemplate:

```
{suppressSend}
... html Gerüst, feste Einleitung, Überschriften etc ...

{if|inChannel('x')|{forceSend} Inhalt}
{if|inChannel('y')|{forceSend} Anderer Inhalt}

... Gerüst, fester Abspann ...
```

Fehlermeldungen unterdrücken

Bei folgenden Prädikaten der Querylanguage kann jetzt ein "@" vorangestellt werden, um Fehlermeldungen zu unterdrücken, die durch das Fehlen des Channels oder VChannels hervorgerufen werden:

- `@inChannel`
- `@notInChannel`
- `@inChannelMail`
- `@notInChannelMail`
- `@inAnyChannel`
- `@notInAnyChannel`
- `@inVChannel`

Prädikate, welche die Existenz eines Channels oder VChannels prüfen, liefern `FALSE`, falls er nicht mehr existiert.

Prädikate, welche die Nicht-Existenz eines Channels oder VChannels prüfen, liefern `TRUE`, falls er nicht mehr existiert.

Hintergrund: Newsletter-Pflege und Channel-Pflege werden unter Umständen unterschiedlich aktuell gehalten, so dass es vorkommen kann, dass in einem Newsletter eine Channel-Abfrage enthalten ist, die auf einen Channel zeigt, der bereits gelöscht wurde. In diesem Fall würde der Versand abgebrochen.

Escaping und Unicodezeichen

Da alle Personalisierungskonstrukte mit geschweiften Klammern umschlossen werden, müssen diese Zeichen dementsprechend mittels `{#x7b}` (für "{") bzw. `{#x7d}` (für "}") umschrieben werden, wenn sie nicht als Personalisierungsfunktion interpretiert werden sollen.

Auf diese Weise können auch beliebige andere Unicode-Zeichen eingefügt werden – beispielsweise `{#x1F984}` oder `{#129412}` für das Unicode-Einhorn.

6.4 Benutzerdefinierte Anredeformel

Mit den Personalisierungsvariablen kann eine benutzerdefinierte Anredeformel zusammengestellt werden, die für den Versand eines Newsletters verwendet werden kann. Die Anredeformel erlaubt die Zusammenfassung der Kontrollstrukturen und Personalisierungsvariablen zu einer Einheit. Es können verschiedene Anredeformeln für z.B. eine förmliche Anrede ("Sehr geehrte Damen und Herren") oder eine vertraulichere Form ("Liebe Freunde") definiert werden.

Die Konfiguration der Anredeformeln erfolgt in der Konfigurationsdatei

`<UM_HOME>/cmsbs-conf/additional.attributes` bzw. in der in den Konfigurationseinstellungen definierten Datei.

```
standard.saluts = <Liste der Anredeformeln>
```

oder

```
standard.saluts[] = <Anredeformel>
```

Durch Leerzeichen getrennte Liste der (internen) Namen der Anredeformeln. Wird die zweite Variante verwendet, ist die Position der Anredeformeln durch die Reihenfolge in der Datei bestimmt. Die Namen dürfen nur die Buchstaben A bis Z, a bis z und die Zahlen 0 bis 9 enthalten. Für jedes Anredeformel müssen die folgenden Optionen angegeben werden.

```
salutgrp.<Anredeformel>.title = <Titel zur Anzeige>
```

Der Titel der Anredeformel wird in der Benutzeroberfläche bei der Zusammenstellung eines neuen Newsletters zur Auswahl angezeigt.

```
salutgrp.<Anredeformel>.value.<Attributwert> = <Personalisierungstext>
```

Den verschiedenen erlaubten Werten des vordefinierten Attributs `salut` wird jeweils ein Personalisierungstext zugeordnet. Für jeden definierten Wert (auch den leeren Wert) muss diese Option angegeben werden. Als besonderer Attributwert kann mit `default` ein Personalisierungstext für alle anderen, nicht definierten Werte vorgegeben werden.

Eine förmliche Anredeformel könnte als Beispiel mit folgenden Angaben definiert werden:

```
salutgrp.offiziell.title= "Förmliche Anrede - Sehr geehrte"
salutgrp.offiziell.value.male = "Sehr geehrter Herr {lastname},"
salutgrp.offiziell.value.female = "Sehr geehrte Frau {lastname},"
salutgrp.offiziell.value.family = "Liebe Familie {lastname},"
salutgrp.offiziell.value.company = "Sehr geehrte Firma {company},"
salutgrp.offiziell.value. = "Sehr geehrte Damen und Herren,"
salutgrp.offiziell.default = "Sehr geehrte Damen und Herren,"
```

6.5 Attachments

Es ist möglich, an ausgehende E-Mail-Newsletter zusätzliche Attachments (Dateianhänge, z.B. PDF) anzuhängen. Dazu gibt es prinzipiell zwei Möglichkeiten.

In der Event-Datei

Im `<email>`-Element können beliebig viele `<file>`-Elemente vorkommen, welche Pfade oder beliebige URLs enthalten können. Die Dateien werden vor dem Versand einmalig heruntergeladen. Beim Versand kann der Dateiname geändert werden.

Beispiele:

```
<file>/pfad/zu/foo.gif</file>

<file name="bar.gif">file:///pfad/zu/foo.gif</file>
<file name="infos.pdf">http://intranet.example.com/35j35hj5235.pdf</file>
```

Im Rumpf

Auch beim Personalisieren eines Text- oder HTML-Rumpfs kann das Anhängen weiterer Dateien angefordert werden. Ähnlich wie bei `{suppressSend}`/`{forceSend}` werden dabei nur die Dateien verschickt, die beim Personalisieren erreicht werden. Jede E-Mail einer Newsletter-Aussendung kann also individuelle Attachments enthalten. Auch hier ist es möglich, lokale Dateien und beliebige URLs zu verwenden, wobei der Dateiname angepasst werden kann. Auch der Content-Type kann ggf. geändert werden, was aber selten nötig ist.

Beispiele:

```
{attach|/pfad/zu/foo.gif}
{attach|file:///pfad/zu/foo.gif|bar.gif}
{attach|foo.gif|bar.gif|application/x-octetstream}
```

Relative Dateinamen werden von der `downloadUrl` bezogen, wie das auch für die referenzierten Bilder des HTML-Rumpfes gilt. Der Pfad kann Special-Variablen der Art `{special:foo}` enthalten. Personalisierende Variablen aus Attributen eines Eintrags sind nicht möglich. Die Dateien werden vor dem Versand heruntergeladen und sind während der Personalisierung fest. Individuelle Dateien (z.B. Rechnungen) können mit dieser Methode nicht angehängt werden.

Beispiel:

```
{if$defined(firstname)$
  Mit Vorname: {attach|{special:localPath}|mit_firstname.gif}
$else$
  Ohne Nachname: {attach|http://{special:remotePath}/foo.pdf|bar.pdf}
}
```

Der Personalisierungsausdruck `{attach|...}` selbst ist im personalisierten Rumpf nicht mehr vorhanden (leerer String).

Allgemein

Es empfiehlt sich, die Pflege von Dateianhängen direkt im HTML-Newsletter zu realisieren und dann die dafür verwendeten CMS-Elemente mit den Logiken des CMS (z.B. Linklisten) in der Event-Datei auszugeben. So können Redakteure zentral alle Inhalte pflegen und müssen nicht zwischen Newsletter und Event-Datei wechseln.

Bitte beachten Sie, dass Dateianhänge an jeden Empfänger versendet werden. Dateianhänge können somit einen großen Einfluss auf die Versandgeschwindigkeit und die zu versendende Datenmenge haben. Die Datenmenge beeinflusst u.a. auch den Speicherplatz, den der Mailserver zur Pufferung noch nicht versendeter Mails bereit halten muss.

Das Schema `file:` kann absolute und relative Pfade aufnehmen. Der Wert der Einstellung `cmsbs.server.home` wird für die Auswertung von relativen Pfaden genutzt.

6.6 Eigene Variablen

Eventdatei

In der Eventdatei können Sie mit dem XML-Element `<global>` eigene Variablen definieren, die im Newsletter z.B. zur Personalisierung oder zum Tracking verwendet werden können.

Beispiel einer Definition in der Eventdatei:

```
<global special="etracker" trim="true">XXXXX</global>
```

CSE und Connectoren

In CSE und den Connectoren können eigene Variablen ebenfalls vor dem Versenden von Newslettern und Benachrichtigungen angegeben werden.

Beispiel für die CSE:

```
var n = entry.notifications.add(<Name der Benachrichtigung>);  
n.specialVars['url'] = 'http://example.org/';  
n.specialVars['foo'] = 'bar';
```

Weitere Details finden Sie in den jeweiligen API-Dokumentationen.

Globale Variablen

Globale Variablen können in der `cmsbs-conf/cmsbs.properties` definiert werden:

```
custom.special = "<Namen der Variablen>"  
custom.special.<name> = "<value>"
```

Das ist beispielsweise sinnvoll bei unterschiedlichen URLs für Test- und Entwicklungssysteme, die so global an einer Stelle geändert werden können.

Direktive zum Setzen einer globalen Variable

Im Template selbst kann eine globale Variable mit folgender Direktive gesetzt werden:

```
{setSpecial:unsubscribeUrl:https://www.my-company.com/unsubscribe}
```

Diese Variante ist vor allem im Zusammenhang mit dem [Smart Editor](#) nützlich.

Der so übergebene Wert kann keine Personalisierungsstrukturen und keine empfängerspezifischen Daten enthalten, da die Auswertung der `setSpecial`-Direktive schon beim Parsen des Templates und nicht erst beim Rendern des Newsletters erfolgt.

Ausgabe

Die eigene Variable kann im Newsletter dann wie folgt zur Personalisierung eingebunden werden:

```
{special:etracker}
```

Die eigenen Variablen können auch mit den üblichen Formatierungsanweisungen im Newsletter verwendet werden:

```
{urlencode:{special|etracker}}
```

Beispiel für die Verwendung von eigenen Variablen in Benachrichtigungen (Notifications) bzw. im Newsletter:

Klicken Sie hier: <{special:publicHost}/confirm?c={msgid}>

Beispiel für Tracking-Url in der `cmsbs-conf/cmsbs.properties`:

```
cmsbs.tracker.pixel = "<img src='{special:publicHost}/track/nt?t={msgid}' />"
```


7 Abfragesprache

Mit der hier beschriebenen Abfragesprache werden Suchkriterien definiert, mit denen die gespeicherten Einträge durchsucht werden können. Die Abfragesprache wird in der Benutzeroberfläche und über die Programmierschnittstellen zur Suche nach Einträgen, in der Konfigurationsdatei für die benutzerdefinierten Statistiken, für die Personalisierung und in der XML-Eventdatei verwendet.

7.1 Aufbau einer Abfrage

Eine Abfrage ist eine (optional in Klammern gesetzte) Liste von `AND` und `OR` Ausdrücken. Wenn keine Klammern gesetzt werden, sind die beiden Operatoren linksbündig, `AND` hat eine höhere Priorität als `OR`. Diese abstrakte Beschreibung entspricht dem Verhalten, wie Sie es aus Programmiersprachen oder SQL gewohnt sind. Eine Abfrage entspricht also im Prinzip dem `WHERE`-Teil eines `SQL-Selects`.

Seit Version 7.34.0 ist auch der `NOT`-Operator verfügbar. Der zu negierende Ausdruck muss immer in Klammern gesetzt werden.

Ein Ausdruck ist meistens eine Zeichenkette im Format "Attributname Operator Wert". Der Attributname gibt den Namen eines Attributs in dem Universal Messenger an. Als Werte sind Zeichenketten (in einfachen oder doppelten Anführungszeichen) oder ganze Zahlen möglich.


In dem Universal Messenger werden Attribute nur gespeichert, wenn sie einen definierten und von Null abweichenden Wert haben. Dies ist anders, als in SQL. Mit diesem Verfahren kann erheblich Speicherplatz gespart werden, da zu einem Eintrag häufig nur einige wenige Informationen bekannt sind, vom System aber eine große Menge an Attributen bereitgestellt wird.

Als Null gilt bei Zeichenketten der Leerstring, bei ganzen Zahlen die 0 und bei Wahrheitswerten das `false`. Direkte Abfragen und Vergleiche auf diese Nullwerte mit den Operatoren `==` und `!=` sind nicht möglich und werden deshalb intern automatisch umgeschrieben, für den Anwender bleibt diese Konvertierung unbemerkt.

Falls Sie andere Operatoren wie z.B. `>` oder `>=` in Abfragen einsetzen, müssen Sie bedenken, dass Einträge mit Attributen mit einem Nullwert nicht als Treffer der Abfrage gefunden werden. Das betrifft auch einen Vergleich z.B. `< 2`, da hier die Einträge gleich dem Nullwert nicht in der Treffermenge enthalten sein werden. Weiterhin ist ein Vergleich mit den Nullwerten nicht möglich, sondern muss mit den im folgenden beschriebenen Operatoren für Nullwerte erfolgen.

Beim Vergleich von zwei Attributen miteinander sollten Sie beachten, dass die Abfrage nicht für die Einträge ausgeführt wird, bei denen mindestens eines der verglichenen Attribute einen Nullwert hat. Diese Einträge sind also nicht in der Treffermenge enthalten, auch wenn die Attribute bei beiden Einträgen einen Nullwert haben.

Abfrage	Beschreibung
<code>cmsbs.autoresponds > cmsbs.softbounces</code>	Die Einträge mit <code>softbounces</code> gleich 0 werden nicht gefunden, da <code>softbounces</code> bei diesen Einträgen einen Nullwert enthält.
<code>firstname == lastname</code>	Die Einträge, bei denen der Vorname oder der Nachname nicht angegeben ist, werden nicht gefunden.
<code>defined(firstname)</code>	Es werden nur die Einträge gefunden, bei denen ein Vorname angegeben ist.
<code>cmsbs.hardbounces > 0</code>	Ein direkter Vergleich mit einem Nullwert ist nicht möglich, es werden keine Treffer gefunden.
<code>cmsbs.hardbounces < 2</code>	Im Ergebnis fehlen die Einträge mit <code>hardbounces</code> gleich Null, der korrekte Ausdruck müsste lauten: <code>cmsbs.hardbounces < 2 or undefined(cmsbs.hardbounces)</code>
<code>cmsbs.isadmin == status</code>	Beide Attribute sind vom Typ <code>BOOLEAN</code> . Es werden nur die Einträge mit einem <code>true</code> für beide Attribute gefunden, ein <code>false</code> in beiden Attributen führt wegen des Nullwerts nicht zu einem Treffer.

 Die Abfragesprache enthält kein allgemeines `NOT`, so dass sich Gegenteile/Ausschlüsse nicht direkt formulieren lassen. Wir empfehlen die Verwendung des Operators `notInVChannel(vchannelid)`. Mit diesem Operator können Sie Ergebnisse von gespeicherten Abfragen von Ihrer aktuellen Abfrage ausschließen.

Innerhalb von Abfragen können Kommentare eingefügt werden.

```
firstname = "Fritz" /* Kommentar */
```

7.2 Operatoren

Es können die von SQL gewohnten Operatoren und einige zusätzliche Operatoren verwendet werden, die sich je nach Typ des Werts (Zeichenkette oder ganze Zahl) unterscheiden.

Operatoren für Nullwerte

Operator	Beschreibung
<code>defined(attribute)</code>	Attribut definiert, enthält einen Wert.
<code>undefined(attribute)</code>	Attribut nicht definiert, enthält keinen Wert.

In dem Universal Messenger werden keine Nullwerte gespeichert. Um Attribute trotzdem auf diese Werte abzufragen, müssen die hier beschriebenen Operatoren verwendet werden.

Der Name des Attributs wird in der Klammer direkt ohne Anführungszeichen angegeben. Für spezielle Abfragen auf Sekundärattributen kann in der Klammer der Operator `like` verwendet werden, um den Suchausdruck mit dem Jokerzeichen auf mehrere Attribute zu beziehen. Z.B. findet `defined(like 'liefer%')` alle Einträge, bei denen eines der mit "liefer" beginnenden Attribute zur Angabe der Lieferadresse gesetzt ist.

Operatoren für Wahrheitswerte

Operator	Beschreibung
<code>isSet(attribute)</code>	Attribut ist true
<code>isNotSet(attribute)</code>	Attribut ist false

Die Abfrage von Wahrheitswerten kann mit den hier vorgestellten Operatoren erfolgen. Alternativ können auch die Ziffern 0 und 1 oder die Zeichenketten `'true'` und `'false'` verwendet werden, die dann von dem Universal Messenger automatisch konvertiert werden.

Operatoren für Zeichenketten

Operator	Beschreibung
<code>=</code>	gleich
<code>==</code>	gleich
<code>!=</code>	ungleich
<code><></code>	ungleich
<code>LIKE</code>	gleich mit Jokerzeichen
<code>ILIKE</code>	gleich mit Jokerzeichen ohne Beachtung der Groß/Kleinschreibung
<code>equals(Attribut, Wert)</code>	Attribut ist gleich der angegebenen Zeichenkette
<code>iequals(Attribut, Wert)</code>	Attribut ist gleich der angegebenen Zeichenkette ohne Beachtung der Groß/Kleinschreibung
<code>contains(Attribut, Wert)</code>	Attribut enthält die angegebene Zeichenkette
<code>icontains(Attribut, Wert)</code>	Attribut enthält die angegebene Zeichenkette ohne Beachtung der Groß/Kleinschreibung
<code>startsWith(Attribut, Wert)</code>	Attribut beginnt mit der angegebenen Zeichenkette
<code>istartsWith(Attribut, Wert)</code>	Attribut beginnt mit der angegebenen Zeichenkette ohne Beachtung der Groß/Kleinschreibung
<code>endsWith(Attribut, Wert)</code>	Attribut endet mit der angegebenen Zeichenkette
<code>iendsWith(Attribut, Wert)</code>	Attribut endet mit der angegebenen Zeichenkette ohne Beachtung der Groß/Kleinschreibung

Beim Vergleich mit Zeichenketten können die Jokerzeichen % für beliebig viele Zeichen und _ für ein beliebiges Zeichen verwendet werden. Für die Suche nach einem Zeilenumbruch kann der Zeilenumbruch direkt innerhalb der Anführungszeichen angegeben werden.

Bei den Operatoren `[i]equals`, `[i]contains`, `[i]startsWith` und `[i]endsWith` entfällt im Unterschied zur Schreibweise mit `[I]LIKE` die spezielle Bedeutung von % und _ als Jokerzeichen und es kann normal mit diesen Zeichen verglichen werden.

Operatoren für ganze Zahlen

Operator	Beschreibung
=	gleich
==	gleich
!=	ungleich
<>	ungleich
<	kleiner
<=	kleiner gleich
>	größer
>=	größer gleich
between(Attribut, Minimum, Maximum)	entspricht: Attribut >= Minimum and Attribut <= Maximum

Operatoren für Listen und Segmente (Channels und VChannels)

Operator	Beschreibung
inChannel(channel)	Channel abonniert
inChannelMail(channel)	Channel (per E-Mail) abonniert und eine E-Mail-Adresse angegeben
inChannelWithTag(tag)	mindestens ein Channel mit diesem Semantiktag abonniert
notInChannel(channel)	Channel nicht abonniert
notInChannelMail(channel)	Channel nicht (per E-Mail) abonniert
notInChannelWithTag(tag)	kein Channel mit diesem Semantiktag abonniert
inVChannel(vchannel)	Suchausdruck des virtuellen Channels zutreffend
notInVChannel(vchannel)	Suchausdruck des virtuellen Channels nicht zutreffend
inAnyChannel()	mindestens ein Channel abonniert
notInAnyChannel()	kein Channel abonniert

<code>removable()</code>	Gehört keiner Liste oder Segment an, die als "Verhindert Eintragslöschung" gekennzeichnet sind.
<code>notRemovable()</code>	Gehört mindestens einer Liste oder einem Segment an, die als "Verhindert Eintragslöschung" gekennzeichnet sind.

Operatoren für Datumswerte

Operator	Beschreibung
<code>inPast(Attribut, Zahl, Einheit)</code>	Timestamp liegt genau Zeitraum vor dem aktuellen Datum (z.B. alle, die genau vor 2 Tagen geändert wurden)
<code>sincePast(Attribut, Zahl, Einheit)</code>	Timestamp liegt zwischen heute und Zeitraum vor dem aktuellen Datum (z.B. alle, die in den letzten 2 Tagen geändert wurden)
<code>beforePast(Attribut, Zahl, Einheit)</code>	Timestamp liegt mindestens Zeitraum vor dem aktuellen Datum oder ist noch älter (z.B. alle, die vor 2 Tagen geändert wurden oder älter sind)
<code>between(Attribut, Minimum, Maximum)</code>	Timestamp liegt innerhalb von Minimum und Maximum (einschließlich Minimum und Maximum)

Mit diesen Operatoren für Timestamps werden Zeit- und Datumswerte mit dem aktuellen Zeitpunkt verglichen, beim Operator `inPast()` wird lediglich der Datumswert verglichen.

Der Vergleich hat den Wahrheitswert `true`, wenn der Timestamp innerhalb des angegebenen Zeitraums zum aktuellen Datum liegt. Der Zeitraum wird in Tagen, Wochen, Monaten oder Jahren vor dem aktuellen Datum angegeben und mit den Buchstaben `d` (Tage), `w` (Wochen), `m` (Monate), `y` (Jahre), `h` (Stunden), `m` (Minuten) und `s` (Sekunden) als Einheit bezeichnet.

Operator	Beschreibung
<code>dateLike(Attribut, Muster, Zahl, Einheit)</code>	Timestamp entspricht dem Muster oder liegt optional einen Zeitraum vor dem Muster (z.B. alle, die am gestrigen Tag Geburtstag haben)

Mit diesem Operator für Timestamps werden Zeit- und Datumswerte mit dem aktuellen Zeitpunkt verglichen. Der Vergleich hat den Wahrheitswert `true`, wenn der Timestamp dem angegebenen Muster entspricht oder optional einen Zeitraum vor dem angegebenen Muster liegt.

Datumswerte werden intern im Format `YYYY-MM-DD | hh:mm:ss` gespeichert. Das Muster bezieht sich auf dieses Format und kann die folgend angegebenen Bezeichner sowie die Jokerzeichen `%` für beliebig viele Zeichen und `_` für ein beliebiges Zeichen verwenden.

Bezeichner	Beschreibung
<code>y</code>	aktuelles Jahr
<code>m</code>	aktueller Monat
<code>d</code>	aktueller Tag

H	aktuelle Stunde
M	aktuelle Minute
S	aktuelle Sekunde

Mit dem Muster "____-m-d|00:00:00", angewendet auf das Attribut `birthday`, finden Sie also alle Einträge, die am aktuellen Tag Geburtstag haben. Dieses Muster ist als Standardwert definiert, so dass Sie auch mit der abgekürzten Suchanfrage `dateLike(birthday)` alle Einträge mit Geburtstag am aktuellen Tag finden. Dieses Muster kann zwar auch auf andere Attribute angewendet werden (`dateLike(<attributname>)`), entspricht in den meisten Fällen aber nicht das, wonach gesucht werden soll. Es wird immer nach dem Muster "____-m-d|00:00:00" gesucht, also einem Datum, das dem aktuellen Tag (unabhängig vom Jahr) entspricht. Wichtig ist an dieser Stelle auch, dass sich diese Abfrage nur auf Felder vom Typ `DATE` anwenden lässt, da dort die Zeitangaben mit "00:00:00" gespeichert werden.

Optional können Sie zusätzlich angeben, dass die gefundenen Einträge genau den angegebenen Zeitraum vor dem durch das Muster beschriebenen Zeitpunkt liegen sollen. Das Verhalten entspricht dabei dem `inPast()` Operator. Mit `dateLike(birthday, "____-m-d|00:00:00", 1, d)` können Sie also alle Einträge mit Geburtstag am gestrigen Tag finden.

Weitere Beispiele:

Datum/Timestamp von gestern: `dateLike(<attributname>, "y-m-d|%", 1, d)`

Datum/Timestamp von heute: `dateLike(<attributname>, "y-m-d|%")`

Jahrestag: `dateLike(<attributname>, "____-m-d|%")`

Operatoren für die Datenpflege

Operator	Beschreibung
<code>duplicateEmail()</code>	Das Abfrageprädikat <code>duplicateEmail()</code> erlaubt die Suche nach Einträgen, deren E-Mail-Adresse (im Attribut <i>email</i>) noch mindestens einem weiteren Eintrag zugewiesen ist.
<code>duplicateAttribute(<attributname>)</code>	Dieses neue Abfrageprädikat stellt die Verallgemeinerung von <code>duplicateEmail()</code> dar: es erlaubt die Suche nach doppelten Werten für beliebige Primärattribute wie z.B. Vorname, Nachname usw.

7.3 Sortierung der Ergebnisse

Die Ergebnisse sind implizit immer nach der `OID` sortiert. Die Sortierung kann durch einen Ausdruck im Format `ORDER BY [attributname] {ASC|DESC}` angepasst werden. `ASC` bezeichnet eine aufsteigende Sortierung, `DESC` eine absteigende Sortierung. Die Sortierung ist derzeit nur nach den am häufigsten benötigten vordefinierten Attributen (Primärattributen) möglich:

uid, email, mobile, cookie, salut, title, firstname, lastname, company, street, post code, city, country, phone, password, cmsbs.creation, cmsbs.lastchanged

7.4 Abfragen für Tabellenattribute

Die Abfragen für Attribute vom Typ TABLE folgen ähnlichen Prinzipien wie für normale Attribute, sind aber um einige spezielle Möglichkeiten erweitert.

Die Operatoren für Nullwerte `defined(attribute)` und `undefined(attribute)` erkennen, ob eine Tabelle angelegt ist und Zeilen enthält.

Mit der Notation `attribut->spalte` oder `attribut[]->spalte` kann ein Operator auf den Wert einer bestimmten Spalte in einer beliebigen Zeile der Tabelle angewendet werden. Durch `attribut[index]->spalte` kann ein Operator auf eine bestimmte Spalte in einer bestimmten Zeile der Tabelle angewendet werden.

Bitte beachten Sie, dass die Nummerierung der Zeilen einer Tabelle zwar immer aufsteigend, aber nicht unbedingt lückenlos fortlaufend erfolgt. Beim Löschen einer Zeile aus einer Tabelle wird deren Index entfernt, der Index der übrigen Zeilen wird aber nicht entsprechend neu gezählt.

Um in einer Abfrage mehrere Operatoren auf die selbe Zeile einer Tabelle zu beziehen, kann ein Dummy-Index verwendet werden. Der Dummy-Index gibt an, dass ein Bezug zu der selben Zeile hergestellt werden soll, auch wenn die Zeilennummer selbst nicht bekannt ist, z.B.:

```
attribut[x]->spalte1 = 'foo' || attribute[x]->spalte2 = 'bar'
```



Bei einer logischen Verknüpfung mehrerer Suchausdrücke auf Spalten einer Tabelle wird implizit angenommen, dass es sich um die selbe Zeile handelt. Der Ausdruck `attribut[]->spalte1 = 'foo' && attribute[]->spalte2 = 'bar'` findet also Einträge, bei denen 'foo' und 'bar' in `spalte1` und `spalte2` der selben Tabellenzeile vorkommen. Ist dies nicht gewünscht, muss mit unterschiedlichen Dummy-Indizes gearbeitet werden, um deutlich zu machen, dass die Spaltenwerte 'foo' und 'bar' auch in unterschiedlichen Zeilen stehen dürfen.

Sobald in einer Abfrage auf eine Tabelle Bezug genommen wird, werden nur die Einträge gefunden, bei denen die Tabelle definiert ist. Mit der Abfrage `undefined(attribute->spalte)` werden also nur die Einträge gefunden, bei denen die Tabelle mit einer leeren Spalte vorhanden ist. Um auch die Einträge ganz ohne eine definierte Tabelle zu finden, müsste zusätzlich nach `undefined(attribute)` gesucht werden.

7.4.1 Extrema

Mit den Funktionen `min` und `max` können Zeilen qualifiziert werden, die den jeweiligen Extremwert, also das Minimum oder Maximum, der angegebenen Spalte einer Tabelle enthalten. Diese Funktion ist nur für Spaltentypen mit Datums- oder Zahlenwerten möglich.

Aus allen oder durch Teilabfragen eingeschränkten Tabellenzeilen kann die Zeile, die ein Extremum in einer Spalte mit Datumswert oder Zahl enthält, gefiltert und auf weitere Eigenschaften geprüft werden. Man kann so z.B. nach Einträgen suchen, deren letzte Bestellung (nur die Zeile mit dem jüngsten Datum wird berücksichtigt) nicht älter als 7 Tage ist.

Beispiel:

```
(sincePast(bestellungen[x]->datum, 7, d) HAVING bestellungen[x]: max(datum))
```

Die hier verwendete `HAVING`-Klausel ist nicht mit gleichnamigen SQL-Konstrukt vergleichbar. Das Schlüsselwort wurde so genannt, da es auch einer zusätzlichen Qualifikation entspricht. Es wird nur benutzt, um Tabellenzeilen näher zu qualifizieren.

Wollen Sie die Zeilen schon vor der Bestimmung des Extremwerts einschränken (z.B. auf einen bestimmten Status), würde dies so aussehen:

```
(bestellungen[x]->cent > 50000 HAVING bestellungen[x]: min(datum) and status != "bezahlt")
```

Alle Einträge, die Bestellungen haben, die nicht im Status "bezahlt" sind. Von den gefundenen Bestellungen soll die älteste einen Gesamtpreis von mehr als 500 Euro haben.

Nach dem Schlüsselwort `HAVING` wird zunächst der Tabellename und der Index angegeben, um den Bezug zur selben Zeile herzustellen. Danach werden die Abfragen ohne Tabellename und Index zusammengestellt. Hierbei gilt, dass alle Eigenschaften zutreffen müssen und aus den gefundenen Zeilen das jeweils angegebene Extremum für die weitere Überprüfung verwendet wird. Vor dem `HAVING` stehen also immer die Eigenschaften, auf welche die jeweils durch `min` oder `max` qualifizierte Zeile überprüft werden soll.

Je Abfrage auf einer Tabellenzeile kann nur eine Extremwertabfrage erstellt werden. Diese Abfrage darf nur mit `AND` an die zusätzlichen Bedingungen geknüpft sein.

7.4.2 Aggregationen

Mit Aggregationen können Tabellen auf ihre Zeilenanzahl geprüft werden. Außerdem können Zahlenwerte einer bestimmten Spalte aufsummiert oder deren Durchschnittswert berechnet und gegen einen angegebenen Wert geprüft werden. Die Funktionsnamen für diese Berechnungen lauten `count`, `sum` und `avg`.

Die Aggregationen sind ähnlich den Abfragen auf Extremwerte aufgebaut. Die Spalten-Abfragen nach dem Schlüsselwort `HAVING` qualifizieren zunächst die Zeilen, die überprüft werden sollen. Vor `HAVING` werden diese dann zusammengefasst und auf einen Wert überprüft.

Beispiel:

```
(count(bestellungen[x]) > 5 HAVING bestellungen[x]: status = "bezahlt" and
sincePast(datum, 6, m))
```

Alle Einträge, die mehr als 5 Bestellungen haben, welche den Status "bezahlt" haben und nicht älter als 6 Monate sind.

Möchte man alle Einträge, die mehr als 5 Bestellungen haben, ohne weitere Einschränkungen abfragen, genügt es den Teil vor `HAVING` anzugeben, da die Zeilen der Tabelle nicht weiter qualifiziert werden müssen.

```
count(bestellungen[x]) > 5
```

Um die Summe der Bestellungen, die bezahlt sind und innerhalb der letzten 6 Monate getätigt wurden, gegen einen Wert zu prüfen, werden die Zeilen wieder mit Spalten-Abfragen nach `HAVING` qualifiziert und vor `HAVING` mit der Funktion `sum` zusammengefasst und überprüft, ob diese größer als 50000 ist.

```
(sum(bestellungen[x]->preis) > 50000 HAVING bestellungen[x]: status = "bezahlt" and
sincePast(datum, 6, m))
```

Je Abfrage auf einer Tabellenzeile kann nur eine Aggregation erstellt werden.



Die hier aufgeführten Beispiele für Extrema und Aggregationen entsprechen denen im "Handbuch zur Benutzeroberfläche". Extrema und Aggregationen können nicht kombiniert werden.

Da das Ergebnis einer Abfrage immer eine Liste von Einträgen ist, können die berechneten Spaltenaggregate nicht exportiert werden. Die Werte können immer nur mit einem konkreten Wert verglichen werden, so dass ein logischer Ausdruck entsteht.

7.5 Abfragen für Newsletter

Kontaktierte Einträge, Einträge mit Klicks, Öffnungen, Conversions, Abmeldungen und Bounces können mit Hilfe der Abfragesprache segmentiert werden.

Dafür stehen folgende Operatoren zur Verfügung:

Operatoren	Beschreibung
newsletterContacted	Einträge, die kontaktiert wurden

newsletterMailContacted	Einträge, die via E-Mail kontaktiert wurden
newsletterNotContacted	Einträge, die nicht kontaktiert werden konnten (z.B. fehlende E-Mail-Adresse)
newsletterNotMailContacted	Einträge, die nicht via E-Mail kontaktiert werden konnten (z.B. fehlende E-Mail-Adresse)
newsletterHasView	Einträge, die den Newsletter geöffnet haben
newsletterNotHasView	Einträge, die den Newsletter nicht geöffnet haben
newsletterHasClick	Einträge, die einen Link im Newsletter geklickt haben
newsletterNotHasClick	Einträge, die keinen Link im Newsletter geklickt haben
newsletterHasConversion	Einträge, die eine Conversion haben
newsletterNotHasConversion	Einträge, die keine Conversion haben
newsletterHasUnsubscription	Einträge, die sich abgemeldet haben
newsletterNotHasUnsubscription	Einträge, die sich nicht abgemeldet haben
newsletterHasBounce	Einträge, die einen Bounce erzeugt haben
newsletterNotHasBounce	Einträge, die keinen Bounce erzeugt haben



Die "Not"-Varianten beziehen sich immer auf das zum Versandzeitpunkt gültige Zielsegment, nicht auf die Gesamtzahl der Einträge.

Folgende Parameter können optional in der angegebenen Reihenfolge definiert werden:

- Newsletter, <Newsletter-ID>
- NewsletterGroup, <Newsletter-Serien-ID>
- NewsletterTag, <Newsletter-Tag>
- NewsletterGroup, <Newsletter-Serien-ID>, NewsletterTag, <Newsletter-Tag>
- After, <Timestamp (YYYY-MM-DD|hh:mm:ss) ODER Zeitraum relativ zu heute, z.B. '3w' für "innerhalb der letzten 3 Wochen">
- Before, <Timestamp (YYYY-MM-DD|hh:mm:ss) ODER Zeitraum relativ zu heute, z.B. '3d' für "mindestens vor 3 Tagen">

Der Zeitraum wird in Tagen, Wochen, Monaten oder Jahren vor dem aktuellen Datum angegeben und mit den Buchstaben *d* (Tage), *w* (Wochen), *m* (Monate), *y* (Jahre), *h* (Stunden), *m* (Minuten) und *s* (Sekunden) als Einheit bezeichnet.

Für die Auswertung von Klicks auf Links können folgende optionale Parameter am Ende angefügt werden:

- LinkTag, <Link-Tag>
- LinkName, <Link-Name>

Bei der Auswertung von Bounces kann der Typ eines Bounce eingeschränkt werden. Wird kein Bounce-Typ angegeben, werden nur Hard- und Softbounces ausgewertet.

- `BounceType`, `<Bounce-Typ>`

Der Bounce-Typ kann dabei folgende Werte haben:

Bounce-Typ	Beschreibung
-1	Alle
0	Unbekannt
1	Hardbounce
2	Softbounce
3	Autorespond
4	Delay
5	Spam
6	Abmeldung
7	Feedback-Look: Abuse
8	Feedback-Look: Fraud
9	Feedback-Look: Virus
10	Feedback-Look: Other
11	Feedback-Look: No Spam

Beispiele für Newsletterabfragen

```
newsletterHasView(NewsletterGroup, 2360613)
```

Einträge, die zwischen dem 22.05.2015 (inkl.) und 26.05.2015 (exkl.) Newsletter aus der Serie 2360613 geöffnet haben.

```
newsletterHasView(NewsletterTag, 'Presse', After, '2015-05-22|00:00:00', Before,  
'2015-05-26|00:00:00')
```

Einträge, die Newsletter, die mit dem Tag "Presse" versehen wurden, geöffnet haben.

```
newsletterHasView(NewsletterGroup, 2360613, NewsletterTag, 'Presse')
```

Einträge, die Newsletter, die zu der Serie 2360613 gehören und mit dem Tag "Presse" versehen wurden, geöffnet haben.

```
newsletterHasClick(Newslatter, 12345, LinkTag, 'Homepage')
```

Einträge, die im Newsletter mit der ID 12345 Links mit dem Tag "Homepage" geklickt haben.

```
newsletterHasClick(LinkName, 'Homepage1')
```

Einträge, die in irgendeinem Newsletter auf den Link mit dem Link-Namen "Homepage1" geklickt haben.

```
newsletterHasBounce(BounceType, 1)
```

Einträge, die einen Hardbounce hatten.

```
newsletterHasBounce(NewslatterGroup, 2360613, NewsletterTag, 'Presse', BounceType, 1)
```

Einträge, die Newsletter, die zu der Serie 2360613 gehören und mit dem Tag "Presse" versehen wurden, gebounct haben.



Um Abfragen für Newsletter nutzen zu können, muss das Lizenzfeature Newsletter-Segmentierung freigeschaltet sein.

7.6 Abfragen für Einwilligungen

Einwilligungen werden als Zeitstempel in der Datenbank gespeichert. Optional kann eine IP-Adresse zu einer Einwilligung gespeichert werden.

Funktion	Parameter	Beispiel
hasConsent	<ul style="list-style-type: none"> consentname Name der Einwilligung form_id [optional] ID der App-Instanz, in die Einwilligung aktualisiert wurde 	<pre>hasConsent("permission", "eqa57c1j-efe5-4091-8c4f-5cad6a21a6ed") hasConsent("consent_contact_email")</pre>

7.7 Abfragen für Referenzattribute

Dieses Abfragekonstrukt ermöglicht es, die Eigenschaften eines per *REFERENCE*-Attribut "verlinkten" Eintrags im Rahmen eines Sub-Querys abzufragen.

Dabei muss <ref-attr> der Name eines Attributes vom Typ REFERENCE sein, das als Primärattribut eine eigene Datenbankspalte hat und sich auf die OID des verlinkten Eintrages bezieht. Folgende Attributeigenschaften sind dabei besonders relevant:

```
ref_attribute_name.type           = REFERENCE
ref_attribute_name.isPrimary      = true
ref_attribute_name.reference.foreignKey = oid
```

<sub-query> ist ein beliebiger Abfrageausdruck, gegen den die referenzierten Einträge geprüft werden sollen.

7.7.1 Beispiel

Das folgende Beispiel bezieht sich auf das Datenmodell der Service Desk App.

```
entrytype="sd_ticket"
AND sd_status="new"
AND REFERENCED sd_assigned_user_oid MATCHES (firstname="John" AND lastname="Doe")
```

Diese Abfrage liefert alle Einträge vom Typ *sd_ticket* (also Service Desk-Tickets), die den Status *new* haben und deren Referenzattribut *sd_assigned_user_oid* auf einen Eintrag zeigt, dessen Vorname *John* und dessen Nachname *Doe* lautet.

7.7.2 Hinweise

Dieses Abfragekonstrukt wird nicht von der grafischen Segmentierung unterstützt und kann nicht für Referenzattribute innerhalb von Tabellenattributen verwendet werden.

7.8 Beispiele für Abfragen

Abfrage	Beschreibung
<code>a == 'b'</code>	Attribut a gleich Zeichenkette 'b'
<code>a == 42</code>	Attribut a gleich ganzer Zahl 42
<code>a LIKE 'Hans%'</code>	Attribut a beginnt mit Hans (findet 'Hans', 'Hansi', Hansen')
<code>email ilike '%@aol.de'</code>	E-Mail-Adresse mit der Domain 'aol.de'
<code>sincePast(cmsbs.lastchanged, 1, m)</code>	Eintrag wurde im letzten Monat geändert
<code>orders[]->quantity = 5</code>	Eintrag mit einer Zeile in der Tabelle orders, bei der in der Spalte quantity fünf angegeben ist
<code>not (firstname = 'Hans' and lastname = 'Meier')</code>	Alle Einträge, die nicht "Hans Meier" heißen
<code>not (firstname = 'Hans' or firstnme = 'Otto')</code>	Alle Einträge die weder "Hans" noch "Otto" heißen

8 Bounce Management

Auf eine Newsletter-Aussendung werden Sie zahlreiche Rückläufer (Bounces) an die Absenderadresse des Newsletters erhalten. Diese E-Mails sind zum größten Teil automatisch generierte Meldungen, die ebenfalls automatisch verarbeitet werden sollten. Diese Aufgabe übernimmt das Bounce Management des Universal Messenger.

Es wird empfohlen, keine Reply-To Adresse anzugeben und alle E-Mails durch den Universal Messenger analysieren zu lassen, da die Angabe einer Reply-To Adresse von einigen Spam-Filtern negativ bewertet wird. Falls der Universal Messenger eine E-Mail nicht in eine der im folgenden beschriebenen Kategorien einordnen kann, wird diese an den Postmaster weitergeleitet.

Die Analyse und Klassifizierung eingehender E-Mails durch den Universal Messenger ermöglicht eine automatische Erkennung falscher bzw. nicht mehr aktiver E-Mail-Adressen und gegebenenfalls das Entfernen des zugehörigen Eintrag aus den Listen des Universal Messenger. Über die Definition von Regeln, basierend auf regulären Ausdrücken, kann der Administrator den Analyseprozess steuern.

Ein Großteil der eingehenden E-Mails kann unerwünschte Werbung und andere Spam-Mails umfassen. Es ist deswegen sinnvoll, zusätzlich einen Spamfilter einzusetzen, der alle eingehenden E-Mails beurteilt und den Spam kennzeichnet. Der Universal Messenger kann die als Spam gekennzeichneten E-Mails aussortieren.

Bei den eingehenden E-Mails wird zwischen Spam, Hardbounces, Softbounces, Auto-Respondmessages, Unsubscribes und sonstigen E-Mails unterschieden:

- **Spam** sind unerwünschte E-Mails, die von einem Spamfilter gekennzeichnet und von dem Universal Messenger aussortiert werden können.
- **Hardbounces** sind zurückkommende E-Mails, die nicht an den Empfänger zugestellt werden konnten, weil die E-Mail-Adresse nicht existiert.
- **Softbounces** sind zurückkommende E-Mails, die zeitweise nicht an den Empfänger zugestellt werden konnten, weil die Mailbox z.B. überfüllt ist.
- **Auto-Respondmessages** sind E-Mails, die vom Empfänger automatisch als Eingangsbestätigung, Urlaubsmeldung oder sonstigen Hinweis verschickt werden.
- **Unsubscribes** sind E-Mails des Empfängers mit der Bitte um Abmeldung vom Newsletter, die das Schlüsselwort "unsubscribe" enthalten.
- Alle anderen eingehenden E-Mails konnten nicht automatisch erkannt werden oder sind persönliche E-Mails, die an den Postmaster weitergeleitet werden.

Wenn ein Bounce als Hardbounce, Softbounce oder Auto-Respondmessage erkannt wird, werden entsprechende Zähler des betroffenen Eintrags erhöht. Sobald diese Zähler einen einstellbaren Grenzwert überschreiten, wird dieser Eintrag standardmäßig aus allen Listen des Universal Messenger entfernt (nicht gelöscht) bzw. der vordefinierte CSE-Callback ausgeführt. In aktuellen Versionen des Universal Messenger ist der Grenzwert für diese Zähler in einer Standardinstallation auf -1 gesetzt, so dass diese Funktion per default deaktiviert ist.

Die Bearbeitung und Klassifizierung von eingehenden E-Mails geschieht in einem mehrstufigen Prozess. Für alle E-Mails werden die fünf Regeldateien in der Reihenfolge Spam, Hardbounces, Softbounces, Auto-Respondmessages und Unsubscribes durchlaufen, die eine E-Mail in die erste passende Kategorie einstufen. Außer bei dem sofort aussortierten Spam wird im nächsten Schritt der Absender der E-Mail ermittelt. Stimmt der Absender mit einer in dem Universal Messenger eingetragenen E-Mail-Adresse überein, wird der entsprechende Zähler des Eintrags erhöht. Handelt es sich um einen Unsubscribe oder wird der eingestellten Maximalwert des Zählers überschritten, wird der Eintrag ggf. gelöscht. Dies muss allerdings explizit per CSE-Callback definiert werden, standardmäßig wird der Eintrag nur aus den Listen entfernt.

Hinweis: Wenn sich die E-Mail-Adresse eines Eintrags im Attribut `email` ändert, werden die Bounce-Zähler `cmsbs.softbounces`, `cmsbs.hardbounces` und `cmsbs.autoresponds` automatisch gelöscht und auf Null zurückgesetzt.

Kann der Absender der E-Mail keinem Eintrag in dem Universal Messenger zugeordnet werden, wird die eMail in den Ordner Error der IMAP-Mailbox verschoben.

Die Analyse und Klassifizierung der eingehenden E-Mails wird von dem Universal Messenger in der in der Konfigurationsoption `cmsbs.bounce.logfile` definierten Logdatei protokolliert. Besonders in der Einführungsphase des Universal Messenger sollten Sie regelmäßig kontrollieren, ob die E-Mails korrekt erkannt werden. Bei einer falschen Konfiguration könnten persönliche E-Mails fälschlicherweise als Auto-Respondmessages eingeordnet und damit ignoriert werden.

Zuordnung zu einem Newsletter

Unter Umständen kann ein Bounce auch ohne VERP dem vorher versendeten Newsletter zugeordnet werden. Hierbei versucht der Universal Messenger die beim Versand erstellte Message-ID des Newsletters im Bounce wiederzuerkennen, die jedoch nur teilweise in den Bounces enthalten ist. Wird die Message-ID nicht erkannt, werden die Bounces bzw. Responds nur beim Eintrag selbst vermerkt.

8.1 Konfiguration des Bounce Management

```
cmsbs.startbounce = true|false
```

`true` startet die Bounceverarbeitung, `false` verhindert den Start. Die Bounceverarbeitung erfolgt über einen Prozess, der in regelmäßigen Abständen das IMAP-Postfach `cmsbs.mail.imap.user` nach eingehenden Bounces abfragt.

```
cmsbs.mail.imap.server = <Hostname oder IP-Adresse>
```

Der IMAP-Server, über den E-Mails für die Bounceerkennung empfangen werden sollen.

```
cmsbs.mail.imap.port = <IP-Port>
```

Optionale Angabe des Ports des IMAP-Servers, über den E-Mails für die Bounceerkennung empfangen werden sollen. Falls kein Port angegeben ist, wird der Standardport 143 verwendet.

Wenn der IMAP-Server per SSL abgefragt werden soll, muss hier in der Regel der Port 993 angegeben werden. Zusätzlich muss die Option `cmsbs.mail.imap.ssl` gesetzt werden.

```
cmsbs.mail.imap.user = <Benutzername>
```

Der IMAP-Benutzer, über den E-Mails für die Bounceerkennung empfangen werden sollen.

```
cmsbs.mail.imap.password = <Passwort>
```

Das Passwort des IMAP-Benutzers, über den E-Mails für die Bounceerkennung empfangen werden sollen.

```
cmsbs.mail.imap.ssl = true | tls | false
```

Protokollauswahl: `true` steht für SSL, `tls` für TLS/STARTTLS, `false` für unverschlüsselte Verbindung.

```
cmsbs.mail.imap.folder.inbox = <Ordnername>
```

Name der Inbox des IMAP-Servers. Als Standard ist "INBOX" definiert, was nur in seltenen Fällen geändert werden muss.

```
cmsbs.mail.imap.folder.debug = <Ordnername>
```

Name des Debug-Ordners zur Ablage der bearbeiteten Bounces relativ zur Inbox. Soll der Debug-Ordner innerhalb der Inbox angelegt werden, so ist die Einstellung "INBOX.debug" zu verwenden. Das ist dann notwendig, wenn der IMAP-Server das Anlegen neuer Ordner nur innerhalb der Inbox erlaubt.

Es kann eine beliebige Hierarchietiefe angegeben werden, der Punkt dient zur Trennung der Ebenen, notwendige Ordner werden von dem Universal Messenger automatisch angelegt.

```
cmsbs.mail.imap.folder.hardbounce = <Ordnername>
```

```
cmsbs.mail.imap.folder.softbounce = <Ordnername>
```

```
cmsbs.mail.imap.folder.autorespond = <Ordnername>
```

```
cmsbs.mail.imap.folder.unsubscribe = <Ordnername>
```

```
cmsbs.mail.imap.folder.spam = <Ordnername>
```


Sollen unterschiedliche Bounces beim Debugging in unterschiedlichen Ordnern abgelegt werden, können Sie die oben genannten Konfigurationseinstellungen ändern. Standardmäßig werden sämtliche Bounces im Debug-Ordner abgelegt.

```
cmsbs.bounce.debug = true|false
```

Bei `true` werden Nachrichten nach der Bearbeitung durch die Bounce-Erkennung in den mit der Konfigurationsoption `cmsbs.mail.imap.folder.debug` angegebenen Ordner in der Inbox verschoben, weiterhin erscheinen einige zusätzliche Ausgaben in der Logdatei.

```
cmsbs.mail.imap.folder.error = <Ordnername>
```

Name des Error-Ordners zur Ablage nicht erkannter Bounces relativ zur Inbox. Soll der Error-Ordner innerhalb der Inbox angelegt werden, so ist die Einstellung "INBOX.error" zu verwenden. Das ist dann notwendig, wenn der IMAP-Server das Anlegen neuer Ordner nur innerhalb der Inbox erlaubt.

Es kann eine beliebige Hierarchietiefe angegeben werden, der Punkt dient zur Trennung der Ebenen, notwendige Ordner werden von dem Universal Messenger automatisch angelegt.

```
cmsbs.bounce.postmaster = <eMail-Adresse>
```

Die E-Mail-Adresse, an die vom Bounce-Management nicht erkannte E-Mails zur manuellen Bearbeitung weitergeleitet werden sollen. Alle nicht erkannten E-Mails verbleiben im Folder "error".

```
cmsbs.bounce.subject = <Text>
```

Dieser Text wird vor das Subject jeder an den Postmaster weitergeleiteten E-Mail gehängt, damit sie leicht zugeordnet werden kann.

```
cmsbs.bounce.spam = <Dateiname mit Pfad>
```

Die Regeldatei für die Bounce-Erkennung von Spam.

```
cmsbs.bounce.hardbounce = <Dateiname mit Pfad>
```

Die Regeldatei für die Bounce-Erkennung von Hardbounces.

```
cmsbs.bounce.hardbounce.max = <Zahl>
```

Bei Erreichen dieser Anzahl von Hardbounces wird der Empfänger-Eintrag automatisch von allen Listen abgemeldet (nicht gelöscht), bzw. der vordefinierte Callback wird ausgeführt. In aktuellen Versionen des Universal Messenger ist der Grenzwert für diesen Zähler in einer Standardinstallation auf -1 gesetzt, so dass diese Funktion per default deaktiviert ist.

```
cmsbs.bounce.softbounce = <Dateiname mit Pfad>
```

Die Regeldatei für die Bounce-Erkennung von Softbounces.

```
cmsbs.bounce.softbounce.max = <Zahl>
```

Bei Erreichen dieser Anzahl von Softbounces wird der Empfänger-Eintrag automatisch von allen Listen abgemeldet (nicht gelöscht), bzw. der vordefinierte Callback wird ausgeführt. In aktuellen Versionen des Universal Messenger ist der Grenzwert für diesen Zähler auf -1 in einer Standardinstallation gesetzt, so dass diese Funktion per default deaktiviert ist.

```
cmsbs.bounce.autorespond = <Dateiname mit Pfad>
```

Die Regeldatei für die Bounce-Erkennung von Auto-Respondmessages.

```
cmsbs.bounce.autorespond.max = <Zahl>
```

Bei Erreichen dieser Anzahl von Autoresponder-Nachrichten wird der Empfänger-Eintrag automatisch von allen Listen abgemeldet (nicht gelöscht), bzw. der vordefinierte Callback wird ausgeführt. In aktuellen Versionen des Universal Messenger ist der Grenzwert für diesen Zähler in einer Standardinstallation auf -1 gesetzt, so dass diese Funktion per default deaktiviert ist.

```
cmsbs.bounce.logfile = <Dateiname mit Pfad>
```

Dateiname inklusive Pfad der Logdatei, in der die Bounce-Erkennung protokolliert werden soll.

```
cmsbs.bounce.logfile.rotate = true|false
```

Bei `true` wird jeweils nächtlich die Logdatei rotiert und die vorhergehende Version mit dem Datum im Dateinamen archiviert. Bei `false` werden neue Einträge an eine bestehende Logdatei angehängt.

```
cmsbs.bounce.processAll = true|false
```

Bei `true` werden alle Nachrichten im IMAP-Postfach zur Bounce-Erkennung abgerufen. Bei `false` werden nur die ungelesenen, als neu markierten Nachrichten bearbeitet, die übrigen Nachrichten verbleiben in der Inbox.

```
cmsbs.bounce.userquery = <Abfrage>
```

Diese Option ist hilfreich, wenn ein Bounce nur via E-Mail-Adresse zugeordnet werden kann (d.h. weder VERP wird benutzt noch kann eine Message ID im Bounce gefunden werden), diese aber kein eindeutiges (unique) Attribut im Universal Messenger ist. Mit der angegebenen Abfrage kann die Ergebnismenge der gleichen E-Mail-Adressen noch einmal reduziert werden und ein Bounce kann so bei eindeutigen E-Mail-Adressen je Mandant eindeutig zugeordnet werden.

8.1.1 IMAP mit OAUTH2

Ab der Universal Messenger Release 7.49.0 kann die Authentifizierung beim Auslesen von IMAP-Postfächern mit OAUTH2 erfolgen.

Folgende Einstellungen müssen dazu vorgenommen werden:

```
cmsbs.mail.imap.server          = outlook.office365.com
cmsbs.mail.imap.port            = 993
cmsbs.mail.imap.ssl             = true
cmsbs.mail.imap.user            = <mailbox_name>
cmsbs.mail.imap.password        = " "
cmsbs.mail.imap.authenticationMethod = oauth2
cmsbs.mail.imap.oauth2.accessTokenUri =
https://login.microsoftonline.com/<tenant_id>/oauth2/v2.0/token
cmsbs.mail.imap.oauth2.clientId  = <client_id>
cmsbs.mail.imap.oauth2.clientSecret = <client_secret>
cmsbs.mail.imap.oauth2.scope     = https://outlook.office.com/.default
cmsbs.mail.imap.oauth2.grantType = client_credentials
cmsbs.mail.imap.oauth2.debug     = true
```

Das Beispiel bezieht sich auf Microsoft Office 365. Bei anderen Anbietern müssen die Parameter entsprechend den jeweiligen Vorgaben gesetzt werden.

Die gesamte Einrichtung für Microsoft Office 365 wird hier Schritt für Schritt beschrieben: [Exchange 365: IMAP with OAUTH2](#)

Wenn mehr als ein IMAP-Postfach ausgelesen werden soll, werden alle weiteren Postfächer nach folgendem Schema konfiguriert:

```

cmsbs.bounce.listeners = inbox_2 inbox_3

cmsbs.bouncelistener.inbox_2.mail.imap.server      = outlook.office365.com
cmsbs.bouncelistener.inbox_2.mail.imap.port       = 993
cmsbs.bouncelistener.inbox_2.mail.imap.ssl        = true
cmsbs.bouncelistener.inbox_2.mail.imap.user       = <mailbox_name>
cmsbs.bouncelistener.inbox_2.mail.imap.password   = ""
cmsbs.bouncelistener.inbox_2.authenticationMethod = oauth2
cmsbs.bouncelistener.inbox_2.oauth2.accessTokenUri =
https://login.microsoftonline.com/<tenant_id>/oauth2/v2.0/token
cmsbs.bouncelistener.inbox_2.oauth2.clientId      = <client_id>
cmsbs.bouncelistener.inbox_2.oauth2.clientSecret   = <client_secret>
cmsbs.bouncelistener.inbox_2.oauth2.scope         = https://outlook.office.com/.default
cmsbs.bouncelistener.inbox_2.oauth2.grantType      = client_credentials
cmsbs.bouncelistener.inbox_2.oauth2.debug         = true

cmsbs.bouncelistener.inbox_3.mail.imap.server      = outlook.office365.com
cmsbs.bouncelistener.inbox_3.mail.imap.port       = 993
...

```

8.2 Organisation des Regelwerks

Die Regeln für das Bounce Management werden in Dateien gespeichert.

Regelwerk	Beschreibung
spam.keywords	Spam-Nachrichten
hardbounce.keywords	Hardbounces
softbounce.keywords	Softbounces
autorespond.keywords	Autorespond-Nachrichten
identify.keywords	
unsubscribe.keywords	Unsubscribe-Nachrichten

8.2.1 Auslieferung

Jedes Release des Universal Messenger enthält ein Regelwerk, das als Ausgangspunkt genutzt werden kann.

Die Regeldateien werden im Verzeichnis <UM_HOME>/cmsbs-conf ausgeliefert.

Beachten Sie, dass ein Release diese Dateien ersetzen kann.

8.2.2 Kundenspezifische Anpassungen

Im Verzeichnis `<UM_HOME>/conf/conf.d` können unter den gleichen Dateinamen eigene Regelwerke abgelegt werden. Ist ein kundenspezifisches Regelwerk vorhanden, wird dieses ausschließlich für das Bounce Management eingesetzt.

8.3 Aufbau der Regeln

Die fünf Regeldateien für Spam, Hardbounces, Softbounces, Auto-Respondmessages und Unsubscribes haben jeweils den gleichen Aufbau:

- Kommentare beginnen mit `'#'`.
- Eine Regel hat einen eindeutigen Namen.
- Regeln haben das Format (siehe [Reguläre Ausdrücke](#)) `<name>.<feld> = <regulärer Ausdruck>`
- `feld` ist entweder der Name eines der Mailheader oder `body` für den Nachrichtentext.
- Eine Regel kann aus mehreren Ausdrücken bestehen.
- Groß-/Kleinschreibung findet keine Beachtung.
- Bei Regeln, die aus mehreren Ausdrücken bestehen, müssen alle Ausdrücke zutreffen.
- Einige Sonderzeichen haben in regulären Ausdrücken eine spezielle Funktion, so dass sie für eine normale Verwendung mit einem doppelten Backslash gequotet werden müssen.
- Die Regeln werden in der Reihenfolge angewendet, in der sie in der Regeldatei aufgeführt sind.

Regeln mit Namen `extract` sind keine Erkennungsregeln, sondern dienen zum Extrahieren der E-Mail-Adresse. Das Format ist hier `extract.<name> = <regulärer Ausdruck>`. Bei der Auto-Respondmessages und Unsubscribe-Erkennung werden diese immer auf das `from`-Feld der Nachricht angewandt, bei Hardbounces und Softbounces immer auf den Text der Nachricht.

Das folgende Beispiel zeigt eine Regel namens `example`, die einen Bounce erkennt, wenn das `from`-Feld mit dem Text `MAILER-DAEMON@` beginnt, das Thema der Nachricht das Wort `failed` enthält und der Text der Nachricht das Wort `bounce`:

```
example.from = "MAILER-DAEMON@.*"  
example.subject = ".*failed.*"  
example.subject = ".*failed.*"  
example.body = ".*bounce.*"
```

Weitere Beispiele befinden sich in den mitgelieferten Beispieldateien `hardbounce.keywords`, `spam.keywords`, `softbounce.keywords`, `autorespond.keywords` und `unsubscribe.keywords` im Konfigurationsverzeichnis des Universal Messenger. Der Name und Pfad der Regeldateien kann über Konfigurationsoptionen geändert werden.

8.4 Spam

Von einem dem Universal Messenger vorgeschalteten Spamfilter können eingehende E-Mails als Spam gekennzeichnet werden. In der Regel fügen die Spamfilter den E-Mails ein zusätzliches Header-Element hinzu, das z.B. X-Spam-Flag heißt. Dieses Element kann über folgende Regel in `spam.keywords` ausgewertet werden:

```
spam.X-Spam-Flag= "yes"
```

Als Spam erkannte E-Mail wird sofort gelöscht. Über weitere Regeln könnten Sie unerwünschte Fehlermeldungen über den Betreff "Virus alert" einer E-Mail aussortieren.

8.5 Hardbounces

In der Regel kann bei Hardbounces unterschieden werden zwischen den E-Mails, die schon vom eigenen Mailer nicht ausgeliefert werden konnten (ungültiges Adressformat, ungültige Domain, Empfänger-Mailserver verweigert die Annahme wegen unbekanntem Empfänger oder anderem Fehler) und den E-Mails, die zwar zunächst ausgeliefert wurden, dann aber vom Empfänger-Mailserver zurückgesendet wurden.

Im ersten Fall trifft die erste Regel aus `hardbounce.keywords` zu, in der der Absender und der übliche Fehlertext des eigenen Mailservers eingetragen wird:

```
own_smtp.from = "MAILER-DAEMON.*"  
own_smtp.subject = "DELIVERY FAILURE*"
```

Im zweiten Fall kann der Absender und die Fehlermeldung je nach Empfänger-Mailserver verschieden sein, so dass eine Reihe häufig vorkommender Fehlermeldungen konfiguriert werden müssen.

```
typ1_smtp.subject = "Returned mail: User unknown"  
typ2_smtp.subject = "^Undeliverable.*"  
typ3_smtp.subject = "^Mail delivery failed.*"
```

In den folgenden Zeile der Beispielkonfiguration sind die Regeln zum Extrahieren der E-Mail-Adresse aufgeführt. Es wird versucht, die E-Mail-Adresse innerhalb einer Fehlermeldung zu finden, damit nicht eine andere, zufällig in der E-Mail enthaltene E-Mail-Adresse als Absenderadresse interpretiert wird.

8.6 Auto-Respondmessages

Auto-Respondmessages können als Absenderadresse die E-Mail-Adresse des ursprünglichen Empfängers enthalten und im zusätzlichen Header-Feld Sender den Auto-Responder eintragen, z.B.

autoresponder@gmx.de. Bei anderen Anbietern ist der Auto-Responder in der Absenderadresse der E-Mail eingetragen und die E-Mail-Adresse des ursprünglichen Empfängers im zusätzlichen Header-Feld Reply-To zu finden.

Falls der Auto-Responder in der Absenderadresse der E-Mail eingetragen ist, können diese E-Mails nicht automatisch ausgewertet werden, da die E-Mail-Adresse des ursprünglichen Empfängers nicht bestimmt werden kann. Die E-Mail-Adresse muss aber als zusätzliches Prüfkriterium mit dem Eintrag verglichen werden können.

Im ersten Fall der E-Mail-Adresse eines Eintrags wird mit den Regeln aus `autorespond.keywords` geprüft, ob es sich um eine Auto-Respondmessage handelt.

```
gmx.sender = "autoresponder@gmx.de (auto-responder)"
gmx.body = ".*Urlaub.*"
```

In den folgenden Zeile der Beispielkonfiguration sind die Regeln zum Extrahieren der E-Mail-Adresse Absenderadresse aufgeführt, die nicht geändert werden sollten.

8.7 Unsubscribes

Unsubscribes müssen zunächst als Absenderadresse die E-Mail-Adresse eines Eintrags enthalten. Falls es eine Übereinstimmung gibt, wird mit den Regeln aus `unsubscribe.keywords` geprüft, ob es sich um einen Unsubscribe handelt.

Der Betreff und der Inhalt der E-Mail werden nach Schlüsselworten durchsucht, die allgemein zur Abmeldung eines Newsletters üblich sind. Sie sollten diese Regeln so anpassen, dass eine E-Mail nicht fälschlicherweise als Unsubscribe interpretiert wird, weil sie eines dieser Schlüsselworte enthält.

1. `subject = ".*unsubscribe.*"`
2. `subject = ".*remove.*"`
3. `body = "^[]*unsubscribe[]*"`
4. `body = "^[]*remove[]*"`

In den Zeilen der Beispielkonfiguration sind die Regeln zum Extrahieren der E-Mail-Adresse Absenderadresse aufgeführt, die nicht geändert werden sollten.

```
cmsbs.unsubscribe.keywords = <Dateiname mit Pfad>
```

Die Regeldatei für die Bounce-Erkennung von Abmeldungen per E-Mail (Unsubscribes).

```
cmsbs.unsubscribe.reason = ""
```

Wenn der Bounce-Listener eine Unsubscribe-Nachricht empfängt, wird der angegebene interne Werte für die `reason`-Spalte in der Tabelle `chaction` bzw. `newsletter_channel_activity` verwendet, um den Empfänger von allen Channels abzumelden.



Der Default-Wert `cmsbs.unsubscribe.reason = ""` schaltet das Verhalten komplett ab, so dass Unsubscribe-E-Mails keine Wirkung haben.

8.8 Mehrere Bouncelistener

Der Universal Messenger kann beliebig viele IMAP-Postfächer abfragen, um Bounces zu empfangen und auszuwerten.

Die Konfiguration weiterer IMAP-Postfächer zusätzlich zum primären Postfach erfolgt in der `cmsbs.properties`-Datei. Die Namen aller zusätzlichen Postfächer müssen aufgelistet werden:

```
cmsbs.bounce.listeners = "mandantA mandantB"
```

Die weiteren Einstellungen pro Postfach gleichen denen für das primäre Postfach, jedoch muss jeweils immer der Name des Postfachs in den Variablennamen in der Konfigurationsdatei eingesetzt werden:

```
cmsbs.bouncelistener.mandantA.mail.imap.server = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.port = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.user = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.password = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.ssl = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.protocol = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.folder.inbox = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.folder.error = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.folder.hardbounce = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.folder.softbounce = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.folder.autorespond = "..."  
cmsbs.bouncelistener.mandantA.mail.imap.folder.unsubscribe = "..."  
cmsbs.bouncelistener.mandantA.postmaster = "..."  
cmsbs.bouncelistener.mandantA.subject = "..."  
cmsbs.bouncelistener.mandantA.logfile = "..."  
cmsbs.bouncelistener.mandantA.userquery = "..."
```


Bis auf `.user`, `.password`, `.server`, `.userquery` und `.logfile` wird für alle Konfigurationsvariablen jeweils die Einstellung des primären Postfachs übernommen, falls sie nicht explizit überschrieben wird. Für alle IMAP-Postfächer werden die zentralen Regeldateien zur Analyse der Rückläufer verwendet. Es ist nicht möglich, pro IMAP-Postfach andere Regeldateien zuzuweisen.

Empfängerzuordnung bei Mandantentrennung

Falls der ursprüngliche Empfänger weder per VERP noch per Message-ID (siehe Reihenfolge der Bouncelistener) ermittelt werden kann, versucht der Universal Messenger, die Zuordnung über die E-Mail-Adresse herzustellen. In diesem Fall kann für jeden Bouncelistener ein eigener Abfrageausdruck angegeben werden, der beispielsweise die Zuordnung zu einem logischen Mandanten sicherstellt.

```
cmsbs.bouncelistener.mandantA.userquery = "tenant='a' "  
cmsbs.bouncelistener.mandantB.userquery = "tenant='b' "
```

8.8.1 Reihenfolge der Bouncelistener

Der Universal Messenger verfügt über vier unterschiedliche Bouncehandler, die beim Empfang einer E-Mail aktiv werden können. Der erste Bouncehandler in dieser Liste, der eine E-Mail akzeptiert, konsumiert die E-Mail und beendet damit die Aufrufkette. Die Handler werden dabei immer allgemeiner, bis der Standard-Bouncehandler nur noch Heuristiken (Keyword-Dateien) verwenden kann, um Art und Absender eines Bounces zu identifizieren.

1. CSE-Bouncehandler, sofern CSE verfügbar ist (Informationen zu CSE-Bouncehandlern finden sie in der CSE-API Dokumentation)
2. Feedback-Report-Handler, sondern konfiguriert (siehe [Feedback-Loops](#))
3. VERP-Bouncehandler, sofern VERP konfiguriert ist
4. MessageID-Bouncehandler
5. Standard-Bouncehandler

8.9 Bounce Management per VERP

8.9.1 Bounce Management per VERP

Das E-Mail Protokoll SMTP hat die Besonderheit, dass zwischen den Envelope-From- und Envelope-To-Adressen und den From- und To-Adressen im Header der E-Mail unterschieden wird. Das ist vergleichbar mit der Zustellung eines Briefs, bei dem die Angaben auf dem Umschlag (Envelope) für die Zustellung entscheidend sind. Die im Brief angegebenen Adressen sind für die Zustellung hingegen ohne Bedeutung.

Für die Verarbeitung (Versand und Zustellung) der E-Mails sind die Envelope-Adressen entscheidend, beim Lesen der E-Mails werden aber im Normalfall nur die Header-Adressen angezeigt. Rückläufer werden immer an die Envelope-From-Adresse gesendet.

Bei dem regulären Bounce-Management werden Ursache des Rückläufers und der betroffene Abonnent per Textanalyse ermittelt, was einen Ungenauigkeitsfaktor bedeuten kann. Es ist zudem nicht möglich, den Newsletter zuzuordnen, der den Rückläufer ausgelöst hat. Um diese Probleme zu umgehen, kann das Bounce-Management durch VERP (variable envelope return path) erweitert werden. VERP wird von vielen Mail-Servern, aber nicht von Domino und Exchange, unterstützt.

Beim Einsatz von VERP wird jeder Newsletter für jeden Abonnenten mit einer anderen individuellen Envelope-From-Adresse versendet. Diese Adresse wird während des Versands vom Universal Messenger generiert. Eine nach diesem Verfahren versendete E-Mail wird deswegen als per VERP (variable envelope return path) versendet bezeichnet, da jede E-Mail eine variable Envelope-From Adresse als Absender erhält. Da Rückläufer immer an die Envelope-From-Adresse versendet werden und diese pro Abonnent eindeutig ist, kann der Universal Messenger exakt ermitteln, auf welchen Abonnenten sich der Rückläufer bezieht.



Wird kein VERP verwendet, setzt der Universal Messenger für Envelope-From und Header-From identische Werte ein, die dem in der XML-Event-Datei angegebenen "sender" entsprechen.

8.9.2 Konfiguration von VERP


Um VERP für den Versand zu aktivieren, muss in der Konfigurationsdatei `cmsbs-conf/cmsbs.properties` die Konfigurationsvariable `cmsbs.bounce.verp.envelopeFrom` auf die gewünschte `envelopeFrom`-Adresse gesetzt werden. Der Platzhalter `"%0"` wird jeweils durch die individuell pro E-Mail generierte ID ersetzt und bildet damit die individuelle Envelope-From-Adresse:

```
cmsbs.bounce.verp.envelopeFrom = "um-%0@SERVER.de"
```

Beispiele für daraus gebildete `envelopeFrom`-Adressen:

```
um-509T.50IM.3D82A15AF694CFF628620DE75A3907AE@SERVER.de
um-509R.50HP.1A9C67757FC1D18B7861F143F69A377F@SERVER.de
```

Eine spezielle Konfiguration des Bounce Managements ist nicht notwendig. Bei der Bearbeitung der Bounces wird vom Universal Messenger zunächst geprüft, ob die To: Adresse dem definierten VERP-Schema entspricht und dementsprechend werden die passenden Regeln zur Verarbeitung der Bounces aktiviert.

 Wird VERP verwendet, so setzt der Universal Messenger für Envelope-From das beschriebene VERP-Schema ein. Für Header-From wird der XML-Event-Datei angegebene "sender" verwendet.

Bei eingehenden Bounces durchsuchen die Bounce-Listener außerdem Felder im Header der Bounce-E-Mail nach einem gültigen VERP DeliveryTicket, um den Bounce zuordnen zu können. Die Namen der E-Mail-Header können pro Listener konfiguriert werden. Mit dieser Ergänzung kann ein Bounce-Management per VERP mit noch mehr SMTP/IMAP-Server-Kombinationen realisiert werden.

```
# Default
cmsbs.bounce.verp.envelopeToHeaders
= "Envelope-To X-Original-To Original-Recipient Original-Recipients
X-Original-Recipient X-Original-Recipients"
# Pro Bounce-Listener
cmsbs.bouncelister.[NAME].verp.envelopeToHeaders = "..."
```

Nicht automatisch klassifizierbare Bounces, die an den Postmaster weitergeleitet werden, erscheinen beim Postmaster mit den Absenderangaben des Universal Messenger.

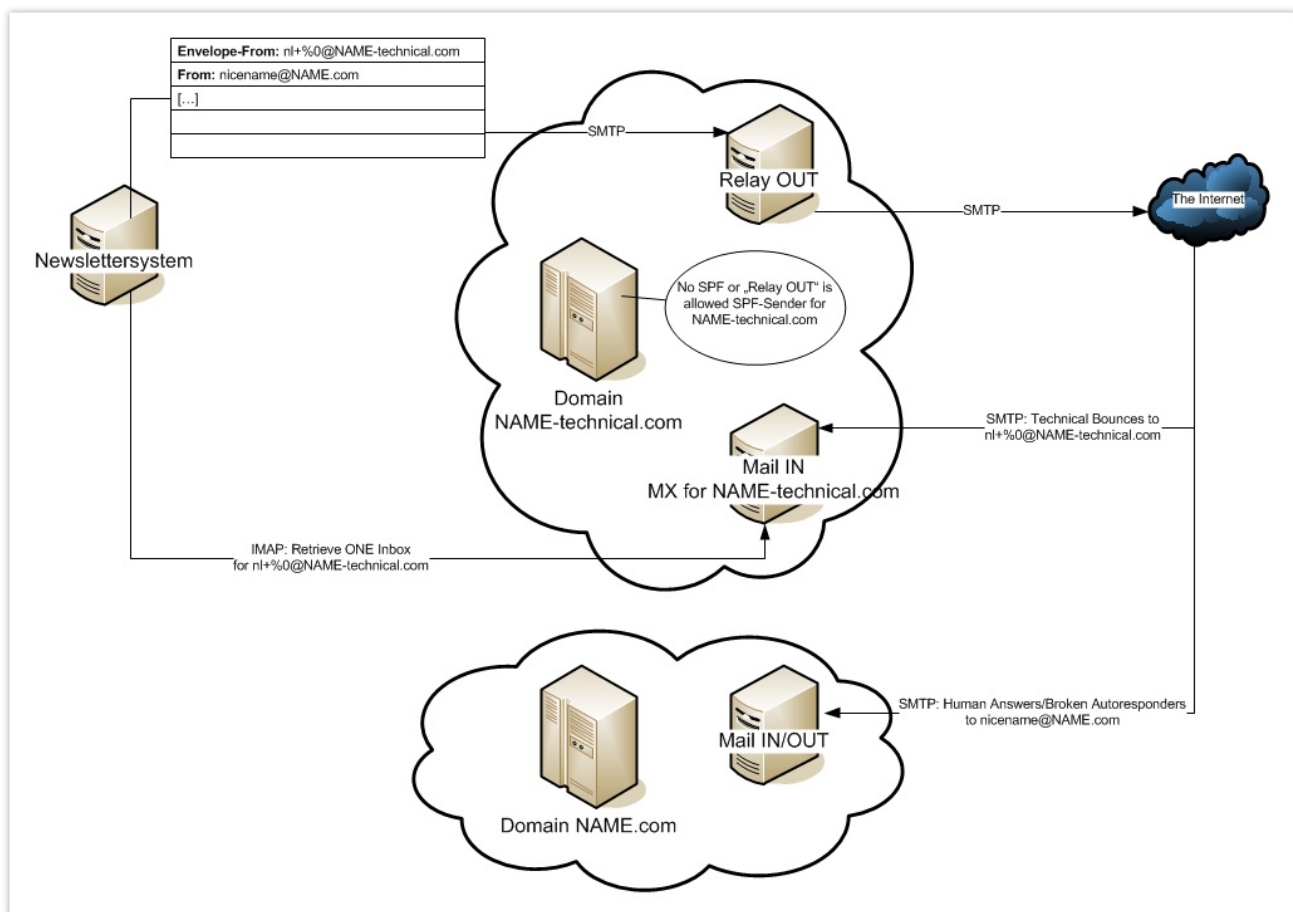
Um die originalen Daten aus dem ursprünglichen Header nicht zu verlieren, werden diese Daten wie folgt in den Header der Mail eingefügt:

```
X-UM-Original-Recipient - "To" des eingegangenen Bounces
X-UM-Original-Envelope-to - "Envelope-to" des eingegangenen Bounces
X-UM-Original-Return-Path - "Return-Path" des eingegangenen Bounces
X-UM-Original-From - "From" des eingegangenen Bounces
Return-path: <> (verhindert erneute Bounces)
From: Bounce Handler <$(cmsbs.properties: cmsbs.mail.senderaddress)>
```

8.9.3 Einrichtung der beim Newsletterversand benötigten E-Mail-Adressen

Ein Newsletter wird mit einer Absenderadresse, ohne ReplyTo-Angabe, aber mit einer variablen Envelope-From-Adresse versendet. Damit ist eine Unterscheidung zwischen einer redaktionell betreuten Adresse (die Absenderadresse) und einer technischen Adresse (die Envelope-From-Adresse) möglich:

- Rückläufer auf den Newsletter werden der technischen Adresse zugestellt.
- Auto-Respondmessages werden der technischen oder betreuten Adresse zugestellt (dies ist von Mailer zu Mailer unterschiedlich).
- Persönliche Antworten werden der betreuten Adresse zugestellt.



Das technische Postfach wird vom Universal Messenger per IMAP eingelesen und die E-Mails werden analysiert. Durch den variablen Bestandteil der Envelope-From-Adresse ist eine exakte Zuordnung zwischen dem Rückläufer und dem Newsletter möglich, sodass die Newsletterstatistik um die Bounce-Meldungen erhöht wird.

Nicht erkannte E-Mails werden an die betreute E-Mail-Adresse weitergeleitet.

Es werden somit folgende E-Mail-Adressen benötigt:

- Absender-Adresse des Newsletters, die persönliche Antworten enthält und deshalb redaktionell betreut werden sollte.
- Technische Adresse, die Bounces entgegen nimmt und vom Newslettersystem per IMAP eingelesen wird.
- Postmaster-Adresse, die vom Universal Messenger nicht erkannte Bounces als Weiterleitung erhält. Hierbei handelt es sich primär um Bounces, aber auch persönliche Antworten/Autorespond-Nachrichten können enthalten sein. Diese Adresse sollte deshalb administrativ betreut werden, so dass eine Anpassung der Bounce-Regeln oder eine Weiterleitung an Redakteure erfolgen kann.

8.9.4 Konfiguration von VERP im IMAP-Server

Die Einrichtung des IMAP-Postfachs für die Envelope-From Adresse erfordert bei der Nutzung von VERP eine gesonderte Konfiguration, da alle an die individuellen Envelope-From Adressen eingehenden Bounces auf ein Sammelpostfach geleitet werden müssen. Hierzu wird im Mailserver eine catch-all Regel konfiguriert. Mit dieser Regel werden alle eingehenden E-Mails, die an eine per VERP individualisierte Adresse gerichtet sind, in ein gemeinsames Postfach geleitet. Zur Verarbeitung der Bounces liest der Universal Messenger alle E-Mails aus diesem gemeinsamen Postfach aus.

8.9.5 Konfiguration von VERP im SMTP-Server

Auch der SMTP-Server muss für den Versand der mit individuellen Envelope-From Adressen versehenen E-Mails entsprechend konfiguriert werden, so dass er diese Mails vom Newslettersystem entgegennimmt und in das Internet ausliefert.

Zum Spamcheck führen viele empfangende Mailserver einen "Callout" an den sendenden Mailserver durch, um zu prüfen, ob die E-Mail wirklich von dort versendet wurde. Es gibt hierfür zwar das spezielle SMTP-Kommando `VERIFY`, das aber nicht durchgängig unterstützt wird. Deswegen wird meist eine SMTP-Verbindung aufgebaut und simuliert, als ob eine E-Mail versendet werden soll und dann durch ein `QUIT` abgebrochen. Der SMTP-Mailserver muss bei der Konfiguration für VERP also E-Mails an die individuellen Envelope-From Adressen annehmen und das auch nach außen signalisieren.

8.10 E-Mail-Empfang per POP3

Das Bounce-Management kann zwar auch POP3-Server abfragen, dieses Vorgehen wird allerdings nicht empfohlen. Häufig ist POP3 aber bei Lotus Notes oder Exchange-Installationen notwendig, bei denen kein IMAP zur Verfügung steht.

Die größte Einschränkung bei der Benutzung von POP3 ist, dass im Server keine Folder hinterlegt werden können. Das Bounce-Management verwendet einige Folder (vor allem "error"), um bestimmte eingehende E-Mails aufzuheben.

Wenn eingehende E-Mails per POP3 abgefragt werden, müssen u.a. die Folder "error" und "debug" lokal auf dem Server des Universal Messenger abgelegt werden. Standardmäßig werden sie in diesem Fall unterhalb des `cmsbs-work/maildir/default/-`Verzeichnisses im Maildir-Format abgelegt. Dort können sie z.B. mit Hilfe eines IMAP-Servers zugänglich gemacht werden.

Um die Bounce-E-Mails per POP3 abzuholen, müssen folgende Einstellungen in `cmsbs-conf/cmsbs.properties` gesetzt werden:

```
cmsbs.mail.imap.protocol = "pop3"
cmsbs.mail.imap.port = "995"
cmsbs.mail.imap.ssl = "true"
```

(Der Port ist 110, wenn kein SSL verwendet werden kann.)

Daneben gelten die sonst auch verwendeten Einstellungen:

```
cmsbs.startbounce = "true"
cmsbs.mail.imap.server = "SERVER"
cmsbs.mail.imap.user = "USER"
cmsbs.mail.imap.password = "PASSWORT"
```

Desweiteren steuern folgende Einstellungen, wo die lokalen Maildirs abgelegt werden:

```
# Standardverzeichnis für alle Maildir-Verzeichnisse
cmsbs.mail.imap.local = "cmsbs-work/maildir"
# Verzeichnis für den Standard-Listener
cmsbs.mail.imap.local.folder = "cmsbs-work/maildir/default"
```

Auch die zusätzlichen Bounce-Listener können wahlweise auf POP3 oder IMAP gestellt werden:

```
cmsbs.bounce.listeners = "P1 M2"

cmsbs.bouncelistener.P1.mail.imap.server = "some.pop3.server.example.com"
cmsbs.bouncelistener.P1.mail.imap.protocol = "pop3"
cmsbs.bouncelistener.P1.mail.imap.port = "995"
cmsbs.bouncelistener.P1.mail.imap.ssl = "true"
cmsbs.bouncelistener.P1.mail.imap.user = "USER"
cmsbs.bouncelistener.P1.mail.imap.password = "PASS"
# Implies default:
# cmsbs.bouncelistener.P1.local.folder = cmsbs-work/maildir/P1

cmsbs.bouncelistener.M2.mail.imap.server = "some.imap.server.example.com"
cmsbs.bouncelistener.M2.mail.imap.protocol = "imap"
cmsbs.bouncelistener.M2.mail.imap.port = "993"
cmsbs.bouncelistener.M2.mail.imap.ssl = "true"
cmsbs.bouncelistener.M2.mail.imap.user = "USER"
cmsbs.bouncelistener.M2.mail.imap.password = "PASS"
```

8.11 Bounce Management mit AWS

Beim E-Mail-Versand per AWS SES (siehe [E-Mail-Versand](#) unten) erfolgt der Empfang von Bounces nicht wie sonst über ein IMAP-Postfach, sondern wird von AWS übernommen und durch Aufruf einer URL an den Universal Messenger weitergemeldet.

Als Endpunkt für diesen *Web Hook* dient im Universal Messenger der REST-Controller `de.pinuts.aws.ses.api`, der per REST-Proxy aus dem Internet erreichbar gemacht werden muss. Zur Einrichtung des REST-Proxy siehe auch [REST-Proxy](#).

8.11.1 Konfiguration in AWS / SNS

In AWS SNS müssen drei *Topics* eingerichtet werden; jeweils eins für:

1. Bounces
2. Deliverys
3. Complaints

Für jedes der drei Topics muss eine *Subscription* eingerichtet werden:

- Protokoll: HTTPS
- Endpunkt: `https://um.acme.com/p/de.pinuts.aws.ses.api/event`
- (`https://um.acme.com/p` muss durch die öffentlich erreichbare URL des REST-Proxy ersetzt werden.)

8.11.2 Konfiguration in AWS / SES

In AWS SES müssen nun für jede Absenderadresse oder -domain die oben angelegten SNS-Topics als *Feedback notifications* angemeldet werden.

1. Bounce feedback
2. Complaint feedback
3. Delivery feedback

Die Option *Include original email headers* muss jeweils gesetzt sein.

8.11.3 Validierung / Bestätigung der SNS-Topics

Zum Schluss müssen die drei SNS-Topic-Endpunkte noch bestätigt werden. Dazu sendet AWS SNS automatisch an jeden der angelegten Endpunkte einen Request, der im Log dessen angezeigt wird. Dieser Request enthält u.a. eine URL, die manuell im Browser aufgerufen werden muss, um den Endpunkt zu bestätigen.

Sind alle Endpunkte bestätigt, wird das in der Übersichtsliste der *Subscriptions* in AWS SNS entsprechend angezeigt.

8.12 Bounce Management mit SendGrid

Beim E-Mail-Versand per SendGrid (siehe [E-Mail Versand](#) unten) erfolgt der Empfang von Bounces nicht wie sonst über ein IMAP-Postfach, sondern wird von SendGrid übernommen und durch Aufruf einer URL an den Universal Messenger weitergemeldet.

Als Endpunkt für diesen *Web Hook* dient im Universal Messenger der REST-Controller `de.pinuts.cmsbs.sendgrid.Webhook`, der per REST-Proxy aus dem Internet erreichbar gemacht werden muss. Zur Einrichtung des REST-Proxy siehe auch [REST-Proxy](#).

8.12.1 Konfiguration in SendGrid

Unter [SendGrid / Mail Settings](#) muss *Event Notification* aktiviert und konfiguriert werden:

The screenshot shows the SendGrid Mail Settings page. On the left is a sidebar with navigation links: DASHBOARD, MARKETING, TEMPLATES, STATS, ACTIVITY, SUPPRESSIONS, and SETTINGS. The SETTINGS section is expanded, showing sub-links like Account Details, Plan & Billing Details, Whitelabels, Overview, Domains, Email Links, Alert Settings, Inbound Parse, Mail Settings (selected), Partners, Tracking, API Keys, Credentials, and Multifactor Authentication. Below the sidebar, there are status indicators: REPUTATION 75%, EMAILS REMAINING 12K / 12K, and EMAILS SENT TODAY 0.

The main content area displays a table of settings:

STATE	SETTING	DESCRIPTION	OPTIONS
INACTIVE	Address Whitelist	Address / domains that should never have email suppressed.	▼
INACTIVE	BCC	Automatically BCC an address for every e-mail sent.	▼
INACTIVE	Bounce Purge	Allows you to automatically purge bounce records from SendGrid after a specified number of days.	▼
ACTIVE	Event Notification	Controls notifications for events, such as bounces, clicks, and opens.	▲

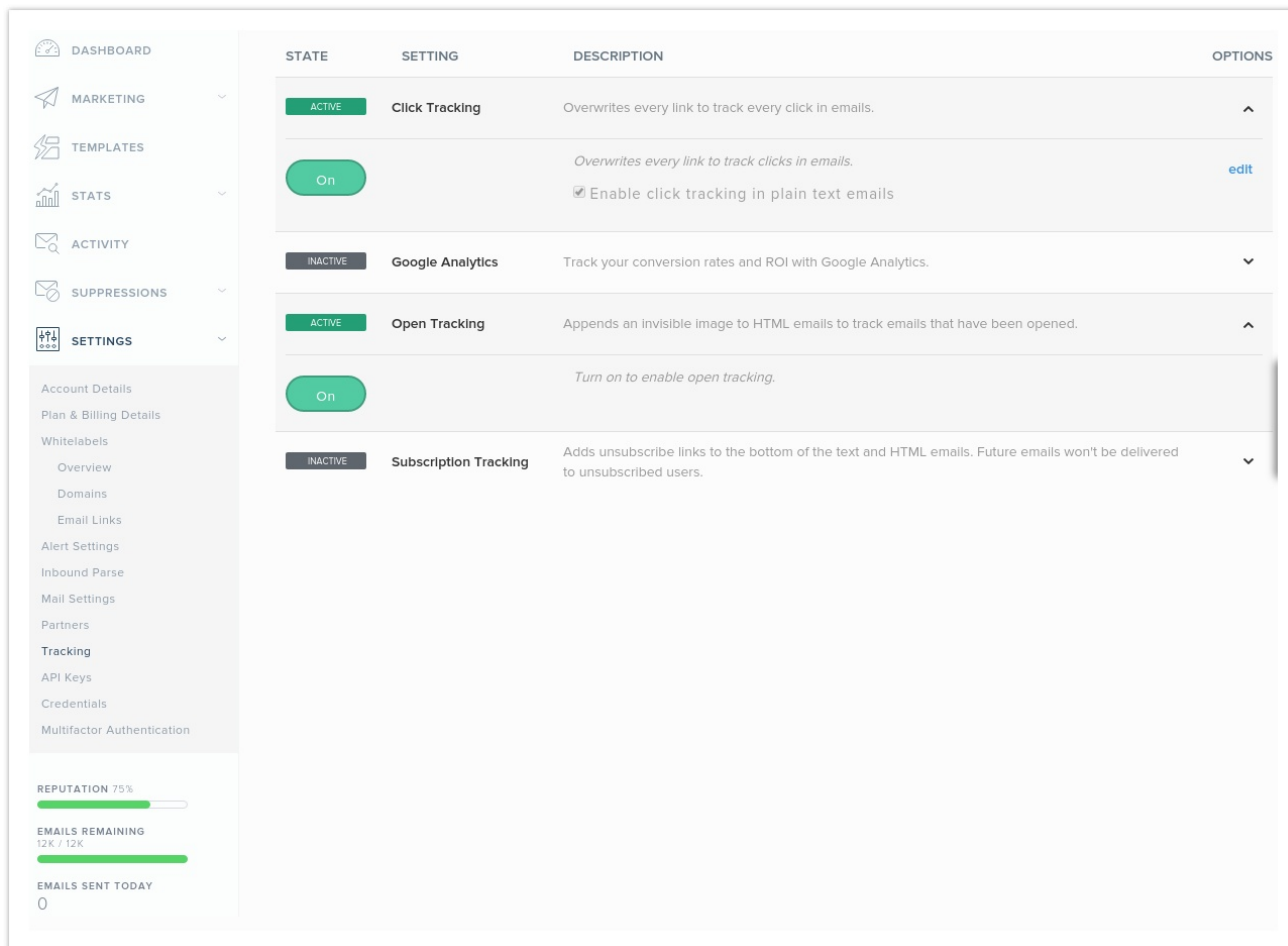
The **Event Notification** setting is expanded, showing a toggle switch set to **On**. Below the toggle is a **CONFIGURATION** section with a note: "You can utilize our Event Notification framework to gather the data required for the most detailed reporting. If you need more assistance, please see our [Event Notification documentation](#). Please note: If the open and click tracking apps are not enabled, the open and click events will not be posted." Below this is the **HTTP POST URL** field, which contains the text: `https://www.example.org/restproxy/de.pinuts.cmsbs.sendgrid.Webhook/index`. There is also an **INTEGRATION TESTING TOOL** section with a button labeled "Test Your Integration". At the bottom is the **SELECT ACTIONS** section, which allows selecting which actions to report back to the system. The selected actions are: All, Processed, Dropped, Deferred, Delivered, Bounced, Opened, Clicked, Unsubscribed From, Mark as Spam, ASM Group Unsubscribe, and ASM Group Resubscribe.

Unter *HTTP POST URL* muss die aus dem Internet erreichbare URL des obengenannten Web Hooks angegeben werden. Diese URL setzt sich zusammen aus der Basis-URL des REST-Proxy (hier im Beispiel `https://www.example.org/restproxy`) und dem Pfad zum Web Hook: `de.pinuts.cmsbs.sendgrid.Webhook/index`

Unter *ACTIONS* sollten wie im Screenshot folgende Optionen aktiviert werden:

- Dropped
- Delivered
- Bounced
- Opened (optional, siehe unten)
- Clicked (optional, siehe unten)

Falls Click- bzw. View-Tracking auch über SendGrid abgewickelt werden sollen, müssen zusätzlich unter [SendGrid / Tracking](#) die Funktionen *Click Tracking* und *Open Tracking* aktiviert werden:



8.13 Bounce Management mit Mailjet

Beim E-Mail-Versand per Mailjet (siehe [E-Mail Versand](#) unten) erfolgt der Empfang von Bounces nicht wie sonst über ein IMAP-Postfach, sondern wird von Mailjet übernommen und durch Aufruf einer URL an den Universal Messenger weitergemeldet.

Als Endpunkt für diesen *Web Hook* dient im Universal Messenger der REST-Controller `de.pinuts.cmsbs.mailjet.Webhook`, der per REST-Proxy aus dem Internet erreichbar gemacht werden muss. Zur Einrichtung des REST-Proxy siehe auch [REST-Proxy](#).

8.13.1 Konfiguration in Mailjet

Unter [Mailjet / Event API](#) muss die URL zu diesem Web Hook angegeben und die gewünschten Events ausgewählt werden. Diese URL setzt sich zusammen aus der Basis-URL des REST-Proxy (hier im Beispiel `https://example.org/p`) und dem Pfad zum Web Hook: `de.pinuts.cmsbs.mailjet.Webhook/events`.

Daraus ergibt sich in diesem Beispiel folgende URL:

<https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook/events>

Es werden nur die folgenden *Events* benötigt:

- Versendet
- Bounce
- Blockiert

Event	Endpoint-URL	Events zusammenfassen ?	Test versenden ?	Testergebnis
<input type="checkbox"/> Alle auswählen	<input type="text" value="https://example.org/p/de.pir"/>	<input type="button" value="Auf alle anwenden"/>	<input type="checkbox"/> Alle auswählen	
<input checked="" type="checkbox"/> Versendet	<input type="text" value="https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook"/>	<input checked="" type="checkbox"/>	<input type="button" value="Versenden"/>	
<input type="checkbox"/> Öffnungen	<input type="text" value="https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook"/>	<input type="checkbox"/>	<input type="button" value="Versenden"/>	
<input type="checkbox"/> Klicks	<input type="text" value="https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook"/>	<input type="checkbox"/>	<input type="button" value="Versenden"/>	
<input checked="" type="checkbox"/> Bounce	<input type="text" value="https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook"/>	<input checked="" type="checkbox"/>	<input type="button" value="Versenden"/>	
<input type="checkbox"/> Spam	<input type="text" value="https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook"/>	<input type="checkbox"/>	<input type="button" value="Versenden"/>	
<input checked="" type="checkbox"/> Blockiert	<input type="text" value="https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook"/>	<input checked="" type="checkbox"/>	<input type="button" value="Versenden"/>	
<input type="checkbox"/> Abbestellt	<input type="text" value="https://example.org/p/de.pinuts.cmsbs.mailjet.Webhook"/>	<input type="checkbox"/>	<input type="button" value="Versenden"/>	

[← Zurück zu meinem Konto](#)

8.14 Feedback-Loops

In der Customer Interaction Edition kann der Universal Messenger Feedback-Reports im [Abuse Reporting Format](#) auswerten und entsprechend reagieren. Der dafür erforderliche Bounce-Handler wird in der globalen Konfigurationsdatei `cmsbs.properties` wie folgt aktiviert:

```
cmsbs.bounce.feedbackreports = true
```

Um die Zuordnung der Report-Mail zu Empfänger und Newsletter bzw. Benachrichtigung zu ermöglichen, muss ein zusätzlicher Mail-Header konfiguriert werden, welcher dann aus der empfangenen Report-Mail ausgelesen werden kann:

```
cmsbs.mail.headers = "X-UM-Delivery-Ticket"
cmsbs.mail.header.X-UM-Delivery-Ticket = "{msgid}"
```

In der Standardimplementierung werden die folgenden Schritte ausgeführt, sobald ein Feedback-Report empfangen und eindeutig einem Empfänger zugeordnet werden konnte:

1. Austragen aus allen Listen inkl. Eintrag in der jeweiligen Historientabelle ("chaction" bzw. "newsletter_channel_activity"),
2. Tracken eines Bounce vom Typ FBL_ABUSE, FBL_FRAUD, FBL_VIRUS oder FBL_OTHER und
3. Archivierung der Report-Mail, beim jeweiligen Empfänger, sofern das Tabellenattribut "incoming_mail" vorhanden ist (siehe dazu [Archivierung eingegangener E-Mails](#)).

Diese Standardimplementierung kann per Core Scripting überschrieben werden.

8.15 Archivierung eingegangener E-Mails

8.15.1 Zugeordnete Bounces

Seit Release 7.51 besteht mit allen Lizenzen die Möglichkeit, eingegangene Bounces, die einem konkreten *Eintrag* zugeordnet werden konnten, zu archivieren.

Bei Bestandsinstallationen kann die Archivierung durch Hinzufügen folgender Zeile in `cmsbs-conf/additional.attributes` aktiviert werden:

```
include: cmsbs-conf/cse/api/plugins/de.pinuts.cmsbs.bouncemanagement/incoming_mail.attributes
```

Bei Installationen, in denen diese Datei bei jedem Update aus dem Installer übernommen wird (z.B. in Docker-Umgebungen), kann es durch dieses Update dazu kommen, dass die Datei `incoming_mail.attributes` mehrfach inkludiert wird. Dies führt zu einem Fehler beim Start, der durch Entfernen der doppelten Konfigurationszeile behoben werden kann.

Sobald dieses Attribut vorhanden ist, werden alle eingegangenen und zugeordneten Bounces darin gespeichert und können über die GUI betrachtet werden. Das Tabellenattribut *Eingegangene E-Mails* ("incoming_mail") wird in der Attributgruppe *Mail-Eingang* angezeigt:

Mail-Eingang			
Eingegangene E-Mails			
Datum	Bounce-Typ	Betreff	E-Mail
Fr, 19. Jun. 2015, 11:42:55	Unknown	Ich bin zurzeit im Urlaub	✉ E-Mail anzeigen
Do, 01. Okt. 2015, 10:14:12	Hardbounce	Mail delivery failed: returning message to sender	✉ E-Mail anzeigen

Über den Link *E-Mail anzeigen* wird die jeweilige E-Mail in einem Dialogfenster angezeigt. Dabei erfolgt aus Sicherheitsgründen eine Filterung des HTML-Inhalts und der angezeigten Attachments.

8.15.2 Nicht-zuordenbare Bounces

Bounces, die keinem konkreten Eintrag zugeordnet werden konnten, können ebenfalls archiviert werden. Diese Funktion muss durch Setzen der Konfigurationsoption

```
cmsbs.bounce.IncomingMailArchive = true
```

aktiviert werden.

Unter *Extras / Unerkannte Bounces* wird dann eine Liste der eingegangenen E-Mails angezeigt, die keinem Eintrag zugeordnet werden konnte.

9 Newsletter Controlling

Mit dem Universal Messenger können Sie die Öffnungsrate eines HTML-Newsletters und die Klickraten auf die enthaltenen Links auswerten. Dazu wird der Newsletter nach den folgenden Prinzipien erweitert:

1. Es wird ein unsichtbares Zählpixel in den Newsletter eingefügt, über das das Öffnen des Newsletters erkannt wird.
2. Alle Links werden über einen Redirector zur Weiterleitung umgeleitet, so dass über den Aufruf des Redirectors der Klick auf die Links gezählt werden kann.

Die automatische Ergänzung des Newsletters wird über einige Optionen in der Konfigurationsdatei `cmsbs.properties` kontrolliert. Das Tracking kann entweder mit dem Universal Messenger durchgeführt werden oder über externe Anbieter wie z.B. etracker.

Wird der Universal Messenger für das Tracking verwendet, so stehen im Newsletterarchiv und bei den einzelnen Personen Informationen über die Öffnungen und Klicks zur Verfügung. Die Verwendung des integrierten Trackings ist empfehlenswert, da hierbei eine direkte Verbindung zwischen Einträgen, versendeten Newslettern und versendeten Benachrichtigungen ermöglicht wird.

9.1 Konfiguration des Trackings

Im Folgenden wird die Konfiguration des Newsletter-Controllings in der Datei `cmsbs.properties` beschrieben. Eine grundsätzliche Einführung in dieses Thema bietet das Handbuch zur Benutzeroberfläche im Abschnitt Newsletter-Tracking.

```
cmsbs.tracker.mode = personal|anonymous|off
```

Über diese Einstellung wird die Art des Trackings definiert.

- **personal:** Personen-bezogenes Tracking
- **anonymous:** Anonymisiertes Tracking
- **off:** Kein Tracking

Die Zustimmung der Empfänger wird als Einwilligung in den Consent-Attributen gespeichert. Der Name der Consent-Attribute kann über Konfigurationsoptionen gesetzt werden. Die folgende Tabelle führt diese Konfigurationsoptionen zusammen mit den Standardwerten auf. Bei einem Update des Universal Messenger von einer Version älter als 7.21 werden die Standardwerte so gesetzt, dass sie keine Änderungen an der Attributkonfiguration erfordern und das Verhalten des Universal Messenger möglichst unverändert bleibt. Wenn die neuen Funktionen des Universal Messenger genutzt werden sollen, muss die Konfiguration allerdings angepasst werden.

Konfigurationsoption	Default bei Update	Default bei Neuinstallation	Beschreibung
----------------------	--------------------	-----------------------------	--------------

<code>cmsbs.tracker.tracking_personal.attribute</code>	<code>behavior_date</code>	<code>consent_nl_tracking_personal</code>	Name des Consent-Attribut für personen-bezog Tracking
<code>cmsbs.tracker.tracking_anonymous.attribute</code>		<code>consent_nl_tracking_anonymous</code>	Name des Consent-Attribut für anonymes Tracking
<code>cmsbs.tracker.tracking_optout.attribute</code>	<code>anonymous</code>	<code>consent_nl_tracking_optout</code>	Name des Consent-Attribut für explizites Tracking Opt-Ou

```
cmsbs.tracker.fallback_mode = personal|anonymous|off
```

Fallback-Tracking-Modus, wenn beim Empfänger kein Consent-Attribut gesetzt ist

```
cmsbs.tracker.secret = <Hashing Salt, mind. 20 Zeichen lang>
```

Die Angabe `cmsbs.tracker.secret` ist optional. Ist sie vorhanden und der Wert nicht leer, überprüft der Linktracker die übergebene URL anhand eines ebenfalls übergebenen Hash-Wertes. Dadurch kann sichergestellt werden, dass der Linktracker nicht für Phishing missbraucht werden kann.

Setzen des Wertes auf "`{random}`" bewirkt, dass beim ersten Start ein Zufallswert erzeugt und in der Datenbank gespeichert wird. Bei allen folgenden Starts wird dann der Wert aus der Datenbank verwendet.

Wird der Wert später in der Konfigurationsdatei auf einen anderen festen Wert geändert, so wird dieser Wert beim nächsten Start in die Datenbank übernommen.

```
cmsbs.tracker.autoview = true|false
```

Lädt ein Abonnent die Bilder des Newsletters nicht nach und damit auch nicht das Zählpixel, wird eine Newsletter-Öffnung nicht getrackt. Klickt dieser Abonnent auf einen Link, wird jedoch sein Klick getrackt, woraus sich eine ungenaue Statistik ergibt. Diese Konfigurationsoption, die standardmäßig auf `true` gesetzt ist, sorgt dafür, dass bei einem gezählten Link-Tracking implizit ein View-Tracking mitgezählt wird, wenn dieses noch nicht geschehen ist.

```
cmsbs.gui.showtracking = true|false
```

Gibt global an, ob das Tracking auf der Benutzeroberfläche des Universal Messenger angezeigt wird.

9.1.1 Tracking der Newsletter Öffnungen

Mit der folgenden Konfigurationsoption wird die Personalisierungsvariable `{trackerpixel}` definiert, die an geeigneter Stelle in den HTML-Newsletter eingefügt werden muss. Beim Versand des Newsletters wird der Universal Messenger die Personalisierungsvariable `{trackerpixel}` durch den hier definierten Inhalt ersetzen und damit ein Zählpixel in den Newsletter einfügen.

```
cmsbs.tracker.pixel = <Formatierungsstring>
```

Angabe eines Formatierungsstrings für die Ausgabe des Zählpixels in einem HTML-Newsletter. Der Formatierungsstring erlaubt alle Angaben (Variablen und Kontrollstrukturen), die auch bei der Personalisierung eines Newsletters erlaubt sind. Weiterhin können über die Formatierungsanweisungen der Form `{special:variablenname}` die Werte von Variablen aus dem Content Management System eingefügt werden, die in der XML-Eventdatei definiert werden, siehe dazu [Variablen zur Personalisierung mit global](#).

Beispiel für das Tracking mit dem Universal Messenger:

```
cmsbs.tracker.pixel = "<img
src='http://SERVER:PORT/cmsbs-proxy/t/nt?t={msgid}&v={splitTestVariant}' border='0'
width='1' height='1' />"
```

Beispiel für ein Tracking mit etracker:

```
cmsbs.tracker.pixel = "<img
src='http://www.etracker.de/cnt.php?et=XXXXX&java=n&et_easy=0&et_pagename={urlencode
{email}}&et_areas=HTMLNewsletter&et_ilevel=0&et_target=,,0&et_lpage=&et_tri
width='1' height='1' border='0' />"
```

9.1.2 Link-Tracking

Mit der folgenden Konfigurationsoption werden Formatierungsanweisungen definiert, mit denen alle absoluten Links innerhalb des Newsletters formatiert werden können, um die Links zur Zählung der Klicks über ein Zählskript umzuleiten.

```
cmsbs.tracker.link = <Formatierungsstring>
```

Angabe eines Formatierungsstrings, mit dem alle absoluten Links in einem E-Mail-Newsletter bearbeitet werden. Der Formatierungsstring erlaubt alle Angaben (Variablen und Kontrollstrukturen), die auch bei der Personalisierung eines Newsletters erlaubt sind. Zusätzlich können folgende spezielle Variablen verwendet werden:

Variable	Beschreibung
----------	--------------

<code>trackedurl</code>	enthält den ursprünglichen absoluten Link in einer URL-kodierten Form (unter Beachtung des System Default Encodings), so dass er einem Redirector als GET-Parameter übergeben werden kann.
<code>trackedurl:raw</code>	enthält den ursprünglichen absoluten Link mit Universal Messenger Markup und ohne Personalisierung.
<code>trackedurlname</code>	enthält den Namen des ursprünglichen absoluten Links in einer URL-kodierten Form, so dass er einem Redirector als GET-Parameter übergeben werden kann. Der Name des Links wird in dem <code>a href</code> Tag als <code>name</code> Attribut angegeben. Wenn das Attribut leer oder nicht angegeben ist, wird <code>trackedurl:raw</code> verwendet.
<code>trackedurltitle</code>	enthält den Titel des ursprünglich absoluten Links. Der Name des Links wird in dem <code>a href</code> Tag als <code>title</code> Attribut angegeben.
<code>trackedurlid</code>	enthält eine eindeutige ID, die automatisch für jeden Link gesetzt wird. (seit Version 7.3)
<code>trackedurl:sha1</code>	enthält den SHA1-Hashwert der Ziel-URL zum Schutz gegen Missbrauch des Linktrackers.
<code>splitTestVariant</code>	enthält die Nummer (0, ...) der Split-Test-Variante, welche für den jeweiligen Empfänger ausgewählt wurde.

Weiterhin können über die Formatierungsanweisungen der Form `{special:variablenname}` die Werte von Variablen eingefügt werden, die in der XML-Eventdatei definiert werden, siehe dazu [Variablen zur Personalisierung mit global](#).

Beispiel für das Tracking per REST-Proxy ab Version 7.3:

```
cmsbs.tracker.link =
"http://SERVER:PORT/cmsbs-proxy/t/nl?t={msgid}&d={trackedurl}&i={trackedurlid}&
h={trackedurl:sha1}&v={splitTestVariant}"
```

Beispiel für das Tracking per REST-Proxy bis Version 7.2:

```
cmsbs.tracker.link =
"http://SERVER:PORT/cmsbs-proxy/t/nl?t={msgid}&d={trackedurl}&n={trackedurlname}&
h={trackedurl:sha1}"
```

Beispiel für ein Tracking mit etracker (Einbindung von Google Analytics erfolgt analog):

```
cmsbs.tracker.link = "http://www.etracker.de/lkcnt.php?et={urlencode:{special|etracker}}&
url={trackedurl}&lnkname={trackedurlname}"
```

9.1.2.1 Delivery-Ticket / msgid an Links anhängen

Die globale Konfigurationsoption `cmsbs.tracker.alwaysAppendMsgid` aktiviert das automatische Anhängen des Delivery-Tickets an jeden vom Linktracking behandelten Link. Das Delivery-Ticket wird vor dem Redirect als URL-Parameter `msgid` an die gegebene URL angehängt.

9.1.2.2 Conversion beim Linktracking

Beim Linktracking per REST-Proxy kann durch einfaches Hinzufügen von URL-Parametern an den zu trackenden Link automatisch ein Conversiontracking implementiert werden.

Beispiel: Ein Link im Newsletter enthält folgenden Link:

```
<a href="https://www.acme.com/shop/?umc.what=enter-shop&umc.value=1&x=true">Shop</a>
```

Wenn für den Newsletter das Linktracking aktiviert ist, wird hier beim Klick auf den Link im Newsletter nicht nur der Klick gezählt, sondern auch eine Conversion mit dem Schlüsselwort `enter-shop` und dem Wert `1` gespeichert. Beim folgenden Redirect werden die beiden `umc.*`-Parameter entfernt, so dass der Nutzer auf die Seite `https://www.acme.com/shop/?x=true` weitergeleitet wird.

9.1.3 Konfiguration im Event-File

Die globale Einstellung kann bei Bedarf im Event-File für einen einzelnen Versandvorgang überschrieben werden:

```
<event>
<destination>...</destination>
<data>
<email trackingMode="[personal | anonymous | off]" ...>
<subject>...</subject>
<plaintext ...>...</plaintext>
<htmltext ... />
</email>
</data>
</event>
```

9.2 Installation und Anwendung

Der Universal Messenger ermöglicht die Protokollierung von Benutzeraktionen wie Klicks auf Links, Öffnungen von Newslettern, Abmeldungen von Newslettern und Aktionen in Drittsystemen wie Käufe in Shops, die vom Newsletter ausgelöst wurden.

Die Auswertung erfolgt im Newsletter-Archiv (siehe Handbuch zur Benutzeroberfläche "Newsletter Archiv"). Pro Newsletter werden hierbei die Aktionen aufgeführt, die über den Newsletter ausgeführt werden. Es ist so auch eindeutig zu sehen, welcher Newsletter eine Abmeldung bewirkt hat oder welche Links häufig angeklickt werden. Ebenfalls können die Aktionen des Benutzers direkt auf der Eintragsseite des Benutzers eingesehen werden.

Das Tracking der Benutzeraktionen erfolgt über unterschiedliche Technologien. Für das Zählen der Öffnung wird ein Zählpixel verwendet, angeklickte Links werden über einen Redirector ("Weiterleitungsseite") geführt. Für beide Funktionen wird der REST-Proxy benötigt, der zusammen mit dem Universal Messenger ausgeliefert wird. Die im Newsletter ausgeführten Benutzeraktionen werden über den REST-Proxy geleitet, der die Tracking-Informationen an den Universal Messenger weitergibt.

Für die Protokollierung von Abmeldungen und Aktionen in Drittsystemen (Conversions) werden ebenfalls Beispiele mitgeliefert, die als Basis für die Integration in die eigene Umgebung (Abmeldungsseiten, Shop-Systeme) dienen können.

9.2.1 Tracking per REST-Proxy

Im Folgenden wird die Einrichtung des Newsletter Trackings mit den Standard-Komponenten des Universal Messenger erklärt, so dass neben dem Tracking auch die personalisierte Version eines HTML-Newsletters als "Newsletter Archiv" im Browser angezeigt werden kann.

Im Beispiel wird mit `http://intern.example.com/` der interne Host bezeichnet, auf dem der Universal Messenger installiert wird. `http://www.example.com/` bezeichnet die im Internet sichtbare Website.

Schritt 1: Web-Komponente installieren

Deployen Sie `<UM_HOME>/web-integration/cmsbs-restproxy.war` als Web-Applikation in einem externen Context auf einem Server separat von dem Universal Messenger, so dass die Web-Applikation als `http://www.example.com/p/` im Internet sichtbar ist. Der Name "cmsbs-restproxy" sollte nicht erhalten bleiben, sondern möglichst kurz werden.

Details zu dieser Installation finden Sie im *Handbuch für Administratoren* im Abschnitt REST-Proxy.

Die Definition des Contextes in einem Tomcat sieht mit Parametern dann etwa so aus:

```
<Context path="/p" docBase="/PFAD/cmsbs-restproxy.war">
<Parameter name="cmsbs.resturl" value="http://intern.example.com:8080/cmsbs/rest/"
override="false"/>
<Parameter name="cmsbs.restproxy.tracking" value="true" override="false"/>
<Parameter name="cmsbs.restproxy.urlprefix" value="http://www.example.com/p/" override="false"/>
</Context>
```

Schritt 2: Tracking konfigurieren

In `<UM_HOME>/cmsbs-conf/cmsbs.properties` müssen dann folgende Einstellungen vorgenommen werden:

```
cmsbs.tracker.pixel = 
cmsbs.tracker.link =
http://www.example.com/p/t/nl?t={msgid}&d={trackedurl}&i={trackedurlid}&h={trackedurl:sha1}
```

Nach einem Neustart des Tomcat steht das Tracking nun zur Verfügung.

Schritt 3: Browser-Link im HTML-Template hinterlegen

Der folgende Abschnitt muss am Anfang der HTML-Newsletter-Vorlage (meist im CMS) hinterlegt werden.

```
{ifcse¹showBrowserLink()¹
...
Für Anzeige im Browser
<a title="Browser-Ansicht" href="http://www.example.com/p/t/review/{msgid}"/>klicken Sie hier!<a>
...
}
```

Die "ifcse"-Klammer wird um den kompletten HTML-Block herum gesetzt, der nur beim Versand als E-Mail angezeigt wird und beim Abruf des Newsletters aus dem Newsletter Archiv ausgeblendet wird. Üblicherweise bildet dieser HTML-Block ein DIV oder eine Tabellen-Zeile, die vor dem Inhalt steht.

Schritt 4: Whitelisteinträge erstellen

Folgende Whitelisteinträge müssen in der REST-Proxy Konfiguration eingetragen werden:

Beispiel: TOMCAT/conf/Catalina/localhost/cmsbs-restproxy.xml

```
<Context path="/p" docBase="/PFAD/cmsbs-restproxy.war">
...
    <Parameter name="cmsbs.restproxy.limit.controller.whitelist.0" value="t" />
    <Parameter name="cmsbs.restproxy.limit.controller.whitelist.1"
value="de.pinuts.cmsbs.api.tracking.Application" />
...
</Context>
```

Optionale Einstellungen für das Caching im REST-Proxy

Da das Tracking und die Darstellung des Newsletters im Browser über den REST-Proxy stattfindet, können Inhalte eine bestimmte Zeit gecached werden.

Die folgenden Parameter können optional angepasst werden:

```
cmsbs.tracker.trackingpixel.ttl = <Sekunden>
```

Gibt in Sekunden an, wie lange ein Trackingpixel vom Proxy gecached wird. Wird das Pixel aus dem Cache geladen, wird dieser View trotzdem gezählt. Dies geschieht über eine Queue, welche die Views asynchron an den Universal Messenger weiterleitet, der sie dann speichert. Standardmäßig wird das Trackingpixel unbegrenzt lange im Cache gespeichert.

```
cmsbs.tracker.linktracking.ttl = <Sekunden>
```

Gibt in Sekunden an, wie lange ein Link-Tracking-Redirect im Proxy gecached wird. Wird ein Link-Tracking-Redirect aus dem Cache geladen, wird dieser trotzdem gezählt. Dies geschieht über eine Queue, welche die Link-Tracks asynchron an den Universal Messenger weiterleitet, der sie dann speichert. Standardmäßig wird ein Link-Track unbegrenzt lange im Cache gespeichert.

```
cmsbs.tracker.review.ttl.asset = <Sekunden>
```

Gibt in Sekunden an, wie lange Grafiken aus der Browser-Ansicht des Newsletters im Proxy gecached werden. Standardmäßig werden Grafiken unbegrenzt lange im Cache gespeichert.

```
cmsbs.tracker.review.ttl.body = <Sekunden>
```

Gibt in Sekunden an, wie lange ein personalisierter Body aus der Browser-Ansicht des Newsletters im Proxy gecached wird. Standardmäßig wird der Body 10 Minuten im Cache gespeichert. Dies bedeutet, dass jeder weitere View innerhalb dieser 10 Minuten nicht getrackt wird.

```
cmsbs.tracker.error.ttl = <Sekunden>
```

Gibt in Sekunden an, wie lange ein 404 Error aus der Browser-Ansicht des Newsletters im Proxy gecached wird.

9.2.2 Integration in Newsletter

Ob das Tracking für einen Newsletter verwendet werden soll, kann über die XML-Event-Datei gesteuert werden:

```
<plaintext inline="false" viewTracking="false"
linkTracking="false">http://www.domain.de/newsletter.txt</plaintext>
<htmltext inline="false" viewTracking="true"
linkTracking="true">http://www.domain.de/newsletter.html</htmltext>
```

Conversions und Unsubscriptions können direkt über die installierten Komponenten geführt werden, da diese ähnlich wie das Zählpixel eine Grafik ausgeben und somit als verstecktes Bild in Seiten/Formulare eingefügt werden können. Wir empfehlen aber, den Code der installierten Komponenten als Beispiel für eine direkte Integration in die Applikation (z.B. durchgeführte Bestellungen, erfolgte Newsletter-Abmeldungen) zu verwenden.

9.2.2.1 Link-Tracking Konfiguration

Zur besseren Auswertung von Links in Newslettern gibt es einige Optionen, die mit zusätzlichen Attributen im Link-Tag gesteuert werden können. Diese Attribute beginnen alle mit `data-cmsbs` und dienen lediglich zur Übergabe an den Universal Messenger und werden beim Versand des Newsletters automatisch entfernt.

Möchten Sie mehrere Links im Newsletter für die Auswertung zu einem logischen Link zusammenfassen, bspw. einen Link zur Firmen-Homepage, der mehrfach im Newsletter vorkommt, können Sie mit dem Attribut `data-cmsbs-id` an alle diese Links die selbe ID mehrfach zuweisen. (Bitte beachten Sie aber, dass diese ID numerisch sein muss.)

```
<a href="http://..." data-cmsbs-id="MeineLinkId">...</a>
```

Bei bestimmten Links im Newsletter soll der Klick nicht als Klick, sondern nur als Öffnung des Newsletters gezählt werden, z.B. bei einem Link, der den Newsletter in einer Browseransicht öffnet. Mit dem Setzen des Attributs `data-cmsbs-track-view` wird der Universal Messenger angewiesen, einen Klick auf diesen Link nur als "View" und nicht als "Click" zu zählen. In der Link- & Klickübersicht des Newsletter-Archivs werden solche Links zwar aufgeführt, aber mit dem Hinweis "kein Tracking" gekennzeichnet.

```
<a href="http://..." data-cmsbs-track-view>Newsletter im Browser ansehen</a>
```

Zur Auswertung des Nutzungsverhaltens können die Links im Newsletter über Tags kategorisiert werden. Die Tags können nach dem Versand in der Benutzeroberfläche des Universal Messenger gesetzt oder auch schon im HTML-Quellcode mit dem Attribut `data-cmsbs-tags` an den Universal Messenger übergeben werden. Mehrere Tags werden durch `" , " , " ; " oder " | "` getrennt und außerdem automatisch getrimmt.

```
<a href="http://..." data-cmsbs-tags="Tag A,Tag B">...</a>
```

Sollen Klicks auf bestimmte Links im Newsletter weder als Klick noch als Öffnung getrackt werden, z.B. bei Links zur Abmeldeseite, können Sie mit dem Attribut `data-cmsbs-dont-track` gekennzeichnet werden.

```
<a href="..." data-cmsbs-dont-track>...</a>
```

Bestimmung des Linktitels

In der Newsletter-Detailansicht wird unter *Link- und Klickübersicht* neben der Ziel-URL auch jeweils der Titel jedes Links angezeigt. Dieser Titel wird seit Version 7.3 wie folgt bestimmt:

- das *title*-Attribut des Linkelements, falls es vorhanden und nicht leer ist;
- das *name*-Attribut des Linkelements, falls es vorhanden und nicht leer ist;
- der Text zwischen öffnendem und schließendem Linktag, falls vorhanden und nicht leer;
- das *id*-Attribut des Linkelements, falls es vorhanden und nicht leer ist;
- die URL selbst in allen anderen Fällen.

9.2.3 Linktracker gegen URL-Manipulationen absichern

Der Linktracker bekommt die Ziel-URL als GET-Parameter übergeben. Ohne weitere Vorkehrungen könnte der Linktracker somit missbraucht werden, um z.B. im Rahmen von Phishing-Mails das wahre Linkziel zu verschleiern. Beispiel:

```
<a href="http://your-public-host/p/t/nl?d=http://phishing-seite.com">Mit entsprechendem  
URL-Encoding des "d"-Parameters ließe sich die Zieladresse sogar noch besser verschleiern, so dass der  
normale Anwender nicht merken würde, dass der Link nicht auf die seriöse Seite "your-public-host" verweist,  
sondern er letztendlich auf der Seite "phishing-seite.com" landen wird.
```

Um diese Art des Missbrauchs zu verhindern, kann der Universal Messenger einen Hashwert (SHA1) für die Ziel-URL berechnen und als weiteren GET-Parameter an den Linktracker übergeben. Der Linktracker kann diesen Hashwert dann selbst ebenfalls errechnen und mit dem übergebenen Wert vergleichen. Nur wenn der übergebene Hashwert mit dem vom Linktracker nachberechneten Hashwert übereinstimmt, gilt die Ziel-URL als authentisch und der Redirect wird tatsächlich ausgelöst.

Zur Umsetzung dieser Schutzstrategie sind die folgenden Schritte erforderlich:

In der Datei `cmsbs.properties` muss ein geheimes Hashing Salt gesetzt werden, das bei der Berechnung des Hashwertes mit einbezogen wird, z.B.:

```
# Secret salt to secure {trackedurl} parameter by SHA1 hash code:  
cmsbs.tracker.secret = "SECRET-HASHING-SALT"
```

Der Wert sollte gegen eine wenigstens 20 Zeichen lange zufällige Zeichenkette ersetzt werden.

Ebenfalls in der Datei `cmsbs.properties` muss die URL für den Linktracker um den GET-Parameter "h" erweitert werden, welcher den Hashwert an den Linktracker übergibt:

```
cmsbs.tracker.link =  
"http://your-public-host/p/t/nl?t={msgid}&d={trackedurl}&i={trackedurlid}&h={trackedurl:sha1}"
```

Der Linktracker selbst muss den Hashing Salt kennen und den Hashwert nachrechnen und überprüfen. Wie der Hashing Salt dem Linktracker bekannt gemacht werden kann, steht oben in den jeweiligen Beschreibungen der drei Tracking-Clients. Seit Build 516 führen alle Tracking-Clients diese Überprüfung durch, sofern das Hashing Salt in dieser Weise festgelegt wurde.

9.3 Schlagworte und Filter

Newsletter können verschlagwortet werden. Die Schlagworte werden im Newsletterarchiv als Aufklappmenü angezeigt, mit welchem nach einem dieser Schlagworte gefiltert werden kann. Es werden anschließend nur die Newsletter angezeigt, die auch diesem Schlagwort zugeordnet wurden.

Gängige Anwendungsfälle

- Trennung zwischen Live- und Testversand
- Unterscheidung mehrerer Sprachen des gleichen Newsletters
- Filterung nach Zielchannel
- Filterung nach Absenderadresse
- Limitierung der Archiv-Anzeige auf bestimmte Newsletterarten (Trennung z.B. nach Fachbereichen, die nur die versendeten Newsletter ihres Fachbereichs sehen dürfen)

Setzen von Schlagworten

Um einen Newsletter mit Schlagworten zu versehen, ist die Event-Datei zu ergänzen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<event archive="true">
  [...]
  <tag>Liveversand</tag>
  <tag>de-de</tag>
  <tag>abonnenten_berlin</tag>
  <tag>newsletter@domain.de</tag>
  [...]
</event>
```

Darstellung von Schlagworten

Schlagworte werden automatisch im Archiv als Aufklappmenü angezeigt. Zusätzlich kann eine Schnellumschaltung zwischen definierten Schlagworten aktiviert werden. Hierzu ist die Konfigurationsdatei `cmsbs-conf/cmsbs.properties` zu überarbeiten:

```
cmsbs.gui.newsarchive.requireTags = "Testversand *Liveversand"
```

Das Beispiel ermöglicht die Schnellumschaltung zwischen den Schlagworten "Testversand" und "Liveversand". Das "*" vor Liveversand definiert dieses Schlagwort als Standard-Schlagwort.

Die über die Schnellumschaltung ausgewählten Schlagworte werden mit den im Aufklappmenü ausgewählten Schlagworten angewendet (UND-Verknüpfung). Es lässt sich hiermit folglich eine Kombination aus Kampagnenauswahl im Aufklappmenü und Test/Liveversand-Unterscheidung über die zusätzliche Schnellauswahl erzeugen.

Die Schlagworte sind zudem vollständig in die Rollenstruktur des Universal Messenger eingebunden. Redakteuren kann somit anhand der Schlagworte nur Zugriff auf bestimmte Newsletter des Archivs gegeben werden:

```
area.NewsletterArchive.tags = "Testversand Liveversand"
```


10 Newsletterversand

Ein Newsletterversand kann auf verschiedenen Wegen gestartet werden:

1. Direkt der Benutzeroberfläche des Universal Messenger
Der Newsletter kann direkt in der Benutzeroberfläche mit der vorlagenbasierte WYSIWYG-Bearbeitung des Smart Editor erstellt werden.
2. Parametrisierter Einsprung in die Benutzeroberfläche des Universal Messenger
Die Angaben für den Newsletterversand können aus einem Drittsystem als URL-Parameter an den Universal Messenger übergeben werden, so dass Voreinstellungen automatisch übernommen werden und der Versand aus der Benutzeroberfläche des Universal Messenger gestartet werden kann.
3. Als Ablage einer XML-Eventdatei im Filesystem des Universal Messenger
Content Management Systeme mit einer statischen Exportfunktion können den Newsletter in das Filesystem des Universal Messenger exportieren und den Newsletterversand über eine XML-Eventdatei automatisch starten.
4. Als Übergabe einer XML-Eventdatei per REST-API
Content Management Systeme mit einer dynamischen Funktionsweise können den Newsletter und die XML-Eventdatei per REST-API an den Universal Messenger übergeben und den Newsletterversand starten.
5. Mit eigener Programmierung per CSE (Core Scripting Engine)
Andere beliebige Abläufe und Prozesse zum Newsletterversand können mit der CSE in serverseitigem JavaScript realisiert werden.

10.1 Parametrisierter GUI Einsprung

Mit dem parametrisierten Einsprung können die Angaben für den Newsletterversand aus einem Drittsystem als URL GET-Parameter an den Universal Messenger übergeben werden, so dass Voreinstellungen automatisch übernommen werden und der Versand aus der Benutzeroberfläche des Universal Messenger gestartet werden kann. Es erfolgt ein Einsprung in den Menüpunkt "Newsletter - Newsletter versenden", wobei die folgenden Parameter übergeben werden können:

GET-Parameter	Beschreibung
htmlSourceUrl	URL, von welcher der HTML-Rumpf geladen und als Newsletter versendet wird.
mailSubject	Betreff des Newsletters
template	Interner Name der Mailingvorlage, die für den Newsletterversand verwendet werden soll.

Alle Parameter sind optional. Wenn ein Parameter angegeben ist, werden in der Benutzeroberfläche die Einstellungen zum Newsletterversand mit den übergebenen Werten vorbelegt.

Beispiel:

`http://server:port/cmsbs/pCreateNewsletter?htmlSourceUrl=https://www.debian.org/releases&mailSuk`

10.2 XML-Eventdatei

Über die XML-Schnittstelle des Universal Messenger kann der Versand eines Newsletters aus einem externen System, z.B. einem Content Management System aktiviert werden. Durch das Anlegen einer im folgenden beschriebenen XML-Eventdatei in dem dafür angegebenen Verzeichnis wird der Versand des Newsletters gestartet und gesteuert. Die Eventdatei enthält alle zum Versand des Newsletters notwendigen Informationen. Die zu verschickenden Inhalte können direkt in der XML-Eventdatei ("inline") oder als Referenz auf eine externe Datei oder als Referenz mit einer URL übergeben werden.

Falls eine URL angegeben ist, werden die benötigten Dateien und Bilder von dem Universal Messenger zunächst per HTTP zur Verarbeitung heruntergeladen. Dies ermöglicht eine sehr flexible Integration des Universal Messenger mit anderen Systemen, da der Zugriff per HTTP meist einfacher möglich ist als direkt über das Dateisystem.

Die benötigten Dateien und Bilder werden von dem Universal Messenger sofort beim Einlesen der Eventdatei heruntergeladen, auch wenn ein späterer Versandzeitpunkt angegeben ist oder gerade ein Newsletterversand aktiv ist und der neue Versandauftrag deswegen in eine Warteschlange gelegt werden muss. Dadurch wird sichergestellt, dass für den Newsletterversand die zum Zeitpunkt der Beauftragung aktuellen Dateien aus dem Content Management System heruntergeladen werden.

Nach der Verarbeitung eines Events verschiebt der Universal Messenger die Eventdatei in speziell angegebene Verzeichnisse, wobei zwischen erfolgreich versandten und fehlerhaften Events unterschieden wird. Die benutzten Verzeichnisse können in den Konfigurationseinstellungen festgelegt werden.

Die XML-Schnittstelle ermöglicht eine einfache Integration des Universal Messenger in ein Content Management System. Für einige Systeme werden geeignete Templates und Skripte zum Erzeugen der XML-Eventdatei bereits mitgeliefert bzw. können beim Support erfragt werden, so dass lediglich Anpassungen an die spezifischen Anforderungen notwendig sind.

Die Elemente (Tags) der XML-Eventdatei lassen sich in zwei Gruppen einteilen:

- Obligatorische Elemente
- Zusätzliche Elemente

Einige dieser XML-Elemente bzw. deren Attribute können in der Konfigurationsdatei vordefiniert werden. Wird ihr Wert zusätzlich in der Eventdatei gesetzt, so überschreibt dieser den Wert der Konfigurationsdatei.

Zusammen mit den Installationsdateien des Universal Messenger werden die Document Type Definition (DTD) der XML-Eventdatei und einige Beispiele im Verzeichnis `doc/event` ausgeliefert. Für die Erstellung eigener XML-Dateien ist es am zweckmäßigsten, diese Beispiele als Vorlage zu verwenden.

10.2.1 Variablen zur Personalisierung mit <global>

Mit der Konfigurationsoption `cmsbs.tracker.pixel` wird eine Formatierungsanweisung definiert, anhand derer ein Zählpixel in einen HTML-Newsletter eingefügt wird. Mit der Konfigurationsoption `cmsbs.tracker.link` wird analog die Formatierung aller absoluten Links des Newsletters definiert, um die Links zur Zählung der Klicks über ein Zählskript umzuleiten.

Es kann gefordert sein, innerhalb dieser Formatierungsanweisungen einige vom Newsletterversand abhängige Variablen verwenden zu können, um z.B. die Zählung für einen bestimmten Account vorzunehmen. Diese Variablen können aus dem Content Management System zusammen mit dem Versandauftrag für den Newsletter in der XML-Eventdatei übergeben werden.

Die Übergabe der Variablen erfolgt mit dem Element <global>.

10.2.1.1 <global>

Das global-Element kann mehrfach verwendet werden und umschließt jeweils die Definition einer Variablen.

Attribut	Beschreibung
special	Name der definierten Variable, die innerhalb der Formatierungsanweisung {special:variablenname} verwendet werden kann.
trim	'true', falls Leerzeichen um den Wert innerhalb des global-Elements entfernt werden sollen 'false', falls Leerzeichen um den Wert innerhalb des global-Elements in die Variable übernommen werden sollen

Beispiel:

```
<global special="etracker" trim="true">
  Accountname
</global>
```

In den Konfigurationsoptionen `cmsbs.tracker.pixel` und `cmsbs.tracker.link` kann über die Formatierungsanweisung {special:variablenname} der Inhalt der Definition übernommen werden, im Beispiel also {special:etracker}.

Häufig ist es notwendig, den Wert zur Verwendung innerhalb einer URL zu kodieren, so dass sich folgende verschachtelte Formatierungsanweisung ergibt:

```
{urlencode:{special|etracker}}
```

Als Beispiel zur Verwendung mit etracker könnten die beiden Konfigurationsoptionen folgendermaßen definiert werden:

```

cmsbs.tracker.pixel =
"" cmsbs.tracker.link =
"http://www.etracker.de/lnkcnt.php?et={urlencode:{special|etracker}}
&url={trackedurl}&lnkname={urlencode|{msgname} {now:date} -> }{trackedurlname}"

```

10.2.2 XML-Elemente und Attribute

Eine XML-Datei enthält Elemente (Tags), die den Inhalt mit einem öffnenden Tag und einem schließenden Tag umgeben. Ein Element kann Attribute haben, die dann innerhalb der spitzen Klammern des öffnenden Tags angegeben sind. Die Attribute müssen als gültiges XML kodiert werden, ein > wird also als > und < als < kodiert.

In dieser Dokumentation werden jeweils die XML-Elemente mit ihrer Bedeutung und die möglichen Attribute beschrieben.

10.2.2.1 <event>

Das event-Element umschließt den gesamten Inhalt einer XML-Eventdatei und steht damit an der Spitze der Hierarchie. Optional kann mit dem Attribut `id` ein eindeutiges Kennzeichen für diesen Event angegeben werden.

Attribut	Beschreibung
<code>id</code>	optionales eindeutiges Kennzeichen (falls keine <code>id</code> angegeben ist, wird eine zufällige <code>id</code> generiert)
<code>newsletterGroup</code>	optionale Angabe einer Newsletter-Serie, welcher der Newsletter zugeordnet werden soll
<code>skipUsedIDs</code>	<code>true</code> , wenn der Versand abgebrochen werden soll, falls es schon einen Newsletter mit der gleichen Event-ID im Archiv gibt.
<code>archiveSkipped</code>	<code>false</code> , wenn übersprungene Newsletter (<code>skippedUsedIDs</code> = " <code>true</code> ") nicht im Newsletterarchiv gespeichert werden sollen. Default: <code>true</code> : Übersprungene Newsletter werden mit dem Status "Abgebrochen" im Archiv gespeichert und können dort ggf. trotzdem versendet werden.
<code>archive</code>	<code>false</code> , wenn der Newsletter nicht im Archiv gespeichert werden soll.
<code>createdBy</code>	optionale Angabe des Nutzernamens, des Nutzers, der den Newsletterversand ausgelöst hat
<code>createdByDisplayName</code>	optionale Angabe des Anzeigenutzernamens, des Nutzers, der den Newsletterversand ausgelöst hat

cache	Wird nur von der OpenText-Anbindung verwendet und automatisch gesetzt.
reddot	Wird nur von der OpenText-Anbindung verwendet und automatisch gesetzt.

Das `event`-Element kann die folgenden Elemente enthalten, die nachfolgend beschrieben werden:

`<global>`, `<date>`, `<exportArchive>`, `<destination>`, `<data>`, `<tag>`, `<preExec>`

`<date>`

Das `date`-Element ermöglicht die Angabe eines Versandzeitpunkts für den Newsletter und ist optional. Wird dieses Element nicht angegeben, wird der Newsletter sofort versendet.

Attribut	Beschreibung
format	definiert das Datums-/Zeitformat zwischen den Tags (siehe Formatierungszeichen fuer Datum und Uhrzeit). Das Defaultformat ist "dd.MM.yyyy HH:mm:ss".

`<exportArchive>`

Das `exportArchive`-Element beinhaltet den Pfad zu einem Verzeichnis, in welchem der (personalisierte) Newsletter inkl. Attachments und eingebetteten Grafiken zusätzlich gespeichert werden soll.

Attribut	Beschreibung
uid	UID des Eintrags, der zur Personalisierung des Newsletters verwendet wird. Wird das Attribut leer gelassen oder nicht angegeben, wird der Newsletter nicht personalisiert gespeichert.
disableLinks	<code>true</code> , wenn externe Links deaktiviert werden sollen.
timestamp	Ist dieses Attribut gesetzt (Format: yyyyMMdd), wird der Newsletter in einem Unterverzeichnis "yyyyMMdd" des angegebenen Pfades gespeichert.

`<destination>`

Das `destination`-Element umschließt die Beschreibung der Ziele, an den der Newsletter versendet werden soll, d.h. die Angabe der Channels, virtuellen Channels oder einer Abfrage. Falls als Ziel eine Liste von Channels oder virtuellen Channels und eine Abfrage angegeben sind, dann wird der Newsletter an die Empfänger versendet, die in einem der Channels enthalten sind und für die zusätzlich die Anfrage zutrifft.

Das `destination`-Element hat keine Attribute und kann die folgenden Elemente enthalten:

Element	Beschreibung
channel	Channel, an den der Newsletter gerichtet werden soll
vchannel	virtueller Channel, an den der Newsletter gerichtet werden soll
query	Abfrage zur Auswahl der Empfänger, an die der Newsletter gerichtet werden soll. Es darf nur ein <code>query</code> -Element enthalten sein.

preview	Wird für die Inbox Preview mit "Litmus" verwendet.
---------	--

<channel>

Das `channel`-Element umschließt jeweils einen Channel, an den der Newsletter versendet werden soll. Als Name des Channels muss der interne Name angegeben werden. Um mehrere Channels anzugeben, kann das `channel`-Element mehrfach verwendet werden. Es hat keine Attribute.

<vchannel>

Das `vchannel`-Element umschließt jeweils einen virtuellen Channel, an den der Newsletter versendet werden soll. Als Name des virtuellen Channels muss der interne Name angegeben werden. Um mehrere virtuelle Channels anzugeben, kann das `vchannel`-Element mehrfach verwendet werden. Es hat keine Attribute.

<query>

Das `query`-Element umschließt eine Abfrage, über die die Empfänger des Newsletters eingeschränkt oder definiert werden können. Die Vergleichsoperatoren `<` und `>` müssen als gültiges XML kodiert werden, ein `>` wird also als `>` und `<` als `<` kodiert. Das `query`-Element hat keine Attribute.

<preview>

Attribut	Beschreibung
service	Angabe des Preview-Service. Aktuell ist nur "litmus" als Wert möglich.

Element	Beschreibung
baseEntry	Enthält im Attribut <code>email</code> die E-Mail-Adresse, mit welcher ein Eintrag im Universal Messenger eindeutig identifiziert werden kann, um die Vorschau zu personalisieren.

<data>

Das `data`-Element umschließt den gesamten Bereich mit der Beschreibung der Inhalte, die in dem Newsletter versendet werden sollen.

Attribut	Beschreibung
mailto	Empfänger E-Mail-Adresse mit Personalisierungsvariablen, Beispiel: <code>mailto="{otherEmailAttribute}"</code> Die Angabe ist nur notwendig, wenn die Standard-E-Mail-Adresse geändert werden soll.

Das `data`-Element kann die folgenden Elemente enthalten:

Element	Beschreibung
email	Angaben zur E-Mail, die versendet werden soll

message	Angaben zur internen Nachricht, die versendet werden soll
---------	---

<email>

Das `email`-Element umschließt die Beschreibung der E-Mail, die als Newsletter versendet werden soll.

Attribut	Beschreibung
sender	überschreibt optional die in der Konfigurationsdatei voreingestellte Absenderkennung (achten Sie auf das korrekte E-Mail-Adressformat: "Max Mustermann <max.mustermann@acme.com>")
replyto	überschreibt optional die in der Konfigurationsdatei voreingestellte Reply-To Adresse (achten Sie auf das korrekte E-Mail-Adressformat: "Max Mustermann <max.mustermann@acme.com>")
envelopeFrom	überschreibt optional die in der Konfigurationsdatei voreingestellte Envelope-From Adresse (achten Sie auf das korrekte E-Mail-Adressformat: "Max Mustermann <max.mustermann@acme.com>")
obeyPreferHtml	'true', falls die HTML-E-Mail nur an Abonnenten mit gesetztem Attribut 'html' gesendet werden soll
sendBothParts	Dieses Flag wird ausgewertet, wenn bei einem E-Mail-Newsletter eine Text-E-Mail und auch eine HTML-E-Mail angegeben sind. Bei 'true' wird die HTML-E-Mail als 'multipart/alternative' versendet und enthält auch den Textteil. Bei 'false' wird nur die HTML-E-Mail versendet.
baseUrl	URL, die den relativen Links vorangestellt wird.
downloadUrl	URL, von der referenzierte Dateien heruntergeladen werden. Ist dieses Attribut nicht angegeben, dann wird stattdessen das Attribut baseUrl aus dem email-Element verwendet.
trackingMode	überschreibt optional das in der Konfigurationsdatei voreingestellte Tracking. Mögliche Werte: 'personal', 'anonymous', 'mixed', 'off'. Der Modus "mixed" kann hier jedoch nur gewählt werden, wenn auch global der Modus "mixed" eingestellt ist. Siehe auch: Konfiguration des Trackings

Das `email`-Element kann die folgenden Elemente enthalten:

Element	Beschreibung
subject	Umschließt den Betreff der E-Mail
plaintext	Angaben zur Text-E-Mail
htmltext	Angaben zur HTML-E-Mail
file	Angaben zu den Attachments

<plaintext>

Das `plaintext`-Element umschließt den Bereich zur Beschreibung der Text-E-Mail, die Interpretation des Inhalts wird über das Attribut `inline` gekennzeichnet.

Attribut	Beschreibung
inline	'true', falls der zu versendende Text direkt als Inhalt des plaintext-Elements angegeben ist 'false', falls der zu versendende Text aus der mit absolutem Pfad angegebenen Datei oder der URL ausgelesen werden soll 'none', falls keine Text-E-Mail versendet werden soll
charset	Angabe des Encodings

baseUrl	URL, die den relativen Links vorangestellt wird.
downloadUrl	URL, von der referenzierte Dateien heruntergeladen werden. Ist dieses Attribut nicht angegeben, dann wird stattdessen das Attribut baseUrl aus dem email-Element verwendet.
linkTracking	false, wenn das Tracking von Klicks für diesen Newsletter deaktiviert werden soll.
renderCallback	Funktionsname eines CSE-Callbacks, der beim Rendern des Newsletters aufgerufen werden soll.


Falls der zu versendende Text von einer URL gelesen werden soll, würde sich folgende Schreibweise ergeben:

```
<plaintext inline="false">
  http://www.domain.de/newsletter.txt
</plaintext>
```

<htmltext>

Das `htmltext`-Element umschließt den Bereich zur Beschreibung der HTML-E-Mail. Falls der HTML-Quellcode einen Link auf eine externe Stylesheet-Datei enthält, wird dieses Stylesheet von dem Universal Messenger direkt in den HTML-Quellcode eingefügt, da externe Stylesheets von vielen Mailprogrammen nicht richtig interpretiert werden. Die Stylesheet-Datei wird von der mit dem Attribut `downloadUrl` angegebenen Adresse heruntergeladen. Falls dieses Attribut nicht angegeben ist, wird die Stylesheet-Datei von der mit dem Attribut `baseUrl` des email-Elements angegebenen Adresse heruntergeladen.

Attribut	Beschreibung
inline	'true', falls der zu versendende HTML-Text direkt als Inhalt des <code>htmltext</code> -Elements angegeben ist 'false', falls der zu versendende HTML-Text aus der mit absolutem Pfad angegebenen Datei oder der URL ausgelesen werden soll 'none', falls keine HTML-E-Mail versendet werden soll
embedImages	'all', falls alle Bilder in die E-Mail eingebettet werden sollen 'byPath', falls wenn alle Bilder mit relativem Link eingebettet werden sollen (Standardeinstellung) 'none', falls keine Bilder eingebettet werden sollen Diese Einstellung kann für einzelne Bilder im <code></code> -Tag (und andere Tags mit Image-Referenz) mit Hilfe des Attributs <code>data-cmsbs-embed</code> überschrieben werden: <ul style="list-style-type: none"> Bild soll immer eingebettet werden: <code></code> Bild soll nie eingebettet werden: <code></code>
baseUrl	URL, die den relativen Links vorangestellt wird.
downloadUrl	URL, von der referenzierte Dateien heruntergeladen werden. Ist dieses Attribut nicht angegeben, dann wird stattdessen das Attribut baseUrl aus dem email-Element verwendet.
charset	Angabe des Encodings
linkTracking	false, wenn das Tracking von Klicks für diesen Newsletter deaktiviert werden soll.
viewTracking	false, wenn das Tracking von Öffnungen für diesen Newsletter deaktiviert werden soll.
renderCallback	Funktionsname eines CSE-Callbacks, der beim Rendern des Newsletters aufgerufen werden soll.

restProxyUrl	<p>Url des öffentlich erreichbaren REST-Proxys.</p> <p>Ist dieses Attribut gesetzt, werden die referenzierten Dateien im jeweils aktuellen Zustand im Universal Messenger gesichert und die URLs in der E-Mail automatisch so umgeschrieben, dass die Dateien über den REST-Proxy ausgeliefert werden. D.h. sie können auf einem System liegen, das nicht direkt über das Internet erreichbar sein muss. Sie sind außerdem jederzeit im Newsletterarchiv abrufbar und Empfänger von Newslettern können ältere Newsletter öffnen, ohne dass referenzierte Dateien inzwischen nicht mehr vorhanden sind.</p> <div>  Für nicht eingebettete Grafiken und andere Dateien empfehlen wir die Auslieferung über einen REST-Proxy. Mehr Informationen zur Installation des REST-Proxys finden Sie im <i>Handbuch für Administratoren</i> unter "REST-Proxy". </div>
--------------	--

Falls der zu versendende HTML-Text von einer URL gelesen werden soll, würde sich folgende Schreibweise ergeben:

```
<htmltext inline="false">
  http://www.domain.de/newsletter.html
</htmltext>
```

Wenn zusätzlich das Attribut `downloadUrl` angegeben wird, werden die im HTML-Text referenzierten Dateien von der hier angegebenen URL heruntergeladen.

`<file>`

Das `file`-Element kann mehrfach innerhalb eines `email`-Elements vorkommen und umschließt jeweils einen absoluten Dateinamen oder eine URL, dessen Inhalt als Attachment an die E-Mail angehängt werden soll. Neben den hier angegebenen Dateien werden der HTML-E-Mail von dem Universal Messenger automatisch alle Dateien angehängt, die im HTML-Text verlinkt sind und auch im `data`-Verzeichnis abgelegt wurden.

Wird das `file`-Element innerhalb einer Text-E-Mail verwendet, dann wird diese E-Mail durch das Anhängen des Attachments als MIME Multipart-Message versendet.

Falls das `file`-Element mehrfach verwendet wird und jeweils eine URL angegeben wird, müssen die verwendeten Dateinamen eindeutig sein, da alle heruntergeladenen Dateien in einem Verzeichnis abgelegt werden.

Attribut	Beschreibung
name	Definiert optional einen 'content-name' für das Attachment (im Standardfall der Dateiname), auf den in der E-Mail über "cid:content-name" Bezug genommen werden kann.

<message>

Das `message`-Element umschließt den internen Namen einer Benachrichtigung, aus der der Text für den Versand des Newsletters und alle anderen Angaben zu Betreff, Absender usw. entnommen werden sollen. Wenn ein `message`-Element angegeben ist, werden evtl. zusätzlich enthaltene `email`-Elemente ignoriert.

<tag>

Mit dem `tag`-Element können optional Tags für den Newsletter vergeben werden. Um mehrere Tags anzugeben, kann das `tag`-Element mehrfach verwendet werden.

<preExec>

Innerhalb des `<event>`-Tags können CSE-Funktionen mit dem `preExec`-Tag aufgerufen werden. Sie werden in der Reihenfolge abgearbeitet wie sie in der XML-Datei stehen. Wenn dabei Exceptions auftreten, werden sie geloggt und der Versand wird an dieser Stelle abgebrochen. Die CSE-Funktionen werden gleich zu Beginn des Versands ausgeführt, unmittelbar vor dem `NewsletterProgressCallback.started()` und auch insbesondere bevor der Download des HTML-Newsletters oder die Materialisierung der Empfängermenge beginnt.

Attribut	Beschreibung
function	Name der CSE-Funktion, die aufgerufen werden soll. Sie wird im custom-Scope ausgeführt.
onResume	<code>false</code> , wenn die CSE-Funktion beim Fortsetzen eines Newsletterversands nicht erneut ausgeführt werden soll.
onResend	<code>false</code> , wenn die CSE-Funktion beim nachträglichen Versenden eines Newsletters an einzelne Einträge nicht erneut ausgeführt werden soll.

Element	Beschreibung
param	Parameter für die CSE-Funktion. Mehrere sind möglich.

Beispiel

```
<event>
  ...
  <preExec function="newsletter_preExecCallback1">
    <param>AAA</param>
    <param>BBB</param>
  </preExec>
  <preExec function="newsletter_preExecCallback2">
    <param>CCC</param>
  </preExec>
</event>
```

Damit werden die folgenden zwei Funktionsaufrufe ausgeführt:

```
newsletter_preExecCallback1("AAA", "BBB");
newsletter_preExecCallback2("CCC");
```

10.2.3 Übergabe der Eventdatei via REST-Aufruf

Neben den diversen anderen Möglichkeiten, einen Newsletterversand zu starten, kann die Eventdatei über einen REST-Aufruf per HTTP POST an den Universal Messenger übergeben werden.

Zur Entwicklung lässt sich die Schnittstelle mit folgendem Kommandozeilenaufruf testen:

```
curl
http://localhost:8080/cmsbs/rest/de.pinuts.cmsbs.restsend.EventFile/?open=<cmsbs.open>
--data-binary @newsletter.xml --header "Content-Type: text/xml; charset=UTF-8"
```

Dieser Aufruf übergibt die Datei `newsletter.xml` im richtigen Format an die URL. Dabei sind folgende Punkte zu beachten:

- Im Parameter `open` muss das `cmsbs.open`-Passwort zur Authorisierung angegeben werden.
- Der `Content-Type` muss `text/xml` lauten.
- Das `charset` muss korrekt angegeben sein.



Der Controller `de.pinuts.cmsbs.restsend.EventFile` sollte im Allgemeinen NICHT im Internet verfügbar sein und sollte entsprechend im REST-Proxy auch nicht freigeschaltet werden.

In Cloud-basierten Installationen hingegen, in denen der Universal Messenger nur aus dem Internet erreichbar ist, muss o.g. Controller im REST-Proxy freigeschaltet werden, wenn die Funktion genutzt werden soll.

Details zur Installation und Konfiguration des REST-Proxys finden Sie im *Handbuch für Administratoren* im Abschnitt REST-Proxy.

10.2.3.1 Status eines Newsletterversands abfragen

Über den folgenden REST-Aufruf kann der Status eines Newsletterversands abgefragt werden:

```
curl
http://localhost:8080/cmsbs/rest/de.pinuts.cmsbs.restapi.NewsletterQueue/status/<id>?open=<cmsbs
```

Dieser Aufruf fragt den Status des Newsletterversands zu der angegebenen `id` ab. Dabei sind folgende Punkte zu beachten:

- Im Parameter `open` muss das `cmsbs.open`-Passwort zur Authorisierung angegeben werden.
- Als `id` muss der Wert angegeben werden, der beim Start des Newsletterversands als Event-ID ("id"-Attribut aus dem `<event>`-Element des Eventfiles) gesetzt wurde.

Falls die angegebene Event-ID nicht gefunden werden kann, wird z.B. die folgende Rückgabe mit dem HTTP-Status-Code 404 gesendet:

```
{
  eventId: "1234",
  error: "NOT_FOUND"
}
```

Falls der Versand bereits beendet ist, wird z.B. die folgende Rückgabe mit dem HTTP-Status-Code 200 gesendet:

```
{
  eventId: "2f326443acd48cf3ddbe",
  oid: "2333276",
  sendState: 5,
  sendDate: "2018-11-12|13:45:24",
  contacted: 54,
  counted: 54,
  notContacted: 0,
  inQueue: false,
  isFailed: false,
  isFinished: true,
  isStopped: false
}
```

Falls der Versand gestoppt wurde, wird z.B. die folgende Rückgabe mit dem HTTP-Status-Code 200 gesendet:

```
{
  eventId: "2af953e71a3ded9d0ae7",
  oid: "2340755",
  sendState: -2,
  sendDate: "2018-11-12|13:49:35",
  contacted: 0,
  counted: 608,
  notContacted: 0,
  inQueue: false,
  isFailed: false,
  isFinished: false,
  isStopped: true
}
```

Falls der Versand noch in der Queue wartet, wird z.B. die folgende Rückgabe mit dem HTTP-Status-Code 200 gesendet:

```
{
  eventId: "test-123456",
  inQueue: true,
  locked: true,
  sendDate: null
}
```

11 Anhang

11.1 Formatierungszeichen für Datum und Uhrzeit

Die Definition und Ausgabe von Datum und Uhrzeit erfolgt über Formatierungszeichen. Das Datumsformat wird als Pattern angegeben, die folgenden Buchstaben dienen als Platzhalter:

	Bedeutung	Art	Beispiel
G	Zeitalter	Text	AD
y	Jahr	Nummer	1996
M	Monat im Jahr	Text & Nummer	July oder 07
d	Tag im Monat	Nummer	10
h	Stunde am/pm (1-12)	Nummer	12
H	Stunde (0-23)	Nummer	0
m	Minute	Nummer	30
s	Sekunde	Nummer	55
S	Millisekunde	Nummer	978
E	Wochentag	Text	Tuesday
D	Tag im Jahr	Nummer	189
F	Wochentag im Monat	Nummer	2 (2nd Wed in July)
w	Woche des Jahres	Nummer	27
W	Woche des Monats	Nummer	2
a	am/pm Angabe	Text	PM
k	Stunde des Tages (1-24)	Nummer	24
K	Stunde des Tages am/pm (0-11)	Nummer	0
z	Zeitzone	Text	Pacific Standard Time
'	Textescape	Begrenzung	'Uhr'
''	Hochkomma	Literal	'

11.1.1 Beispiele

"yyy.MM.dd G 'at' hh:mm:ss z"	1996.07.10 AD at 15:08:56 PDT
"EEE, MMM d, ''yy"	Wed, July 10, '96
"h:mm a"	12:08 PM
"hh 'o'clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:00 PM, PST
"yyyyy.MMMMM.dd GGG hh:mm aaa"	1996.July.10 AD 12:08 PM

11.1.2 Datumsformat der XML-Eventdatei

Wenn das Content Management System das Datum nicht im Standardformat formatiert ausgeben kann, dann muss bei der Ausgabe des `<date>` Tags die Datenformatierung als Attribut angegeben werden.

Beispiel:

```
<date format="dd.MM.yyyy HH.mm.ss">12.10.1976 13:25:00</date>
```

11.2 Reguläre Ausdrücke

Im Folgenden ein Auszug aus der Anleitung der benutzten Bibliothek für reguläre Ausdrücke. Für eine Beschreibung der regulären Ausdrücke verweisen wir auf die umfangreiche Literatur und zahlreiche Benutzeranleitungen zu diesem Thema.

1. Zeichen

x	erkennt das unicode Zeichen 'x'
\	wird als 'escape'-Zeichen benutzt
\\	erkennt ein einzelnes '\'
\Onnn	erkennt das angegebene octale Zeichen
\xhh	erkennt das angegebene 8-bit Hexadezimalzeichen
\uhhhh	erkennt das angegebene 16-bit Hexadezimalzeichen
\t	erkennt ein Tabulatorzeichen
\n	erkennt einen Zeilenvorschub
\r	erkennt einen Wagenrücklauf
\f	erkennt einen Seitenvorschub

2. Zeichenklassen

[abc]	einfache Zeichenklasse
[a-zA-Z]	Zeichenklasse mit Abschnitten
[^abc]	negierte Zeichenklasse

3. POSIX Zeichenklassen

[::alnum:]	alphanumerische Zeichen
[::alpha:]	Buchstaben
[::blank:]	Leerzeichen oder Tabulatoren
[::cntrl:]	Kontrollzeichen
[::digit:]	Zahlenzeichen
[::graph:]	druckbare und sichtbare Zeichen (z.B. ist ein Leerzeichen druckbar, aber unsichtbar; ein 'a' ist dagegen druckbar und sichtbar)
[::lower:]	kleingedruckte, alphabetische Zeichen
[::print:]	druckbare Zeichen (Zeichen die keine Kontrollzeichen sind)
[::punct:]	Interpunktionszeichen (Zeichen die keine Buchstaben, Zahlen, Kontrollzeichen oder Leerzeichen sind)
[::space:]	Leerzeichen (auch Tabulatoren, Seitenvorschub, etc.)
[::upper:]	großgedruckte, alphabetische Zeichen
[::xdigit:]	hexadezimale Zahlenzeichen

4. Vordefinierte Klassen

.	erkennt jedes Zeichen außer dem Zeilenvorschub
\w	erkennt ein Wort (alphanumerisch und '_')
\W	alles was nicht von \w erkannt wird
\s	erkennt ein Leerzeichen (Whitespace)
\S	alles was nicht von \s erkannt wird
\d	ein Zahlenzeichen
\D	alles was nicht von \d erkannt wird

5. Begrenzungen

^	erkennt den Zeilenanfang
\$	erkennt das Zeilenende
\b	erkennt eine Wortgrenze
\B	alles was nicht von \b erkannt wird

6. Einschließende Optionen (Greedy Closures)

A^*	erkennt A mindestens Null-mal (gefräßig)
A^+	erkennt A mindestens einmal (gefräßig)
$A^?$	erkennt A Null- oder einmal (gefräßig)
$A\{n\}$	erkennt A genau n-mal (gefräßig)
$A\{n,\}$	erkennt A mindestens n-mal (gefräßig)
$A\{n,m\}$	erkennt A mindestens n, aber maximal m-mal (gefräßig)

7. Optionen (Reluctant Closures)

$A^*?$	erkennt A mindestens Null-mal (zurückhaltend)
$A^+?$	erkennt A mindestens einmal (zurückhaltend)
$A^??$	erkennt A Null- oder einmal (zurückhaltend)

8. Logische Operatoren

AB	erkennt A gefolgt von B
$A B$	erkennt entweder A oder B
(A)	gruppiert Unterausdrücke

9. Rückwärtsreferenzen

$\backslash 1$	Rückwärtsreferenz auf den ersten geklammerten Unterausdruck
$\backslash 2$	Rückwärtsreferenz auf den zweiten geklammerten Unterausdruck
$\backslash 3$	Rückwärtsreferenz auf den dritten geklammerten Unterausdruck
$\backslash 4$	Rückwärtsreferenz auf den vierten geklammerten Unterausdruck
$\backslash 5$	Rückwärtsreferenz auf den fünften geklammerten Unterausdruck
$\backslash 6$	Rückwärtsreferenz auf den sechsten geklammerten Unterausdruck
$\backslash 7$	Rückwärtsreferenz auf den siebten geklammerten Unterausdruck
$\backslash 8$	Rückwärtsreferenz auf den achten geklammerten Unterausdruck
$\backslash 9$	Rückwärtsreferenz auf den neunten geklammerten Unterausdruck

11.3 Tracking-Client Integration

Damit zu versendende Newsletter das Tracking verwenden können, muss die Datei `cmsbs.properties` des Newsletter-Systems angepasst werden:

```

cmsbs.tracker.pixel = "<img src='http://your-public-host/cmsbs-tracking/nt?t={msgid}' />"
cmsbs.tracker.link =
"http://your-public-host/cmsbs-tracking/nl?t={msgid}&d={trackedurl}&n={trackedurltitle}&h={trackedurl:sha1}"

```

Die beiden Zeilen konfigurieren das Zählpixel für das View-Tracking und die Link-Weiterleitung für das Click-Tracking. Tragen Sie hier die passenden Adressen Ihrer oben installierten Umgebung ein.

Variable	Beschreibung
msgid	Eindeutige Kombination aus dem Newsletter und dem Empfänger des Newsletters. Ermöglicht eine Zuordnung zwischen den im Newsletter durchgeführten Aktionen und dem im Newsletter-Archiv abgelegten Newsletter, sowie dem einzelnen Benutzer-Eintrag. Die <code>msgid</code> wird automatisch generiert.
trackedurl	URL-encodete Ziel-Adresse, die vom Empfänger aufgerufen werden soll. Entspricht dem href-Wert von im Newsletter-Quellcode enthaltenen Links, bevor das Newsletter-System den Versand begonnen hat. Die <code>trackedurl</code> wird automatisch generiert.
trackedurl:sha1	SHA1-Hashwert der Ziel-URL zum Schutz gegen Missbrauch des Linktrackers.
trackedurltitle	URL-encodete Beschreibung für die Ziel-Adresse, die im Newsletter-Archiv anstelle der URL in der Statistik aufgeführt wird. Der <code>trackedurltitle</code> wird automatisch aus dem title-Attribute des Anchor-Elements entnommen, kann allerdings auch durch eigene Zeichenketten ersetzt werden (die dann manuell URL-encodet werden müssen).

Ob das Tracking für einen Newsletter verwendet werden soll, kann über die XML-Event-Datei gesteuert werden:

```

<plaintext inline="false" viewTracking="false"
clickTracking="false">http://www.domain.de/newsletter.txt</plaintext>
<htmltext inline="false" viewTracking="true"
clickTracking="true">http://www.domain.de/newsletter.html</htmltext>

```

Conversions und Unsubscriptions können direkt über die installierten Komponenten geführt werden, da diese ähnlich wie das Zählpixel eine Grafik ausgeben und somit als verstecktes Bild in Seiten/Formulare eingefügt werden können. Wir empfehlen aber, den Code der installierten Komponenten als Beispiel für eine direkte Integration in die Applikation (z.B. durchgeführte Bestellungen, erfolgte Newsletter-Abmeldungen) zu verwenden.

12 Handbuch für Entwickler 7.52