

MyTISM - Ein Datenbank- und Anwendungs-Framework

Inhaltsverzeichnis

Erste Schritte mit MyTISM	2
Einrichtung einer kompilierfähigen MyTISM-Umgebung unter Windows (15.07.2005)	2
Kompilieren eines MyTISM-Projekts	4
Starten eines MyTISM-Servers	5
Umziehen eines syncenden MyTISM-Servers auf eine andere Hardware	5
Aufsetzen eines syncenden MyTISM-Servers aus einem Backup des autoritativen Servers	6
Starten eines MyTISM-Clients (SOLSTICE)	7
Konfiguration des Servers via mytism.ini	8
Die ".checked*" -Dateien	14
Die ".init*" -Dateien	14
Services	15
Benachrichtigungen	21
Grundlagen	22
Schnelle Hilfe :-)	23
Alarmsystem und Benachrichtigungssystem	24
Vorbereitung und Konfiguration	25
Benachrichtigungssystem-Lizenz einspielen	25
Benachrichtigungssystem aktivieren	25
Benachrichtigungssystem deaktivieren	25
Erzeugung / Loggen von Benachrichtigungsversendungen	28
Versenden veralteter Benachrichtigungen verhindern	28
Verschlüsselung und digitale Signatur	28
e-Mail-Einstellungen	31
Solstice-Einstellungen	37
Einstellungen für die Benutzer	38
Manuelles Versenden von Benachrichtigungen	41
Programmatisches manuelles Versenden von Benachrichtigungen	42
Benachrichtigungen einsehen	44
Wer kann Benachrichtigungen erhalten?	45
Analyse von Problemen und Fehlerbehebung	46
Versendung von Benachrichtigungen kontrollieren	46
Benachrichtigungen für spezifische Objekte finden	47
Checkliste mögliche Fehlerquellen	48
Tipps und Tricks	50
Testbenachrichtigungen senden	50
Navigationsbaum	51
Aussehen und Position von Elementen	52
Sichtbarkeit von Elementen	53

Der Ordner "Alle GUI-Benutzer" und seine Konfiguration	55
Konfigurationsparameter	56
Beispiel-Konfigurationen	57
Rechteverwaltung	59
Grundlagen	60
Benutzer	60
Gruppen	61
Voreinstellungen für Benutzer und Gruppen	62
BO-Masken	62
Rechtezuweisungen	66
Rechte vergeben	70
Negativliste - Selektiv verbieten	70
Positivliste - Selektiv erlauben	71
Zusammenhang mit der Methode B0#isReadOnly(AttributeI)	72
Plugins - Zuschaltbare Komponenten im Client	74
Eingabefeld für Befehle in der Aktionsleiste einblenden	74
Server-Gesundheitsanzeige einblenden	74
Fehlerbehebung	76
Der Server startet nicht wegen angeblich fehlenden Hostnamens	77
Situation	77
Fehlerbehebung	77
Der Integrity-Check bricht ab wegen fehlender Ressourcen	78
Fehlerbehebung	78
Doppelte IDs in der Datenbank korrigieren	79
Situation	79
Fehlerbehebung	79
Wie starte ich MyTISM durch, ohne dass der Benutzer es mitbekommt (restart fast / restart silent)?	81
Situation	81
Lösung	81

MyTISM ist ein plattformunabhängiges, objektorientiertes, dezentrales, multiuserfähiges, individuell anpassbares und quelloffenes 3-Tier-Datenbank- und Anwendungs-Framework incl. GUI und Web-Application-Server, entwickelt und betreut von OAshi S.à r.l.

In diesem Handbuch finden Sie alle Informationen, die Sie für das Aufsetzen, Warten und Administrieren eines MyTISM-Systems benötigen.



Beachten Sie bitte, dass sich dieses Dokument noch im Aufbaustadium befindet und noch grosse Lücken aufweist, die wir natürlich nach und nach füllen werden.

Bei Fragen, Problemen oder Anregungen, sei es bzgl. MyTISM selber oder dieser Dokumentation, wenden Sie sich bitte an uns; Kontaktinfos finden Sie im WWW unter <https://mytism.com/#contact>.

Erste Schritte mit MyTISM

Einrichtung einer kompilierfähigen MyTISM-Umgebung unter Windows (15.07.2005)

Im folgenden Text wird auf die einzelnen Schritte der Installation des PostgreSQL-Datenbank-Servers und des Java-Developer-Kits unter Windows eingegangen.

PostgreSQL



Die Installation sollte als angemeldeter Administrator durchgeführt werden, da Installationsprobleme mit Benutzer-Konten berichtet wurden, die der Administrator-Gruppe "nur zugeordnet" waren.

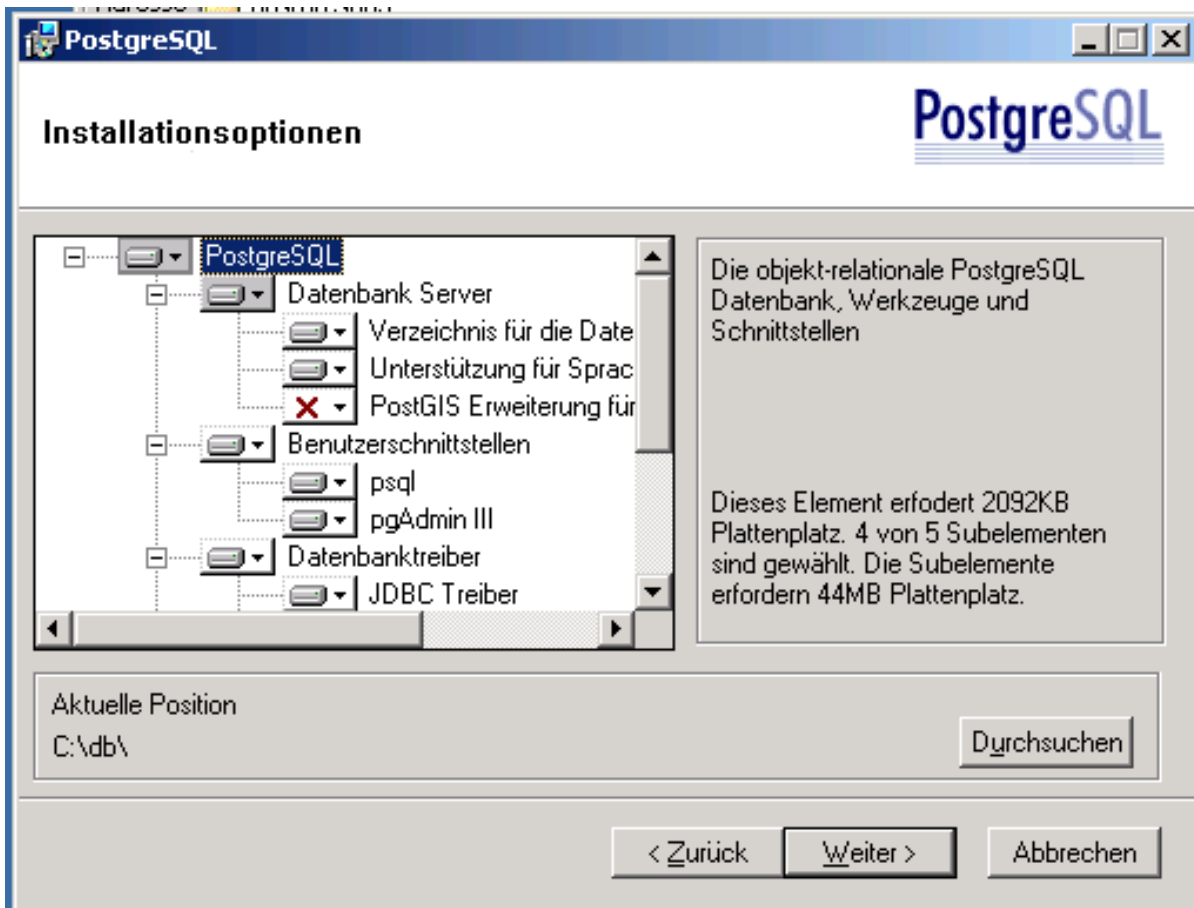
Am Anfang steht der Download des Installations-Archivs. Dieses kann von <http://www.postgresql.org/ftp/win32> bezogen werden. Sollte der Link irgendwann nicht mehr stimmen, gehe man direkt auf <http://www.postgresql.org> und klicke sich bis zum Windows-Download durch. Hier wählt man das ZIP-Archiv `postgresql-8.0.3.zip` zum Download aus.

Das heruntergeladene Archiv muss entpackt werden. U.a. liegen dann zwei Dateien auf der Festplatte, die mit `postgres` beginnen und die Endung `.msi` haben - hiervon ruft man die kleinere Datei auf zum Starten der PostgreSQL-Installation.

Die Installationsabfolge ist weitestgehend unspektakulär, so dass nur auf einige "Spezialitäten" eingegangen wird.

Zu Beginn wählt man lediglich die Installationssprache aus, mit der man durch den weiteren Installationsverlauf geleitet wird.

Im Dialog "Installations-Optionen" stellt man ein, dass alle Sprachen installiert werden sollen und setzt das Installationsverzeichnis auf `c:\db`



Im Dialog "Dienste-Konfiguration" ändert man den Namen des Dienstes auf `postgres` und vergibt KEIN Passwort. Den folgenden Dialog, in dem man gefragt wird ob das fehlende Benutzerkonto angelegt werden soll, bestätigt man - das anschliessend angezeigte Passwort kann man sich notieren, muss man aber nicht.

Im Dialog "Datenbank-Cluster initialisieren" würde man theoretisch das Encoding auf `UTF-8` umstellen. Da es damit aber unter Windows noch ein paar Problemchen gibt, lässt man das einfach auf `SQL-ASCII` stehen.



Wenn man auf der Kommandozeile per `createdb -U postgres -E UTF-8 DBNAME` eine neue Datenbank anlegt, muss man jetzt aber daran denken, das Encoding auf `UTF-8` zu stellen!

Im gleichen Dialog setzt man das Passwort auf `postgres`.

Alle weiteren Dialoge kann man einfach bestätigen.

Nachdem die Installation abgeschlossen ist, muss noch eine Einstellung in der Datei `c:\db\data\pg_hba.conf` vorgenommen werden: Am Ende der Datei in der nicht auskommentierten Zeile den Text `md5` durch `trust` ersetzen. An dieser Stelle wird auch ersichtlich, warum man während der Installation ein doch vermeintlich schwaches Passwort wählen konnte - es werden nämlich eh nur Verbindungen direkt von der lokalen Maschine (127.0.0.1) akzeptiert und durch die Angabe von `trust` erspart man sich die Passwort-Abfrage.

Folgende Tuning-Massnahmen an der Datei `c:\db\data\postgresql.conf` sind nicht zwingend nötig, bringen aber doch einiges an Performance-Gewinn:

```
fsync = true
wal_buffers = 2000
commit_delay = 10000
commit_siblings = 500
shared_buffers = 512MB
```

Damit diese und obige Änderung wirksam werden, muss der PostgreSQL-Server "durchgestartet" werden.

```
net stop postgres
net start postgres
```

Java Developer Kit (JDK)

Auch hier muss man sich erst einmal das Installations-Archiv von <http://java.com> besorgen (dort dann in der Download-Sektion die aktuellste JDK als "Windows-Offline-Version" herunterladen). Die Installation wird per Doppelklick gestartet und verläuft recht unspektakulär. Man sollte sich lediglich das Installationsverzeichnis merken (meist etwas in der Art von `c:\Programme\Java\jdkxxxx`).

Kompilieren eines MyTISM-Projekts

Hierfür werden natürlich die Sourcen benötigt, die man mittels des Programms [SmartCVS](#) aus dem CVS-Repository abrufen kann.

Auschecken aus dem CVS-Repository mit SmartCVS

Zunächst muss man im Repository Manager das Repository definieren. Dazu verwendet man folgende Daten:

Access Method	sserver
Server Name	cv.s.mytism.de
Repository Path	cv.s

Zum Auschecken aus dem CVS per [SmartCVS](#) wählt man nun im Menü "Project" den Eintrag "Checkout" und gelangt zu einem Wizard. Dort trägt man folgende Werte ein:

Repository	{das eben definierte Repository}
Module(s)	nrx
Local Directory	{das Verzeichnis, in das ausgecheckt werden soll; ein Verzeichnis nrx wird dort automatisch angelegt}
Checkout options	Default (keep sticky)
Project Name:	nrx

Text File Encoding	Cp1252
Text File Encoding in Repository	ASCII

Zum Kompilieren wechselt man unterhalb des nrx-Verzeichnisses in das entsprechende Projekt-Verzeichnis und startet die Kompilierung mit folgendem Befehl:

```
j -ant
```

Das Ergebnis der Kompilierung liegt im Build-Verzeichnis (wo dieses zu finden ist, steht im Build-File des jeweiligen Projektes in der Variablen `build.dir`).

Sofern nicht bereits geschehen muss in PostgreSQL natürlich noch eine Datenbank angelegt werden. Dies macht man mithilfe des zentralen mytism-Scripts im Build-Verzeichnis.

Starten eines MyTISM-Servers

```
/mytism init_db
```

Sofern alles klappt ist unter Linux mit dem Aufruf von

```
/mytism start
```

aus dem Build-Verzeichnis heraus alles erledigt.



Unmittelbar nach Aufruf von `./mytism start` kommt die Fehlermeldung `"bash: ./server: Keine Berechtigung"`? → Die Datei `"mytism"` ist aufgrund des fehlenden Executable-Flags nicht ausführbar. Dieses setzt man man unter Linux mittels `chmod 755 DATEINAME`

Umziehen eines syncenden MyTISM-Servers auf eine andere Hardware

FIXME



Die BTs müssen in jedem Fall vollständig bleiben und dürfen auf keinen Fall geflusht werden, wenn man eine Instanz umzieht. Grund ist, dass der SyncChecker sonst nicht prüfen kann, welche Transaktionen evtl. lokal noch aufgrund von Timing-Problemen beim Syncen (lang dauernde Speichervorgänge auf dem Server, von dem gesynct wird, die später vom SyncChecker nachgezogen werden) fehlen.

Aufsetzen eines syncenden MyTISM-Servers aus einem Backup des autoritativen Servers

Neu-Einrichtung eines syncenden Knotens

1. Auf dem (existierenden) autoritativen Server:
 - a. Neue BN für den neuen, syncenden Knoten anlegen und speichern
 - i. Name ist Pflicht
 - ii. ggf. Beschreibung eingeben
 - iii. Synchronisationskonto (Benutzer) auswählen oder falls notwendig noch anlegen
 - b. `.init-syncaccount`-Datei aus dem BN-Formular der neuen BN exportieren (der Name `.init-syncaccount` muss leider im Dateidialog noch Mal von Hand exakt so eingegeben werden)
 - c. Diese `.init-syncaccount`-Datei auf den synchronisierenden Server ins Projektverzeichnis kopieren.
 - d. Backup ziehen `./mytism backup` oder das letzte komplette Nightly Backup verwenden, wenn die BN dann schon in der DB existierte.
 - e. Backup mittels `./mytism prepare-backup-for-node-setup <Name des Unterverzeichnisses des zu verwendenden Backups in "backups">` für das Neuaufsetzen eines synchronisierenden Servers präparieren.
 - f. Das so präparierte Backup (gewähltes Backup-Unterverzeichnis mit Suffix `"_for_node_setup"`) zum synchronisierenden Server kopieren.



Beim Exportieren der `.init-syncaccount`-Datei wird für den ausgewählten Benutzer ein Passwort generiert und dort eingetragen; sollte der selbe Benutzer für mehrere Nodes für die Synchronisation benutzt werden (FIXME Ist das erlaubt? Oder sollte das nicht gemacht werden?) werden die Anmeldedaten von evtl. schon bestehenden `.init-syncaccount`-Dateien anderer Nodes damit ungültig und müssen manuell aktualisiert werden.

1. Synchronisierender Server:
 - a. Backup einspielen.
 - b. (Nur notwendig falls diese auf dem syncenden Server schon existieren:) `.checked-*`-Dateien löschen `rm .checked-*` (WICHTIG! Hier ausnahmsweise auch die `.checked-firstnodestart` löschen!)
 - c. (Nur notwendig falls diese auf dem syncenden Server schon existiert:) `.init-keygen` löschen `rm .init-keygen`, dies leert die lokale bi-Tabelle.
 - d. (Nur notwendig falls diese auf dem syncenden Server schon existiert:) `.init-streamcopy` löschen `rm .init-streamcopy`, dadurch werden die fehlenden BLOBs vom Hauptserver übertragen.
 - e. Server starten `./mytism start_mytism`
 - f. Eine kleine Änderung speichern, damit der Sync das nächste Mal leichter den lokalen

Ansatz finden kann.



Wenn die Meldung `Server has no logs from us yet, so we can't check.` im Server-Log auftaucht, wurde der letzte Punkt nicht beachtet und keine Änderung gemacht.

Erneutes Aufsetzen eines bestehenden Knotens



Beim Neu-Aufsetzen sollte man darauf achten, dass die lokalen Daten alle zum Hauptserver synchronisiert wurden, bevor man das Backup vom Hauptserver zieht und damit den synchronisierenden Server neu aufsetzt.

1. Autoritativer Server:

- a. Backup ziehen `./mytism backup` oder das letzte komplette Nightly Backup verwenden, wenn die Daten des synchronisierenden Servers zu dieser Zeit bereits alle raus gesyncet worden waren.
- b. Das Backup zum synchronisierenden Server kopieren.

2. Synchronisierender Server:

- a. Backup einspielen.
- b. `.checked-*`-Dateien löschen `rm .checked-*` (WICHTIG! Auch die `.checked-firstnodestart` löschen!)
- c. `.init-keygen` löschen `rm .init-keygen`, dies leert die lokale bi-Tabelle.
- d. `.init-streamcopy` löschen `rm .init-streamcopy`, dies initiiert einen sauberen Abgleich der BLOBs im `filesRoot`-Verzeichnis mit dem autoritativen Server.
- e. Server starten `./mytism start_mytism`
- f. Eine kleine Änderung speichern, damit der Sync das nächste Mal leichter den lokalen Ansatz finden kann.



Wenn die Meldung `Server has no logs from us yet, so we can't check.` im Server-Log auftaucht, wurde Punkt 2.e nicht beachtet.

Starten eines MyTISM-Clients (SOLSTICE)

Der Client lässt sich per Kommandozeile aus dem Build-Verzeichnis starten

```
java -jar deploy/XXX-Client.jar localhost
```

oder per JavaWebStart

```
/usr/lib/java/jre/javaws/javaws "http://localhost:8080/deploy/" // statt localhost  
kann auch eine IP-Adresse angegeben werden
```

Das Logfile bzw. Meldungen des Client werden beim Start von der Kommandozeile in der jeweiligen Shell ausgegeben, in der der Client gestartet wurde. Ausserdem wird im Temp-Verzeichnis (unter Linux /tmp/) eine Log-Datei namens "client-log.txt" angelegt.



Unter Windows-Systemen wird die heruntergeladene Anwendung (Datei mit der Endung ".jnlp") im Profil gespeichert, was bei Systemen, deren Profil auf einem zentralen Server liegt den An- bzw. Abmelde-Prozess verlangsamen kann. Im JavaWebStart lässt sich der Speicherort der heruntergeladenen Anwendung festlegen. Hier wählt man dann einen geeigneteren Speicherort aus.

Konfiguration des Servers via mytism.ini

Die Datei `mytism.ini` befindet sich im Projektverzeichnis ("Punkt-Verzeichnis"). Die Einstellungen werden über den Abschnitt `[DBMan]` vorgenommen. Es können weitere Unterabschnitte definiert werden.

Es folgt eine Tabelle mit Erklärungen zu den möglichen Einstellungsvariablen. In der ersten Spalte steht der Variablenname, dem ein Wert zugewiesen werden kann. In der zweiten Spalte steht der Default bzw. falls in Klammern ein Beispiel.

<code>schemaFile</code>	<code>(./demo//schema/schema.xml)</code>	Absoluter Pfad zur <code>schema.xml</code>
<code>driver</code>	<code>org.postgresql.Driver</code>	fully qualified Klassenname des Datenbanktreibers
<code>url</code>	<code>(jdbc:postgresql://localhost:5432/demo)</code>	URL für die Verbindung zur Datenbank
<code>user</code>	<code>postgres</code>	Benutzername für die Verbindung zur Datenbank
<code>pass</code>	<code>postgres</code>	Passwort für die Verbindung zur Datenbank
<code>filesRoot</code>	<code>(./demo/filesRoot)</code>	Speicherort im Dateisystem für die BLOB
<code>fileVault</code>	<code>de.ipcon.db.blob.ServerFileVault</code>	fully qualified Klassenname des File-Vault, der die BLOBs managed
<code>fileVaultCfg</code>	<code>FileVault</code>	Name für den Abschnitt mit weiterer Konfiguration für den File-Vault
<code>persistenceMgr</code>	<code>de.ipcon.db.CastorPersistenceManager</code>	fully qualified Klassenname des Persistenz-Managers
<code>persistenceMgrCfg</code>	<code>Castor</code>	Name für den Abschnitt mit weiterer Konfiguration für den Persistenz-Manager

keyGenerator	de.ipcon.db.JDBCKeyGenerator	fully qualified Klassenname des Key-Generators
keyGeneratorCfg	KeyGen	Name für den Abschnitt mit weiterer Konfiguration für den Key-Generator
protocolDriver	de.ipcon.db.SocketProtocolServer	fully qualified Klassenname des Protokoll-Treibers
protocolDriverCfg	SocketProtocol	Name für den Abschnitt mit weiterer Konfiguration für den Protokoll-Treiber
cryptoMgr	de.ipcon.db.CryptoManager	fully qualified Klassenname des Crypto-Managers
cryptoMgrCfg	Crypto	Name für den Abschnitt mit weiterer Konfiguration für den Crypto-Manager
noMetaDataCheck	0	Unterdrückung des Meta-Daten-Checks beim Serverstart durch Angabe von 1
noInitialDataCheck	0	Unterdrückung des Initial-Daten-Checks beim Serverstart durch Angabe von 1
noIntegrityCheck	0	Unterdrückung des Integritäts-Checks beim Serverstart durch Angabe von 1
noIntegrityDoubleIdChecks	0	Unterdrückung der Checks auf doppelte Ids beim Serverstart durch Angabe von 1 (ist automatisch inaktiv, wenn der Integritäts-Check deaktiviert wurde)
noIntegrityBLOBChecks	0	Unterdrückung des Checks der BLOBs beim Serverstart durch Angabe von 1 (ist automatisch inaktiv, wenn der Integritäts-Check deaktiviert wurde)
integrityCheckEntitiesToExcludeFromNTOMAndDoubleIdCheck (als ein Wort)	(BP, BT, Messwert)	Unterdrückung des Checks auf doppelte Ids und von n-m-Relationen für Entitäten mit den angegebenen Namen sowie deren Subentitäten
tomcatPool	TomcatPool	Aktiviert den TomcatPool für JDBC-Verbindungen. Außerdem der Name für den Abschnitt mit weiterer Konfiguration für den Tomcat-Pool
hikariPool	HikariPool	Aktiviert den HikariCP-Pool für JDBC-Verbindungen. Außerdem der Name für den Abschnitt mit weiterer Konfiguration für den HikariCP-Pool
authoritative	1	Angabe, ob dieser Server autoritativ ist (1 =ja, 0 =nein, also synchronisierende Instanz)

syncService	de.ipcon.db.SyncService	fully qualified Klassenname des Sync-Service
syncServiceCfg	SyncService	Name für den Abschnitt mit weiterer Konfiguration für den Sync-Service

fileVault

FIXME

Konfiguration des ServerFileVault

FIXME

persistenceMgr

FIXME

Konfiguration des CastorPersistenceManager

FIXME

keyGenerator

FIXME

Konfiguration des JDBCKeyGenerator

FIXME

protocolDriver

Der Protokoll-Treiber wird über den Eintrag `protocolDriver` eingestellt, indem dort die Klasse angegeben wird (fully qualified). Die Konfiguration des Protokoll-Treibers geschieht im Abschnitt, der über den Eintrag `protocolDriverCfg` gesetzt wird (default: `SocketProtocol`)

Konfiguration des Socket-Protokoll-Treibers

Für den Socket-Protokoll-Treiber können folgende Werte eingestellt werden, in Spalte 2 der jeweilige Default.

host	-	Der Hostname, unter dem auf unverschlüsselte Verbindungen gehört wird.
port	4242	Der Port, auf dem der Server auf unverschlüsselte Verbindungen hört.
backlog	10	Die maximale Anzahl von schwebenden Verbindungen für die ServerSocket für unverschlüsselte Verbindungen.

tlsHost	-	Der Hostname, unter dem auf (verschlüsselte) TLS-Verbindungen gehört wird.
tlsPort	4243	Der Port, auf dem der Server auf (verschlüsselte) TLS-Verbindungen hört.
tlsBacklog	10	Die maximale Anzahl von schwebenden Verbindungen für die ServerSocket für (verschlüsselte) TLS-Verbindungen.
maxWaitForAuth	210000	Anzahl der Millisekunden, die maximal gewartet werden soll, bis sich der BackendCommandHandlerI (d.h. der Client) authentifiziert hat; nach Ablauf dieser Zeit wird die Verbindung hart getrennt. Bitte beachten, dass der Login-Dialog die Verbindung nach 3 Minuten Untätigkeit selbstständig unterbricht. <i>maxWaitForAuth</i> sollte üblicherweise über 3 Minuten bleiben, da der Login-Dialog des Clients sonst eine Fehlermeldungen loggt.
hardMaxUnauthedBCHPerIP	150	Wenn mehr als diese Anzahl von unauthentifizierten BackendCommandHandlerI von der gleichen IP verbunden sind, werden alle weiteren Verbindungsversuche sofort zurückgewiesen.
softMaxUnauthedBCHPerIP	50	Wenn mehr als diese Anzahl von unauthentifizierten BackendCommandHandlerI von der gleichen IP verbunden sind, werden alle weiteren Verbindungsversuche erst verzögert beantwortet.
delayFactorUnauthedBCH	200	Faktor in Millisekunden, um den jede Verbindung über softMaxUnauthedBCHPerIP von der gleichen IP verzögert wird, d.h. 200ms für die 51. Verbindung, 400ms für die 52. usw.

cryptoMgr

FIXME

Konfiguration des CryptoManager

FIXME

Konfiguration des tomcatPool

maxTotal	128	Die maximale Anzahl aktiver Verbindungen, die gleichzeitig aus diesem Pool zugewiesen werden können.
maxIdle	128	Die maximale Anzahl von Verbindungen, die immer im Pool gehalten werden sollen. Leerlaufverbindungen werden regelmäßig überprüft (falls aktiviert) und Verbindungen, die länger als 60s im Leerlauf sind, werden freigegeben.
minIdle	0	Die Mindestanzahl der hergestellten Verbindungen, die zu jeder Zeit im Pool gehalten werden sollen. Der Verbindungspool kann unterhalb dieser Nummer schrumpfen, wenn Validierungsabfragen fehlschlagen.

initialSize	0	Die anfängliche Anzahl von Verbindungen, die beim Start des Pools erstellt werden.
maxWait	100 00	Die maximale Anzahl von Millisekunden, die der Pool abwarten wird (wenn keine verfügbaren Verbindungen vorhanden sind), bis eine Verbindung wieder freigegeben wird, bevor eine Exception geworfen wird.
maxAge	300 000	Zeit in Millisekunden, für die diese Verbindung gehalten wird. Wenn eine Verbindung an den Pool zurückgegeben wird, überprüft der Pool, ob das <code>now - time-when-connected > maxAge</code> Limit erreicht ist, und wenn ja, schließt er die Verbindung, anstatt sie in den Pool zurückzugeben.

Konfiguration des hikariPool

connectionTimeout	100 00	Diese Eigenschaft steuert die maximale Anzahl von Millisekunden, die ein Client auf eine Verbindung vom Pool warten wird. Wenn diese Zeit überschritten wird, ohne dass eine Verbindung verfügbar wird, wird eine <code>SQLException</code> ausgelöst. Niedrigste akzeptable Verbindungszeitüberschreitung ist 250 ms.
idleTimeout	120 000	Diese Eigenschaft steuert die maximale Zeitdauer, die eine Verbindung im Pool leerlaufen darf. Diese Einstellung gilt nur, wenn <code>minimumIdle</code> kleiner als <code>maximumPoolSize</code> ist. Ob eine Verbindung im Leerlauf verworfen wird oder nicht, unterliegt einer maximalen Variation von +30 Sekunden und einer durchschnittlichen Variation von +15 Sekunden. Eine Verbindung im Leerlauf wird niemals vor diesem Timeout verworfen. Ein Wert von 0 bedeutet, dass Leerlaufverbindungen nie aus dem Pool entfernt werden. Der minimal zulässige Wert beträgt 10000ms (10 Sekunden).
maxLifetime	300 000	Diese Eigenschaft steuert die maximale Lebensdauer einer Verbindung im Pool. Eine Verbindung in Nutzung wird niemals verworfen, sie wird nur dann entfernt, wenn sie geschlossen wurde. Wir empfehlen dringend, diesen Wert einzustellen, und es sollte mindestens 30 Sekunden weniger sein als die Verbindungszeitbegrenzung der Datenbank oder der Infrastruktur. Ein Wert von 0 bedeutet keine maximale Lebensdauer (d.h. unendliche Lebensdauer), die natürlich der <code>idleTimeout</code> -Einstellung unterliegt.
minimumIdle	0	Diese Eigenschaft steuert die minimale Anzahl von Leerlaufverbindungen, die HikariCP versucht, im Pool zu verwalten. Wenn die Anzahl der Leerlaufverbindungen unter diesen Wert sinkt, wird HikariCP versuchen, zusätzliche Verbindungen schnell und effizient hinzuzufügen. Für maximale Leistung und Reaktionsfähigkeit auf Spitzenlasten empfehlen wir jedoch, diesen Wert nicht zu setzen und stattdessen HikariCP als einen <code>fixed size</code> Verbindungspool einzusetzen.

maximumPoolSize	128	Diese Eigenschaft steuert die maximale Größe, die der Pool erreichen darf, einschließlich der Leerlaufverbindungen und der Verbindungen in Nutzung. Grundsätzlich bestimmt dieser Wert die maximale Anzahl der tatsächlichen Verbindungen zum Datenbank-Backend. Ein vernünftiger Wert dafür wird am besten durch Ihre Ausführungsumgebung bestimmt. Wenn der Pool diese Größe erreicht hat und keine Leerlaufverbindungen zur Verfügung stehen, werden die calls an getConnection() bis zu connectionTimeout Millisekunden geblockt.
-----------------	-----	---

syncService

Hier kann die Verbindung für synchronisierende Server zum Hauptknoten konfiguriert werden.
FIXME

Konfiguration des SyncService

FIXME

DeploySite

Einstellungen für das Verhalten der Deploy-Seite, betrifft u.a. den Zugriff und Download von Dawn und des Solstice-Clients. Fehlt diese Sektion in der ini, so werden als Fallback Einstellungen aus der *jetty.xml* verwendet.

host	*	Auf welchem Netzwerkinterface soll die <i>http</i> -Deployseite reagieren. Ist ein reverse-Proxy vorgeschaltet, sollte hier <i>localhost</i> eingetragen werden.
port	8080	Der http-Port, auf dem die Deployseite reagiert.
tlsHost	[host]	Wie <i>host</i> , nur explizit für TLS.
tlsPort	8443	Der https-Port der Deployseite.
useCauldron	0	0=false: Starte einen eigenen Cauldron-Server um die Deploy-Seite auszuliefern. 1=true: Falls eine explizite cauldron.conf für projektspez. APIs existiert, dann soll diese Cauldron Instanz auch für Deploy verwendet werden.
requireAuthentication	1	1=true, 0=false. Wenn 1, muss sich ein Benutzer zuerst im Browser einloggen, um die Deployseite, Dokumente, etc. sehen zu können. Als Benutzername und Passwort muss das normale Solstice-Passwort verwendet werden. Wenn 0, dann kann die Deployseite ohne Login verwendet werden. Die Inhalte des Dokumentationstabs werden dann jedoch nicht angezeigt (Ausnahme: Zugriff über localhost - siehe unten). Weitere Hinweise mit Voraussetzungen zum Login, s.u.

reauthEveryXDays	90	Wie lange ist der Zugriff auf die Deployseite bei Start über Dawn gültig, in Tagen. Hat keinen Einfluß auf den Solstice-Login.
------------------	----	--



Grundsätzlich ist der Zugriff auf die Deployseite von *Localhost* sowie aus dem lokalen Netzwerk erlaubt. Wird ein lokaler reverse-Proxy verwendet, so muss mindestens der "X-Forwarded-For" Header mit der originalen IP übergeben werden, da Cauldron sonst jeden Zugriff nur als *Localhost* erkennen kann.

Im Fall von *nginx* über

```
proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
```



Ein Benutzer muss einer der folgenden Gruppen angehören, damit er bei aktivierter Authentifikation auf die Deployseite zugreifen darf:

- *Admins* : Für Administratoren
- *RG_Solstice_Login* : Für alle Solstice Benutzer
- *RG_Deploy* : Minimale Gruppe für die Deployseite.

Die ".checked*" -Dateien

Das Vorhandensein dieser Dateien signalisiert dem Server, dass bestimmte (normalerweise eher recht lange dauernde) Überprüfungen bereits durchgeführt wurden.

.PROJEKT/.checked-firstnodestart	Wurde die Datenbank auf den ersten Knotenstart vorbereitet? (findet sich nur auf synchronisierenden Servern; wenn nicht vorhanden, werden die Logs soweit weg geräumt, dass der Sync den initialen Ansatz findet, d.h. dass nur noch eine lokale BT dort liegt und keine BPs mehr)
.PROJEKT/.checked-initialdata	Sind die nötigen Benutzer und Gruppen da, Auto-Formulare bauen, usw.
.PROJEKT/.checked-integrity	Überprüfung der Datenintegrität, z.B. auf nicht referenzierte BOs
.PROJEKT/.checked-metadata	Vergleich Schema gegen SQL-Definitionen (Tabellen, usw.)
.PROJEKT/.checked-sync	Enthält die Ids der zuletzt vom Hauptserver bzw. zum Hauptserver übertragenen BTs (findet sich nur auf synchronisierenden Servern)

Um eine der Überprüfungen (nochmal) ablaufen zu lassen, z.B. nach dem Einspielen einer Datenbank-Sicherungskopie, einfach die entsprechende(n) Datei(en) löschen; nach dem Neustart des Servers wird die entsprechende Überprüfung dann durchgeführt.

Die ".init*" -Dateien

FIXME (Erklärung)

.PROJEKT/.init-keygen	Fehlt diese Datei, so wird die bi-Tabelle in der Datenbank geleert (findet sich nur auf synchronisierenden Servern)
.PROJEKT/.init-streamcopy	Fehlt diese Datei, so werden die BLOBs im filesRoot-Verzeichnis mit dem Hauptserver abgeglichen. Der Hauptserver überträgt alle fehlenden BLOBs zum synchronisierenden Server, mit Ausnahme der BLOBs, die noch ausstehende Änderungen im Sync haben.
.PROJEKT/.init-syncaccount	FIXME (findet sich nur auf synchronisierenden Servern)

FIXME - Was darf wann und zu welchem Zweck gelöscht werden?

Services

FIXME Einleitende Erklärung zu BN, BU, BS, Skript-Service, Import-Service

BusinessNode (BN)

Wozu dienen BNs?

FIXME

Wie legt man sie an?

Seit dem Core-Codestand vom 16.11.2009 werden initiale BNs automatisch erstellt.

Wenn keine nodeName/nodeID eingetragen ist (weder in mytism.ini noch in .init-syncaccount - wobei Letzteres eigentlich nicht auftreten kann, da dann bereits vorher ein Fehler auftritt) wird beim Serverstart automatisch eine (zum Hostnamen des Server-Rechners passende) BN gesucht (falls keine solche existiert, angelegt) und ein entsprechender "nodeName"-Eintrag in der mytism.ini eingefügt. Genaueres siehe DBMan.assureServerBN().

Normalerweise wird eine BN für Server oder SyncService nur über nodeName/nodeID gefunden. Wenn allerdings beim Aufruf des SyncService keine Datei .init-syncaccount existiert bzw. wenn beim Start des DBMan keine nodeName/nodeID gefunden werden konnte, wird erst eine BN mit Name = NetworkTools.getHostname() gesucht und ggf. dann diese benutzt (und auch deren Id als nodeName/nodeID automatisch in die jeweilige Konfigurationsdatei eingetragen, d.h. das passiert nur einmal). Ansonsten spielt der Name intern AFAIK keine Rolle.

BusinessUnit (BU)

Wozu dienen BUs?

Eine BU ("Business Unit") ist eine Entität, die einen Nummernkreis repräsentiert. Business Units werden dazu verwendet, eindeutige Identifikationsmerkmale zu vergeben. Mit jeder Wert-Ermittlung wird der "nächste" Wert um eine voreingestellte Schrittweite erhöht (im Lesezeichen: "erhöhen um").

BUs sind an Knoten (BNs) gebunden, damit ein Knoten immer nur aus dem ihm eigenen Nummernkreis Werte ermittelt; die den Knoten zugeteilten Kreise gleichen Namens sind disjunkt.

Damit wird vermieden, dass zwei Knoten die gleichen BU-Werte vergeben können, da der Datenbestand ja synchronisiert wird.

Sind die Bereiche der BU aufgebraucht, ergibt die Frage nach dem "nächsten" Wert null.

Wie legt man sie an?

Man findet unter /Admins/MyTISM/Interna/BUs das Lesezeichen und die Schablone.

Will man z.B. eine BU zur automatischen Erstellung von fortlaufenden Rechnungsnummern anlegen, erstellt man mittels der BU-Schablone eine BU und trägt für die einzelnen Felder folgende Werte ein:

Name	de.oashi.bo.Rechnung.BelegNr
Beschreibung	BU für Rechnungsnummern
Next	240113
Min	240000
Max	999999
Increment	1
Valid	TRUE
Node	hier die BN auswählen, für die die BU gelten soll

Obige BU liefert uns nun in 1er-Schritten (Increments) Rechnungsnummern. Begonnen wurde mit der Rechnungsnummer 240000 und die höchste Rechnungsnummer, die vergeben werden kann, ist 999999. Als nächster Wert würde die 240113 vergeben werden. Falls etwas schiefgehen sollte und man "aus Versehen" eine Rechnungsnummer gezogen hat, kann man durch anpassen des "Next"-Wertes korrigierend eingreifen. "Valid" besagt lediglich, ob die BU aktiv ist oder nicht.

Name und Beschreibung können prinzipiell willkürlich gewählt werden, doch es bietet sich an, den Namen so zu wählen, dass man schnell erkennen kann, wo die von der BU generierte Nummer verwendet wird. In unserem Beispiel also in der Entität "Rechnung" und dort im Attribut "BelegNr".

Wie benutzt man sie?

Um die Server-seitige Vergabe von, um bei unserem Beispiel zu bleiben, Rechnungsnummern zu aktivieren, muss man in der jeweiligen BO-Klasse das SaveVeto-Interface implementieren (Beispiele hierfür finden sich in ausreichender Menge im Sourcecode).

```
method verifyOnServer(nodeNumber = Long, user = Benutzer, tx = Transaction)
  if cancelRecalc() then
    return
  super.verifyOnServer(nodeNumber, user, tx)
  if not getWartendNN() & getBelegNr() == null then
    -- eindeutige BelegNr ziehen und zuweisen
    setBelegNr(BU.nextValueAsString(getClass().getName()'.BelegNr', tx, nodeNumber))
```

In obigem Beispiel ziehen wir uns nur eine Rechnungsnummer, wenn die Rechnung nicht auf "wartend" steht und noch keine Rechnungsnummer (BelegNr) hat.

BusinessService (BS)

Wozu dienen BSe?

Um z.B. Scripte ständig oder wiederkehrend auszuführen. Ein möglicher Anwendungsfall wäre die Synchronisierung von Daten aus Fremdsystemen mit MyTISM.

Wie legt man sie an?

Man findet unter /Admins/MyTISM/Interna/BSs das Lesezeichen und die Schablone.

Will man z.B. einen BS zur automatischen Erstellung von Erinnerungs-Mails anlegen, erstellt man einen neuen Service mittels der BS-Schablone und trägt für die einzelnen Felder folgende Werte ein:

Reiter 'Dienst'

Zur Angabe von Metadaten für den Dienst.

Betreuer	Der für das Skript verantwortliche Benutzer
Bei Ausfall benachrichtigen	Gruppe, an die Fehlermeldungen gesendet werden
Name	Erinnerung an 'etwas' (beliebiger Name)
Beschreibung	BS für Erinnerungen (beliebige Beschreibung)
Java-Klasse	<code>de.ipcon.db.sync.ScriptService</code> (Klasse, die das Skript ausführt. Dies ist für die meisten Skripts die im Beispiel angegebene Klasse.)
Aktiv	Checkbox, um den Aktivitätsstatus des Skripts zu setzen
Ausführungs-Vorschrift	<pre><ExecutionPolicy> <CronJob interrupt="false"> <Commands> <Command>10 15 * * * </Command> </Commands> </CronJob> </ExecutionPolicy></pre> <p>oder</p> <pre><ExecutionPolicy> <ExecutionPolicyKeepRunning/> </ExecutionPolicy></pre>
BNs	hier die Business Nodes auswählen, für die der BS gelten soll. Wichtig! ohne Node wird der Service nirgends ausgeführt.

Reiter 'Script'

Hier wird das Skript eingegeben. Es handelt sich hierbei um XML-Skripte. Das `<Sync>`-Tag signalisiert, dass es sich um einen Dienst handelt. Darauf folgen Metadaten, die angeben, wie der Dienst sich mit dem System verbinden soll. Es folgt die Angabe der Skriptsprache und dann das Skript selbst, das in einem `<![CDATA[>`-Block enthalten ist.

```
<Sync>
  <mytism-connection url="socket://localhost" user="Benutzer eintragen"
  pass="einPasswort"/>
  <script language="groovy"><![CDATA[
    tx = api.getNewTx()
    tx.description = "Versende Benachrichtigung für someone@domain.com"
    def mail = MyTISMAAdresseEmail.getOrCreate(tx, "someone@domain.com")
    def ben = tx.includeNew(MyTISMBenAuftragOhneVorlage)
    ben.betreffQuelle = "Fehler in [xxx]"
    ben.textQuelle = "Bitte korrigieren Sie den Fehler [yyy]"
    ben.betreffIstFestQuelle = true
    ben.textIstFestQuelle = true
    ben.addEmpfaenger(mail)
    api.saveBO(tx)
  ]></script>
</Sync>
```

Reiter 'Stacktrace' Hier wird der letzte Fehler (falls aufgetreten) im letzten Lauf des Skripts angezeigt.

Wie benutzt man sie?

Business Services (BS) können sowohl als Dienst im Hintergrund laufen, als auch manuell über die Kommandozeile ausgeführt werden.

Ausführung über die Kommandozeile

```
./mytism run-bs /Pfad/zum/Script
```

ExecutionPolicy - Cron

Parameter

- *interrupt* (optional) Wenn 'interrupt' auf true steht, wird ein bereits laufender Prozess des gleichen Jobs bei der nächsten Ausführung gestoppt (default: *false*).

Jedes cron-Kommando hat 5 Felder, welche jede Minute gegen die aktuelle Zeit ausgewertet werden. Die Syntax orientiert sich an der unix crontab.

Feld	erlaubte Werte
Minute	0-59, *
Stunde	0-23, *
Tag im Monat	1-31, *

Feld	erlaubte Werte
Monat	1-12, *
Tag der Woche	0-6, * (0 ist Sonntag)

Stern (*) bedeutet: 'erster bis letzter'.

Bereiche sind erlaubt. Ein Bereich sind zwei Zahlen, getrennt durch ein Minus (-).

8-11 im 'Stunden'-Feld bedeutet: zur Stunde 8, 9, 10 und 11.

Listen sind erlaubt. Eine Liste sind zwei oder mehr Zahlen, getrennt durch ein Komma (,).

8,10,11 im 'Stunden'-Feld bedeutet: zur Stunde 8, 10 und 11.

Schritte sind erlaubt. Ein Schritt wird spezifiziert als Stern/Liste/Zeitraum Schrägstrich (/)
Schrittweite.

0-23/2 im Stunden-Feld bedeutet 'jede zweite Stunde' (ist also äquivalent zu 0,2,4,6,8,10,12,14,16,18,20,22) und */2.

*/5 im Minuten-Feld bedeutet 'alle 5 Minuten'

Der Ausführungstag hat zwei mögliche Felder - 'Tag im Monat' und 'Tag der Woche'. Wenn beide Felder verwendet werden, wird das Script evt. mehrfach ausgeführt.

30 4 1,15 * 5 würde den Dienst um 4:30 des 1. und 15. jedes Monats starten, und jeden Freitag.

Ereignisse, welche in Daylight-Saving Korrekturen fallen, werden stillschweigend nicht oder mehrfach ausgeführt.

Besondere Kommandos sind	
@yearly	Alias für "0 0 1 1 *"
@annually	Alias für "0 0 1 1 *"
@monthly	Alias für "0 0 1 * *"
@weekly	Alias für "0 0 * * 0"
@daily	Alias für "0 0 * * *"
@hourly	Alias für "0 * * * *"

Besondere Kommandos für Feld #5 (Tag der Woche)

@lastOfM	Alias für 'letzter \${Tag der Woche} im aktuellen Monat'
@lastOfY	Alias für 'letzter \${Tag der Woche} im aktuellen Jahr'

"1-20/2 16-17 * * */@lastOfY" bedeutet: 'von 16:00 bis 16:20 alle 2 Minuten und von 17:00 bis 17:20 alle 2 Minuten, an jedem Tag der letzten 7 Tage des Jahres, zum Beispiel am 25.12.2012 17:16:00, aber nicht 25.12.2012 17:15:00'

"@weekly" bedeutet: 'jeden Sonntag um 0:00 Uhr'

"30 6 * 5-7 4/@lastOfM" bedeutet: 'um 6:30 Uhr an jedem letzten Mittwoch in den Monaten Mai, Juni, Juli'

SkriptServices

FIXME Einleitende Erklärung

Benachrichtigungen

Grundlagen

MyTISM beinhaltet ein flexibles und mächtiges Benachrichtigungssystem, mit dessen Hilfe man Benutzern Informationen zukommen lassen und sie über Ereignisse informieren kann.

Mögliche Wege für Benachrichtigungen sind z.Zt. per e-Mail und (per Nachrichtenfenster) in Solstice; Benachrichtigung per Fax, per Skype oder über andere Instant-Messaging-Protokolle, etc. können bei Bedarf ebenfalls implementiert werden. Ausserdem kann jeder Benutzer die für ihn angefallenen Benachrichtigungen direkt per Lesezeichen in Solstice ansehen.

Schnelle Hilfe :-)

- Versenden *aller Benachrichtigungen* - insb. e-Mails - schnell stoppen: [Benachrichtigungssystem deaktivieren](#)
- Versenden von Benachrichtigungen *eines bestimmten Alarms* schnell stoppen: Den Alarm deaktivieren (Alarm öffnen, "Alarm ist aktiv" deaktivieren, Alarm speichern)

Alarmsystem und Benachrichtigungssystem

Das Alarmsystem und das Benachrichtigungssystem sind zwei - prinzipiell - vollkommen unabhängige Komponenten von MyTISM.

Benachrichtigungen können vollkommen unabhängig von irgendwelchen Alarmen erzeugt und versandt werden - siehe z.B. [Manuelles Versenden von Benachrichtigungen](#) oder durch Anlegen eines `MyTISMBenachrichtigungsAuftrag` im Code.

Ebenso muss ein Alarm nicht zwingend Benachrichtigungen verschicken (auch wenn das die häufigste Aktion bei Auslösung von Alarmen ist) sondern kann prinzipiell beliebige andere Aktionen ausführen.

- Falls das Alarmsystem aktiviert ist, das Benachrichtigungssystem aber nicht, werden Alarme weiterhin wie gewohnt überwacht und ausgelöst.
Falls der Alarm so konfiguriert ist, dass er Benachrichtigungen versenden soll, wird er bei Auslösung weiterhin `MyTISMBenachrichtigungsAuftrag`-Objekte anlegen - diese werden dann allerdings erst einmal in keiner Weise weiter behandelt.
Erst wenn das Benachrichtigungssystem (re)aktiviert wird, werden diese Aufträge abgearbeitet und entsprechende Benachrichtigungen versendet (siehe hierzu aber auch [Versenden veralteter Benachrichtigungen verhindern](#))
- Falls das Benachrichtigungssystem aktiviert ist, das Alarmsystem aber nicht, können weiterhin auf anderem Wege Benachrichtigungsaufträge angelegt und Benachrichtigungen versendet werden.
Es werden lediglich keine Alarme ausgelöst und daher auch keine Benachrichtigungsaufträge davon erzeugt und damit dann natürlich auch keine daraus resultierenden Benachrichtigungen versendet.
Erst wenn das Alarmsystem (re)aktiviert wird, werden ggf., je nach Alarmkonfiguration, Alarmauslösungen "nachgeholt" und dann auch entsprechende Benachrichtigungen erzeugt und versendet (siehe hierzu auch das Kapitel zum Alarmsystem, insb. Stichwort "Alte Alarme nur auslösen wenn nicht älter als").

Vorbereitung und Konfiguration

Benachrichtigungssystem-Lizenz einspielen

Das Benachrichtigungssystem ist eine optionale Erweiterung des Standard-MyTISM-Systems. Um es aktivieren und nutzen zu können, müssen Sie zuerst eine gültige Benachrichtigungssystem-Lizenz erworben und auf dem Server eingespielt haben.

Benachrichtigungssystem aktivieren

Das Benachrichtigungssystem wird - wenn eine entsprechende Lizenz vorhanden ist - automatisch aktiviert, d.h. es ist keine besondere Aktivierung nötig. Ggf. müssen allerdings Angaben zu [Verschlüsselung und Signatur](#) und zur zu benutzenden [Mailer-Konfiguration](#) eingetragen werden.

Sie können allerdings trotzdem eine explizite Angabe in der Datei `mytism.ini` machen:

```
[Notifications]
activateNotifications=if_possible
```

Sollte der entsprechende Abschnitt noch nicht existieren, fügen Sie ihn einfach ein.

Die Einstellung `activateNotifications` wird vom Server permanent überwacht und kann zur Laufzeit geändert werden. Das Benachrichtigungssystem wird dann automatisch je nach Einstellung gestartet (wobei die Konfigurationsdaten und Schlüsseldatei neu eingelesen werden) oder gestoppt.

Synchronisierende Instanzen/Multi-Node-Cluster

Wenn Benachrichtigungen genutzt werden sollen, muss das Benachrichtigungssystem im Normalfall *auf allen beteiligten Servern* aktiviert werden. Hintergrund ist, dass die eigentlichen Benachrichtigungsobjekte (`MyTISMBenachrichtigung`) jeweils auch nur auf dem Server, der den Benachrichtigungsauftrag (`MyTISMBenachrichtigungsauftrag`) erzeugt hat, angelegt werden; außerdem werden Benachrichtigungs-Popups im Solstice-GUI-Client nur von dem Server "erzeugt", auf dem der jeweilige Benutzer/Client angemeldet ist.

Welcher Server letztendlich für das eigentliche "Verschicken" der Benachrichtigungen in der gewünschten Weise verantwortlich ist, wird in spezifischer Weise bestimmt. Für den Versand via e-Mail spielen die [e-Mail-Routing-Regeln](#) die wichtigste Rolle; siehe Methode `de.ipcon.db.notification.EmailNotificationHandler#determineSender()`. Der "Versand" als Solstice-GUI-Benachrichtigung wird, wie gesagt, vom Server auf dem der jeweilige Benutzer/Client angemeldet ist, veranlasst; siehe Methode `de.ipcon.db.notification.SolsticeNotificationHandler#handleImpl()`.

Benachrichtigungssystem deaktivieren

Sollten Sie das Benachrichtigungssystem - aus welchen Gründen auch immer - deaktivieren wollen,

müssen Sie in der Datei `mytism.ini` im Abschnitt `Notifications` den Schalter `activateNotifications` auf "never" setzen.

```
[Notifications]
activateNotifications=never
```

Wenn das Benachrichtigungssystem deaktiviert ist, können zwar trotzdem weiter Benachrichtigungsaufträge erzeugt werden - z.B. vom Alarmsystem. Diese werden dann jedoch erst einmal in keiner Weise behandelt, d.h. es resultieren daraus keine Benachrichtigungen. Erst wenn das Benachrichtigungssystem (re)aktiviert wird, werden diese Aufträge abgearbeitet und entsprechende Benachrichtigungen versendet (siehe hierzu aber auch [Versenden veralteter Benachrichtigungen verhindern](#))



Die Deaktivierung des Benachrichtigungssystems bewirkt, dass von *MyTISM* keine weiteren Benachrichtigungen (inkl. e-Mails) mehr versendet werden.

Sollten aber bereits Benachrichtigungen als e-Mail beim konfigurierten Mailserver "abgeliefert" worden sein, so werden diese natürlich trotzdem von dort verschickt.

In diesem Fall können Sie lediglich vielleicht noch den Versand durch den entsprechenden Mailserver verhindern, falls Sie Zugang dazu haben.

Synchronisierende Instanzen/Multi-Node-Cluster

Wenn Sie synchronisierende Server benutzen, muss zum Deaktivieren des Benachrichtigungssystems der Schalter `activateNotifications` auch auf *allen* Servern auf `never` gesetzt werden. Server bei denen diese Einstellung nicht gemacht wurde, erzeugen ansonsten weiterhin Benachrichtigungen.

Das bedeutet, dass an Servern mit weiterhin aktiviertem Benachrichtigungssystem dort angemeldete Benutzer weiterhin Solstice-Popups bekommen. Sollte solch ein Server mittels einer [e-Mail-Routing-Regel](#) zum Versand von e-Mails konfiguriert sein, werden auch diese von dort weiterhin versendet.

Achten Sie auch darauf, dass nicht der umgekehrte Fall eintritt und einige Server fälschlicherweise `activateNotifications=never` eingetragen haben, obwohl das Benachrichtigungssystem eigentlich aktiv sein soll.

Versenden von Benachrichtigungen pausieren

Um das Versenden von Benachrichtigungen (e-Mails, Solstice-Popups, etc.) vorübergehend zu unterbinden, können Sie einfach das komplette Benachrichtigungssystem für die gewünschte Zeit deaktivieren (s.o.) Während dieser Zeit erzeugte Benachrichtigungsaufträge bleiben gespeichert und werden nach dem Reaktivieren dann abgearbeitet.



Sollten Sie ein System mit synchronisierenden Servern betreiben und *schnell* das weitere Versenden von e-Mails anhalten wollen reicht es oft prinzipiell aus, das Benachrichtigungssystem nur auf dem autoritativen Server zu deaktivieren. Sollte keine weitergehende Konfiguration über [e-Mail-Routing-Regeln](#) gemacht worden sein, verschickt nur der autoritative Server Benachrichtigungen via e-Mail.

Versenden von Benachrichtigungen an einen bestimmten Benutzer pausieren

Hierzu gibt es aktuell keine Möglichkeit. Die Versendung von Nachrichten kann nur global pausiert werden. Es gibt lediglich die Möglichkeit, die Versendung der Nachrichten für diesen Benutzer komplett zu unterbinden (s.u.).

Erzeugen von Benachrichtigungen generell verhindern

Da die Erzeugung von Benachrichtigungsaufträgen an vielen verschiedenen Stellen stattfinden kann, gibt es derzeit keine vorgefertigte Möglichkeit, die Erzeugung generell zu verhindern. Es ist lediglich möglich, die Versendung der Benachrichtigungen zeitweise zu unterbinden (s.o.)



Aktuell (Stand: 22.09.2021) werden auch gelöschte Benachrichtigungsaufträge ganz normal abgearbeitet.

Dies wird in Zukunft aber möglicherweise geändert, so dass dann gelöschte Aufträge nicht mehr behandelt werden.

In diesem Fall wäre es dann möglich, das Benachrichtigungssystem zu deaktivieren und dann vor der Reaktivierung erst alle seit der Deaktivierung erzeugten Aufträge manuell zu löschen, so dass diese dann nicht mehr behandelt werden würden.

Versenden von Benachrichtigungen an einen bestimmten Benutzer verhindern

Um zu verhindern, dass für einen bestimmten Benutzer Benachrichtigungen versendet werden, gibt es zwei Möglichkeiten:

- Als Administrator im Benutzerformular den Schalter "Anmeldung verweigern" aktivieren. Solange der Schalter aktiviert ist, werden keine Benachrichtigungen für diesen Benutzer mehr versendet.



Wie der Name schon sagt, verhindert der Schalter aber insbesondere auch, dass der Benutzer sich am System anmelden kann!

- Als Administrator in der Benachrichtigungskonfiguration des Benutzers im Benutzerformular alle Adressen entfernen (Reiter "Benachrichtigungssystem", Unterreiter "Adressen").

Benachrichtigungen für Benutzer, für die auf eine der beiden obigen Weisen die Versendung verhindert wurde, werden auch später nicht mehr "nachgeholt" und sind verloren.

Erzeugung / Loggen von Benachrichtigungsversendungen

Das Benachrichtigungssystem protokolliert, ob Benachrichtigungen an gegebene Empfänger (bzw. genauer, die spezifische Adresse oder Adressen, an die die Benachrichtigungen letztendlich gehen sollen) versendet werden konnten und falls nicht, was das Problem war. Die Information, dass eine Benachrichtigung an eine bestimmte Adresse versendet werden konnte oder nicht, wird in jedem Fall benötigt; daher werden für alle Versendungen immer Benachrichtigungsversendungen angelegt.



Ganz stimmt das obige nicht - wenn ein Benutzer im Solstice-Client eine Benachrichtigung angezeigt bekommen soll, aktuell aber nicht angemeldet ist, wird das aktuell nicht protokolliert. Dies wird sich möglicherweise aber noch ändern.

Versenden veralteter Benachrichtigungen verhindern

Für den Fall, dass Benachrichtigungen über einen längeren Zeitraum nicht versendet werden konnten, gibt es die Möglichkeit, die Menge der Benachrichtigungen über ihr Alter einzuschränken. Ist eine Benachrichtigung demnach älter als die angegebene Anzahl Tage, wird sie nicht mehr versendet.

Per default ist diese Funktion ausgeschaltet, d.h. sobald das Versenden von Benachrichtigungen wieder möglich ist, werden alle Benachrichtigungen versendet, ungeachtet ihres Alters.

Konfiguration in der mytism.ini:

```
[Notifications]
maxAgeOfNoticationInDays=3
```



Die Überprüfung ist Millisekunden-präzise; alle Benachrichtigungen, die zum Zeitpunkt der Versendung mindestens eine Millisekunde älter als die angegebene Anzahl Tage sind, werden nicht mehr versendet.

Verschlüsselung und digitale Signatur

Von MyTISM verschickte Benachrichtigungen - z.Zt. gilt das nur für e-Mails - können nach dem OpenPGP-Standard verschlüsselt und/oder mit einer digitalen Signatur versehen werden.



Damit auf Systemen mit Java 8 oder älter OpenPGP-Verschlüsselung und -Signaturen korrekt genutzt werden können müssen die "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" auf dem System eingespielt werden. Diese können bei Oracle heruntergeladen werden (<https://www.oracle.com/java/technologies/javase-jce-all-downloads.html>), siehe entsprechende Anweisungen dort.

Ob Benachrichtigungen verschlüsselt und/oder signiert werden, kann bei jedem Alarm bzw. Benachrichtigungsauftrag konfiguriert werden; außerdem kann jeder Benutzer seine bevorzugten Einstellungen wählen. Wenn bei keinem dieser Objekte explizite Einstellungen dafür eingetragen wurden, werden die Standardvorgaben des Systems verwendet, die hier konfiguriert werden:

```
[Notifications]
encrypt=if_possible
sign=if_possible
privateKeyFile=./<project>/gnupg/secring.gpg
```

Die Angabe bei **encrypt** gibt an, ob Benachrichtigungen standardmässig verschlüsselt werden sollen. Erlaubte Werte sind

never

Standardmässig sollen Benachrichtigungen *nie* verschlüsselt werden, selbst wenn entsprechende öffentliche Schlüssel bei den Benutzern hinterlegt sind.

if_possible

(Der Standardwert, wenn überhaupt nichts konfiguriert wird) Standardmässig sollen Benachrichtigungen verschlüsselt werden, wenn für den Empfänger der Benachrichtigung ein entsprechender öffentlicher Schlüssel hinterlegt ist; ist kein Schlüssel verfügbar wird unverschlüsselt gesendet.

mandatory

Standardmässig sollen Benachrichtigungen *immer* verschlüsselt werden; wenn für den Empfänger der Benachrichtigung kein entsprechender öffentlicher Schlüssel hinterlegt ist, wird der Fehler im Server-Log vermerkt und die Benachrichtigung *nicht gesendet*.

Die Angabe bei **sign** gibt an, ob Benachrichtigungen standardmässig digital signiert werden sollen. Erlaubte Werte sind

never

Standardmässig sollen Benachrichtigungen *nie* signiert werden, selbst wenn ein entsprechender privater Schlüssel für das System hinterlegt ist.

if_possible

(Der Standardwert, wenn überhaupt nichts konfiguriert wird) Standardmässig sollen Benachrichtigungen signiert werden, wenn für das System ein privater Schlüssel hinterlegt ist; ist kein solcher Schlüssel hinterlegt, werden Benachrichtigungen unsigniert gesendet.

mandatory

Standardmässig sollen Benachrichtigungen *immer* signiert werden. Wenn dieser Wert angegeben ist und kein privater Schlüssel hinterlegt ist, loggt der Server eine Fehlermeldung und das Benachrichtigungssystem ist *nicht verfügbar*.

Die Angabe **privateKeyFile** gibt den Pfad zur Datei mit dem privaten Signaturschlüssel für das System an. Der Name und Pfad der Datei ist im Prinzip frei wählbar, sinnvoll ist es aber sich an den von GnuPG vorgegebenen Standard (wie im obigen Beispiel) zu halten.



Lediglich der private Schlüssel muss hinterlegt werden und auch nur, falls von MyTISM versandte Mails signiert werden sollen. Der öffentliche Schlüssel ist *nicht* notwendig (da MyTISM ja keine verschlüsselten Mails empfangen und entschlüsseln muss).

Der private Schlüssel wird bei jedem [\(Neu-\)Start des Benachrichtigungssystems](#) neu eingelesen.

Kurzanleitung Erzeugung und Hinterlegung Schlüssel



Zum Erzeugen und Verwalten von OpenPGP-Schlüsseln via GnuPG gibt es viele Anleitungen im Netz; hier nur eine Kurzübersicht mit einigen MyTISM-spezifischen Anmerkungen.

Unter Linux kann dieser - bei installiertem [GnuPG](#) - z.B. mit dem Befehl `gpg --gen-key` erzeugt werden:



Hier bei **allen** gpg-Aufrufen immer den `--homedir`-Parameter angeben! Ansonsten werden alle Dinge im Schlüsselring des aktuellen Benutzers gemacht, was nicht sinnvoll ist. Statt dem im Beispiel verwendeten `/tmp/gnupg` kann natürlich auch ein anderes Verzeichnis angegeben werden.

1. Verzeichnis `/tmp/gnupg` muss existieren, ggf. mit `mkdir /tmp/gnupg` erzeugen.
2. `gpg --homedir /tmp/gnupg --gen-key`
3. Art: 1 / RSA+RSA ist ok.
4. Länge: Je nach Paranoia Standard von 2048 bestätigen oder 4096 eingeben.
5. Gültigkeit: 0 (verfällt nie) ist normalerweise ok.
6. "Richtig?" bestätigen.
7. Namen eingeben: Sinnvoller, erkennbarer Name für das MyTISM-System.
8. e-Mail-Adresse eingeben: Dies sollte die e-Mail-Adresse sein, die als Absender in vom MyTISM-System verschickten Mails eingetragen wird. Oft also die unter `from` eingetragene Adresse. Falls die Mails mit unterschiedlichen Adressen versandt werden, können diese Adressen später ergänzt werden (s.u.).
9. Kommentar: Keinen eingeben, leer lassen.
10. "Fertig" auswählen
11. Passphrase: **Keine** Passphrase eingeben. Da das MyTISM-System bisher keine Passphrase zum Zugriff auf den Schlüssel angeben kann, darf dieser nicht mit einer geschützt werden. Ggf. wird diese Funktionalität später noch nachgebaut, aber normalerweise sollte die Schlüsseldatei sowieso entsprechend geschützt werden, so dass niemand Zugriff darauf bekommt.
12. Warten bis Schlüssel fertig erzeugt wurde.
13. Die erzeugte Datei `/tmp/gnupg/secring.gpg` an der Stelle, die in der `mytism.ini` unter `privateKeyFile` dafür angegeben wurde, hinterlegen.
14. **Öffentliche** Schlüsseldatei den Empfängern zur Verfügung stellen (Via Mail schicken, auf

Schlüsselserver hochladen, ...) - Export mittels `gpg --homedir /tmp/gnupg --export --armor <schlüssel-id>`.

15. (Optional aber sinnvoll; weitere Infos dazu im Netz) Auf anderem Wege (Telefon, persönliches Treffen) noch die Fingerprints vergleichen und ggf. den Schlüssel von den Empfängern signieren lassen.

Weitere e-Mail-Adresse zu Schlüssel hinzufügen

1. `gpg --homedir /tmp/gnupg --edit-key <schlüssel-id>`
2. Am GPG-Kommando-Prompt: `adduid`
3. Name eingeben (kann einfach der selbe sein, wie ursprünglich eingegeben).
4. e-Mail-Adresse eingeben: Gewünschte, weitere e-Mail-Adresse eingeben.
5. Kommentar: Keinen eingeben, leer lassen.
6. "Fertig" auswählen
7. Am GPG-Kommando-Prompt: `save`
8. Aktualisierte private Schlüsseldatei im MyTISM-System hinterlegen.
9. Aktualisierte **öffentliche** Schlüsseldatei den Empfängern zur Verfügung stellen (Via Mail schicken, auf Schlüsselserver hochladen, ...) - Export mittels `gpg --homedir /tmp/gnupg --export --armor <schlüssel-id>`.

e-Mail-Einstellungen



Alle unten angegebenen Werte - mit Ausnahme der Daten bei `authMethod`, `useTLS`, `useInlinePGP` und `suppressMsgID` - sind nur Beispielwerte! Für Ihr konkretes Projekt müssen Sie natürlich die für die zu benutzenden e-Mail-Konten passenden Werte eintragen. Welche das dann sind kann diese Doku allerdings nicht beantworten, wenden Sie sich im Zweifelsfall an Ihren für e-Mail verantwortlichen Administrator :-)

Damit Benachrichtigungen per e-Mail verschickt werden können, muss vor dem Start von MyTISM konfiguriert werden, über welche(n) Mailserver und mit welchen Einstellungen Mails verschickt werden sollen.

Der einfachste Fall ist der, dass genau ein MyTISM-Knoten immer über genau einen Mailserver Mails verschickt; dieser Fall wird daher zuerst beschrieben. Komplexere Konfigurationsszenarien finden Sie im Abschnitt [Konfiguration mehrerer Mailserver zur Versendung](#).

Die Konfiguration für einen Mailserver ist in der Datei `mytism.ini` in einem Abschnitt "Mailer" zusammengefasst. Dort wird u.A. angegeben, über welchen Mailserver die Mails verschickt werden sollen und welche e-Mail-Adresse als Standard-Absender der Benachrichtigungsmails benutzt werden soll.

Mailserver zur Versendung

Die Adresse des Mailservers, über den ausgehende e-Mails versendet werden sollen wird bei `smtpHost` angegeben:

```
[Mailer]
smtpHost=mail.example.com
```

Soll die Kommunikation mit dem Mailserver über einen bestimmten Port laufen, so kann dieser getrennt mit einem Doppelpunkt direkt hinter dem SMTPHost angegeben werden, also z.B.:

```
[Mailer]
smtpHost=mail.example.com:587
```

Wenn der Mailserver nur verschlüsselte Verbindungen erlaubt kann dies mit `useTLS` aktiviert werden:

```
[Mailer]
smtpHost=mail.example.com
useTLS=1
```

`useTLS=0` TLS ist der Standard, d.h. verschlüsselte Verbindungen sind erst einmal deaktiviert.

Standard-Absenderadresse

Die Angabe bei `from` wird genauso wie sie hier angegeben wird in den "From"-Header von vom Benachrichtigungssystem verschickten e-Mails übernommen; im Prinzip können Sie dort also jeden beliebigen Text angeben:

```
from=mytism@example.com
from=MyTISM-System <mytism@example.com>
from=Ihr werdet mich nie finden!
```

(Wobei der letzte Eintrag zwar technisch möglich ist, in der Realität aber natürlich nicht viel Sinn macht ;-))

Außerdem ist zu beachten, dass es sich bei der Angabe bei `from` nur um den Standardwert handelt. Wenn für Benachrichtigungen explizit ein eigener Absender angegeben wurde, wird stattdessen die Email-Adresse dieses Absenders eingetragen.

Bei vom Alarmsystem verschickten e-Mails wird für Alarme, die keinen explizit gesetzten Verantwortlichen haben z.B. obige Angabe verwendet, da der Alarmsystem-Benutzer (der Standard-Absender dort) keinen Wert bei `Email` eingetragen hat.

Authentifizierung am Mailserver

Die meisten Mailserver verlangen eine Authentifizierung, bevor sie das Verschicken von Mails zulassen. Von MyTISM unterstützt werden zur Zeit die Methoden "POP before SMTP" (bei der die Authentifizierung am Mailserver durch ein Pseudo-Abholen von Mail über das POP3-Protokoll erfolgt) und SMTP Auth (Authentifizierung direkt für das Versenden). Ob und wenn ja welche Authentifizierungsmethode benutzt werden soll wird mittels des Attributes `authMethod` festgelegt.

Keine Authentifizierung notwendig/benutzen

Sollte der Mailserver keine Authentifizierung verlangen, setzen Sie `none` als Wert:

```
[Mailer]
from=mytism@example.com
smtpHost=mail.example.com
authMethod=none
```

Alternativ können Sie in diesem Fall den Eintrag auch ganz weglassen, da dies die Standardeinstellung ist, die bei Fehlen des Eintrages automatisch benutzt wird.



Der früher übliche Wert `0` wird auch noch unterstützt, sollte aber nicht mehr verwendet und falls vorhanden ersetzt werden.

POP before SMTP

Authentifizierung per "POP before SMTP" stellen Sie mit dem Wert `pop_before_smtp` ein; in diesem Fall müssen Sie ausserdem noch den Benutzer- bzw. Konto-Namen und das dazugehörige Passwort für das e-Mail-Konto, das "abgeholt" werden soll, angeben:

```
[Mailer]
from=mytism@example.com
smtpHost=mail.example.com
authMethod=pop_before_smtp
username=pop3kontoname
password=pop3passwort
```

Evtl. für dieses Konto vorhandene Mails werden allerdings *nicht* wirklich abgeholt; es wird lediglich eine POP3-Anmeldung durchgeführt.

Für die POP3-Anmeldung wird im Normalfall derselbe Rechner kontaktiert, der als `smtpHost` eingetragen ist. In manchen Fällen kann es aber auch sein, dass zum Versenden und Abholen von Mails unterschiedliche Rechner eingetragen werden müssen. In diesem Fall können Sie das Attribut `POPBeforeSMTPHost` dazu benutzen, anzugeben, an welchem Rechner die POP3-Anmeldung erfolgen soll:

```
[Mailer]
from=mytism@example.com
smtpHost=mail.example.com
authMethod=pop_before_smtp
username=pop3kontoname
password=pop3passwort
POPBeforeSMTPHost=pop.example.com
```



Der früher übliche Wert **1** wird auch noch unterstützt, sollte aber nicht mehr verwendet und falls vorhanden ersetzt werden.

SMTP Auth

Authentifizierung per SMTP Auth wählen Sie mit dem Wert `smtp_auth`; auch in diesem Fall ist ein Benutzername und ein Passwort notwendig:

```
[Mailer]
from=mytism@example.com
smtpHost=mail.example.com
authMethod=smtp_auth
username=smtppauthname
password=smtppauthpasswort
```



Der früher übliche Wert **2** wird auch noch unterstützt, sollte aber nicht mehr verwendet und falls vorhanden ersetzt werden.

Format für Verschlüsselung und digitale Signatur

Benutzer können konfigurieren, ob sie OpenPGP-verschlüsselte oder signierte e-Mails im (veralteten, aber von einigen Mailern noch benutzten) "Inline"-Format oder im neuen MIME-Format bekommen wollen. Für Benutzer, die diese Einstellung nicht explizit konfiguriert haben, können Sie einen System-weiten Standard vorgeben.

Standardmässig Inline-Format nicht benutzten, e-Mails im MIME-Format generieren:

```
[Mailer]
-- ...
useInlinePGP=0
```

Diese Einstellung wird auch verwendet, wenn kein `useInlinePGP`-Eintrag existiert.

Standardmässig Inline-Format benutzten:

```
[Mailer]
-- ...
useInlinePGP=1
```

Generierung des "Message-ID"-Headers unterdrücken

Im Normalfall wird für die versendeten e-Mails ein "Message-ID"-Header generiert der u.A. das Sendedatum und den Namen des Servers, von dem aus die Mail verschickt wurde, beinhaltet. Soll für die e-Mails kein solcher Message-ID generiert werden, kann diese Generierung unterbunden werden

Standardmässig wird ein Message-ID-Header generiert:

```
[Mailer]
-- ...
suppressMsgID=0
```

Diese Einstellung wird auch verwendet, wenn kein `suppressMsgID`-Eintrag existiert.

Keinen "Message-ID"-Header generieren:

```
[Mailer]
-- ...
suppressMsgID=1
```



Selbst wenn vom MyTISM-System kein "Message-ID"-Header in die Mail eingefügt wird, kann es sein, dass ein anderer Mailserver, der die Mail im Zuge des Transports zum Empfänger weiterleitet, einen solchen Header später trotzdem einträgt.

Konfiguration mehrerer Mailserver zur Versendung

In manchen Fällen kann es notwendig sein, Mails aufgrund gewisser Kriterien (z.B. in Abhängigkeit von der Mailadresse des Empfängers) über unterschiedliche Mailserver zu verschicken. Dies kann erreicht werden, indem man in der `mytism.ini` mehr als eine Mailer-Konfiguration hinterlegt und aus diesen dann je nach zu versendender e-Mail mittels definierter *e-Mail-Routing-Regeln* die passende auswählt.

```
[Mailer.Alt1]
from=mytism@example.com
smtpHost=mail.example.com

[Mailer.Alt2]
from=mytism@someOtherServer.org
smtpHost=smtp.someOtherServer.org
useTLS=1
```

Die Benennung der Sektionen muss nach dem Schema `Mailer.POSTFIX` erfolgen, wobei das POSTFIX beliebig gewählt werden kann. Außerdem ist der Name "Mailer" zugelassen.

Mittels e-Mail-Routing-Regeln können ausgehende Benachrichtigungen aufgrund der Empfänger,

des Absenders oder des Betreffs gefiltert und dann explizit mittels einer ebenfalls dort referenzierten Mailer-Konfiguration über einen bestimmten Mailserver verschickt werden.

Für jedes MyTISM-System wird automatisch eine "Catchall"-Regel angelegt, die benutzt wird, wenn keine spezifischere Regel existiert oder keine der existierenden Regeln für eine Benachrichtigung passt.

Die Filter können als reguläre Ausdrücke angegeben werden; *alle* müssen zutreffen, damit die Regel für eine e-Mail-Versendung benutzt wird. *Ausnahme*: Falls eine Mail an mehrere Empfänger gehen soll, muss der Empfänger-Filter auf *mindestens einen* davon zutreffen.

The screenshot shows a configuration window for an e-mail routing rule. The title bar reads "e-Mail-Routing-Regel [12221719] Bestellungen-Mails". The window is divided into two main sections: "Allgemein" (General) and "Wenn..." (When...).
In the "Allgemein" section, there are several input fields:

- Name: Bestellungen-Mails
- L10n-Name: Bestellungen-Mails
- Beschreibung: (empty)
- L10n-Beschreibung: (empty)
- Position: 2
- Für Server-Node: enterprise.local (Automatisch erzeugte BN fuer Server "enterprise.local.")

The "Wenn..." section contains three conditions, each with a corresponding input field:

- ...der Empfänger dem folgenden Muster entspricht: (empty)
- ...(und) der Absender dem folgenden Muster entspricht: bestellungen@example.com
- ...(und) der Betreff dem folgenden Muster entspricht: Bestellung von .*? wurde aufgegeben

At the bottom, there is a field for the mailer configuration: "...dann benutze Mailer-Konfiguration" with the value "Mailer.FuerBestellungen".

Beim Starten bzw. bei Änderungen an der `mytism.ini` wird überprüft, ob alle in e-Mail-Routing-Regeln angesprochenen Mailer-Konfigurationen (für diesen Knoten) auch wirklich vorhanden sind. Wenn nicht wird gewarnt und eine Versendung von e-Mails ist bis zu einer Korrektur der `mytism.ini` nicht möglich.

e-Mail-Versendung bei mehreren Knoten

e-Mail-Routing-Regeln sind immer für einen bestimmten Knoten bestimmt. Wenn eine Regel für eine Benachrichtigung zutrifft wird die Versendung dieser Benachrichtigung nur von dem in der Regel spezifizierten Server vorgenommen.

Durch Definition von Mailer-Konfigurationen auf mehreren Servern und entsprechende e-Mail-Routing-Regeln, die jeweils eine passende davon auswählen, ist es somit möglich, e-Mails je nach Situation von verschiedenen Servern (Knoten) verschicken zu lassen.

Wenn eine Benachrichtigung erzeugt wurde, arbeiten *alle Knoten* die vorhandenen Regeln in der Reihenfolge der angegebenen Position ab (ausgenommen CATCHALL, s.u.). Sobald eine Regel aufgrund der angegebenen Filter zutrifft, schaut jeder Knoten, ob diese Regel ihm zugewiesen ist. Wenn nein bricht der Knoten die Bearbeitung ab und macht nichts. Wenn ja generiert der Knoten eine e-Mail aus der Benachrichtigung und versendet diese unter Benutzung der in der Regel angegebenen Mailer-Konfiguration.

Es wird automatisch eine CATCHALL-Regel (ohne Filter) gebaut. Diese wird *immer* als letzte

behandelt, falls keine der vorherigen Regeln zutrif. Die CATCHALL-Regel ist immer dem autoritativen Knoten zugewiesen und benutzt dessen Konfiguration namens "Mailer".

Sollten mehrere Regeln mit gleicher Position existieren, werden diese in der alphabetischen Reihenfolge des Namens durchgegangen. Sollten selbst diese gleich sein, wird noch nach Erstellungsdatum sortiert.

Testmail beim Serverstart versenden

Man kann veranlassen, dass beim Starten des Servers automatisch eine Testmail an eine festgelegte e-Mail-Adresse versendet wird.

Testmail an support@oashi.com verschicken:

```
[Mailer]
-- ...
checkAddress=support@oashi.com
```

"Catchall"-Empfänger für Benachrichtigungen

Es ist möglich einen Standardempfänger zu definieren, an den Benachrichtigungen geschickt werden, die aus diversen Gründen nicht an den eigentlichen Empfänger versendet werden konnten.

Dieser Standardempfänger wird in folgenden Fällen benutzt:

- Der eigentliche Empfänger einer Benachrichtigung ist gelöscht.
- Der eigentliche Empfänger ist ein Benutzer, für den "Anmeldung verweigern" gesetzt ist.
- Für den eigentlichen Empfänger sind keine konkreten Adressen definiert, an welche die Benachrichtigungen verschickt werden sollen.

Die Definition dieses Standardempfängers erfolgt über die Einstellungsvariable `notifications.catchall`. Hier kann das Objekt ausgewählt werden, welches als Standardempfänger genutzt werden soll.



Der hier beschriebene Catchall hat nichts mit dem Catchall bei [e-Mail-Routing-Regeln](#) zu tun.

Solstice-Einstellungen

Plugin in Benutzer-Profil eintragen

Um Benachrichtigungen im Solstice-GUI-Client zu erhalten muss im Profil jeden Benutzers das folgende Plugin eingetragen sein:

```
<Plugin class="de.ipcon.form.notification.ClientNotificationManager" silent="yes"/>
```


Normalerweise enthält das Profil neu angelegter Benutzer automatisch diese Zeile. Falls jedoch die Profileinträge von Hand editiert wurden, kann es evtl. sein, dass diese Zeile gelöscht wurde. Auch bei bereits angelegten Benutzer kann diese Zeile noch nicht enthalten sein, wenn Sie vor der Einführung des Benachrichtigungssystems erzeugt wurden. In diesem Fall tragen Sie die Zeile bitte ein.

Der (optionale) Parameter `silent` gibt an, ob neu eintreffende Benachrichtigungen direkt geöffnet werden sollen (`silent="no"`, die Standardeinstellung wenn der Parameter nicht angegeben wurde) oder alle neuen Benachrichtigungen erst beim Anklicken des Knopfes in der "Taskleiste" angezeigt werden sollen (`silent="yes"`).

Leserechte für Benachrichtigungen geben

Um Benachrichtigungen anzeigen zu können, müssen Benutzer die entsprechenden Objekte natürlich laden bzw. lesen können. Alle Benutzer, für die Benachrichtigungen im Solstice angezeigt werden sollen, müssen daher Leserechte für die Benachrichtigungsklassen (jene in der Schema-Datei `core-benachrichtigung.xml`), insb. `MyTISMBenachrichtigung`, `MyTISMBenachrichtigungsauftrag`, `MyTISMBenachrichtigungsvorlage` und `MyTISM*BOEintrag` haben.

Sollen Benutzer (im Solstice) auf Benachrichtigungen antworten bzw. selber welche verschicken können, müssen sie für die Klassen `MyTISMBenachrichtigungsauftrag` und `MyTISM*BOEintrag` auch Erstellen- und Schreibrechte bekommen.

Eine weitere Konfiguration ist hier sonst nicht erforderlich. Es muss lediglich das Benachrichtigungssystem an sich auf jedem Server [aktiviert](#) sein, bei dem die dort angemeldeten Benutzer Benachrichtigungen auf diesem Wege erhalten sollen.

Einstellungen für die Benutzer

Damit ein Benutzer Benachrichtigungen empfangen kann, müssen noch ein paar spezielle Einstellungen vorgenommen bzw. BOs angelegt werden:

Einfache Konfiguration

Auf dem Reiter "*Einfache Benachrichtigungskonfiguration*" können Sie u.A. eine e-Mail-Adresse eintragen und angeben, ob der Benutzer Benachrichtigungen (auch) im Solstice-Client erhalten soll. Wenn Sie diese Konfigurationsfelder nutzen werden im Hintergrund automatisch entsprechende [MyTISMAAdressen](#) erstellt bzw. mit den eingegebenen Werten aktualisiert.

Verschlüsselung und digitale Signatur

Hier können Einstellungen zur OpenPGP-Verschlüsselung und -Signatur vorgenommen werden, falls der Benutzer eine von den Standardeinstellungen des Systems abweichende Konfiguration nutzen möchte.

Öffentlicher OpenPGP-Schlüssel

Hier können Sie den öffentlichen OpenPGP-Schlüssel des Benutzers hinein kopieren. Dieser ist

erforderlich, wenn der Benutzer verschlüsselte Mails erhalten möchte.

Mittels des Knopfes "*Schlüssel aus Datei importieren*" wird der Schlüssel automatisch aus der Datei `/.gnupg/pubring.gpg` im Home-Verzeichnis des aktuellen *Linux-Benutzers* eingelesen.



In gewissen Fällen wird der Schlüssel inkorrekt importiert, Grund dafür ist noch unbekannt, es scheint mit dem Format der Schlüsseldatei zusammen zu hängen.

Mittels des Knopfes "*Schlüssel vom Schlüsselserver importieren*" wird der Schlüssel automatisch von dem in der Einstellungsvariable `pgpKeyServer` angegebenen Schlüsselserver importiert.

Will verschlüsselte Benachrichtigungen

Nie = Nie verschlüsseln, selbst wenn ein öffentlicher Schlüssel verfügbar ist; **Wenn möglich** = Verschlüsseln wenn ein öffentlicher Schlüssel verfügbar ist, sonst unverschlüsselt senden; **Zwingend** = Verschlüsselung erforderlich, wenn nicht möglich Benachrichtigung *nicht senden*. Ansonsten: Standardeinstellung des Systems verwenden.

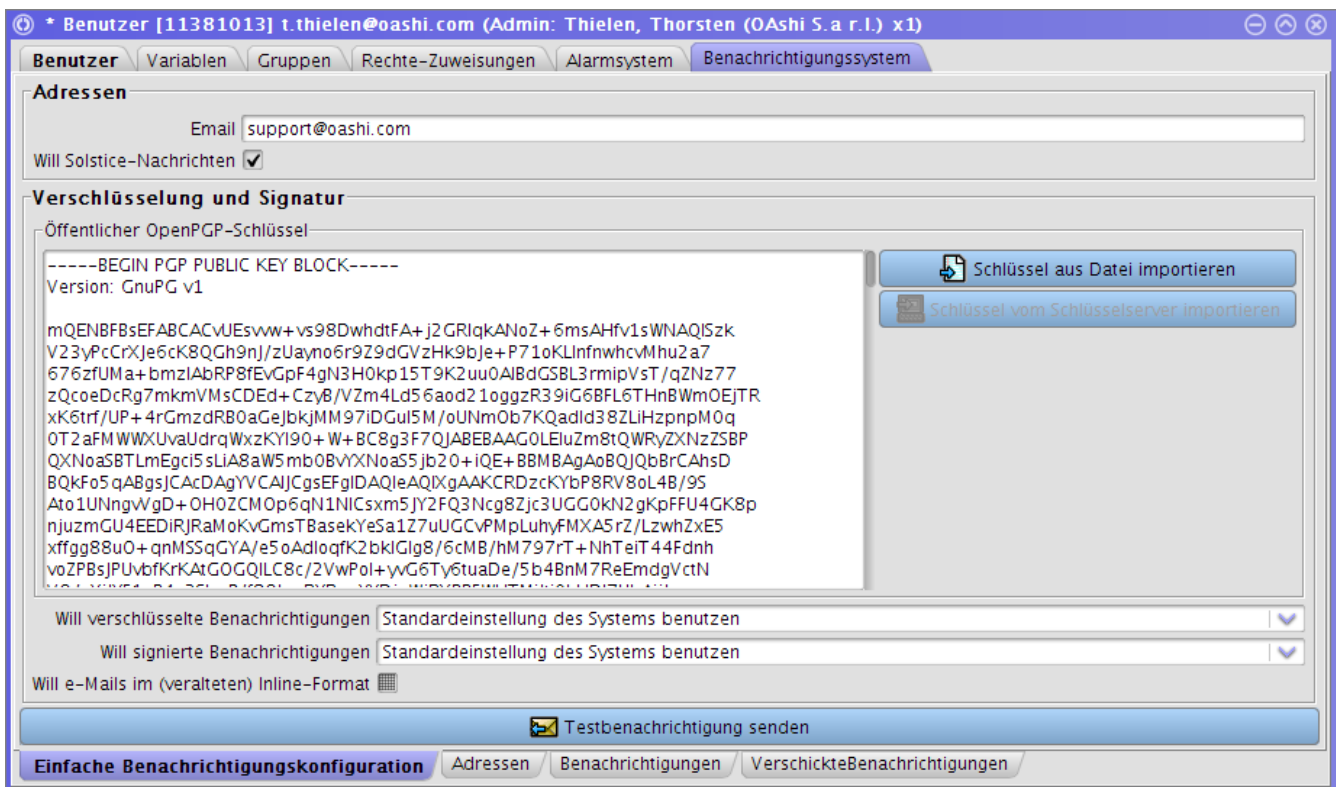
Will signierte Benachrichtigungen

Nie = Nie signieren, selbst wenn ein privater Schlüssel verfügbar ist; **Wenn möglich** = Signieren wenn ein privater Schlüssel verfügbar ist, sonst unsigniert senden; **Zwingend** = Signatur erforderlich. Ansonsten: Standardeinstellung des Systems verwenden.

Will e-Mails im (veralteten) Inline-Format

Ob verschlüsselte und/oder signierte e-Mails im (veralteten, aber von manchen Mailern noch benutzten) "Inline"-Format erzeugt werden sollen. Wenn nicht aktiviert wird das neuere MIME-Format verwendet; wenn "grau" wird die Standardeinstellung des Systems verwendet.

Mittels des Knopfes "*Testbenachrichtigung senden*" können Sie eine Testbenachrichtigung mit Standardtext an den Benutzer auslösen.



MyTISMAAdresse(n) manuell anlegen

MyTISMAAdressen dienen dazu "Ziele" für Benachrichtigungen anzugeben. Dabei kann es sich um e-Mail-Adressen handeln (*MyTISMAAdresseEmail*) oder eine *MyTISMAAdresseSolstice* die für Benachrichtigung per Nachrichtenfenster im Solstice-GUI-Client steht (in gewisser Weise ein Spezialfall, da keine eigentlichen "Adressdaten" angegeben werden müssen). *MyTISMAAdresseFax* (mit Faxnummer) steht z.Zt. ebenfalls schon zur Verfügung, leider ist die Möglichkeit Benachrichtigungen per Fax zu verschicken aber noch in Arbeit. Weitere Unterklassen, z.B. für Instant Messaging, können bei Bedarf in Zukunft hinzukommen.

Für jeden Benutzer, der Benachrichtigungen bekommen soll, muss mindestens eine Adresse angelegt werden.

Sie können die Adressen in einer bestimmten Reihenfolge anordnen, nach der sie beim Eintreffen einer Benachrichtigung abgearbeitet werden. Diese Reihenfolge wird durch das Attribut **Position** festgelegt.

Normalerweise wird nach dem ersten erfolgreichen Verschicken einer Benachrichtigung an eine Adresse die Bearbeitung abgebrochen, d.h. evtl. nachfolgende Adressen werden *nicht* mehr benutzt. Diese dienen somit nur als Ersatz/Fallback falls die vorherige(n) Adresse(n) nicht erreicht werden können. Wenn Sie bei einer Adresse allerdings das Flag **WeiterAuchWennErfolgreich** setzen, wird auch nach erfolgreichem Verschicken an diese Adresse weitergemacht. Damit kann erreicht werden, dass Benachrichtigungen über mehrere Kanäle versendet werden.

Manuelles Versenden von Benachrichtigungen



Die benötigte Schablone ist nur verfügbar, wenn sie von Ihrem MyTISM-Administrator für Benutzer bereitgestellt wurde.

Um eine Benachrichtigung zu versenden, müssen Sie einen neuen *MyTISMBenachrichtigungsauftrag* erstellen. Es gibt zwei Varianten: solche mit und solche ohne Vorlage.

Aktuell gibt es nur für Aufträge ohne Vorlage ein Schablone, da Aufträge mit Vorlage hauptsächlich für automatisch generierte Benachrichtigungen vorgesehen sind.

In der Schablone "*MyTISM-Benachrichtigungsauftrag ohne Vorlage (Vorgebaut; Versenden)*" können Sie folgende Werte angeben:

Empfänger bzw. Empfänger (versteckt)

Wer soll die Benachrichtigung erhalten?

Betreff

Der Betreff der Nachricht.

Text

Der Inhalt der Mitteilung.

Zeichensatz

Hier können Sie den Zeichensatz für die Nachricht festlegen. Dies wird normalerweise nicht benötigt, daher kann dieses Feld leer bleiben.

Priorität

Legen Sie die Wichtigkeit oder Priorität der Nachricht fest. In den meisten Fällen ist dies nicht erforderlich, sodass das Feld leer bleiben kann.

Anhänge

Hier können Sie Objekte an die Nachricht anhängen, die im Solstice-Client direkt zugreifbar sind.

Damit diese Objekte als Anhänge in per E-Mail versendeten Benachrichtigungen verwendet werden können, müssen sie bestimmte Bedingungen erfüllen (dies ist ein fortgeschrittenes Thema).

Sobald Sie den Auftrag speichern, wird er vom Benachrichtigungssystem verarbeitet. Für jeden angegebenen Empfänger wird eine Benachrichtigung versandt. Bei Gruppen erhalten alle Benutzer der angegebenen Gruppe eine Benachrichtigung. Für Empfänger, die eine Sammlung von Benachrichtigungsempfängern (`NotificationReceiverCollectionI`) darstellen, erhalten alle zu dieser Sammlung gehörenden Empfänger eine Benachrichtigung.

Programmatisches manuelles Versenden von Benachrichtigungen

Im Rahmen von Ticket 154786138 wurde eine neue Funktion zur vereinfachten programmatischen Erstellung von Benachrichtigungsaufträgen hinzugefügt.

Benachrichtigungen können jetzt mithilfe eines einfacher konfigurierbaren `MyTISMBenachrichtigungsauftrag`-Objekts versendet werden.

Im Kontext von Alarmskripten kann ein Auftrag beispielsweise durch die Methode `BenachrichtigungsScriptAPI#createBenachrichtigungsauftrag()` erstellt und mit einer neuen Fluent API konfiguriert werden.

Fluent API für `MyTISMBenachrichtigungsauftrag`:

Beispiel zur Verwendung:

```
def ticket = api.getTriggeringBO()
def sender = ticket.getAbarbeiter()
def replyTo = ticket.getEintragVon()
def notificationOrder = api.createBenachrichtigungsauftrag()
    .setSender(sender)
    .addRecipientEmail('develop@oashi.com')
    .addRecipientEmailBCC('team@oashi.com')
    .setReplyToAddress(replyTo)
    .setL10nLocale(L10nLocale.byLocale(Locale.ENGLISH))
api.sendNotification(notificationOrder)
```

In diesem Beispiel wird zunächst das auslösende Business-Objekt (`ticket`) und der Absender (`sender`) sowie die Adresse, an die geantwortet werden soll (`replyTo`) ermittelt. Danach wird ein neuer Benachrichtigungsauftrag (`notificationOrder`) erstellt und konfiguriert.

Zu den Konfigurationsmöglichkeiten gehören:

- **Absender setzen:** `.setSender`
- **Empfänger hinzufügen:** `.addRecipientEmail` (für E-Mail-Adressen) bzw. `.addRecipient` (für Benutzer, Gruppen und andere mögliche Empfängertypen (`NotifiableI`), oder jeweils mit einer Collection über `.addRecipientEmails` / `.addRecipients`)
- **BCC-Empfänger hinzufügen:** dieselben Methoden wie für Empfänger, aber mit Suffix `BCC`, z.B. `.addRecipientEmailBCC`
- **Antwortadresse festlegen:** `.setReplyToAddress` bzw. `.setReplyToAddresses` (nicht für E-Mail-Adressen implementiert, nur für `NotifiableI`)
- **Spracheinstellung vornehmen:** `.setL10nLocale`

Für `MyTISMBenAuftragOhneVorlage` gibt es zusätzlich:

- **Betreff-Quelle setzen:** `.setSubjectSource`

- **Text-Quelle setzen:** `.setBodySource`
- **Betreff-Quelle "ist fest" setzen:** `.setSubjectSourceIsFixed`
- **Text-Quelle "ist fest" setzen:** `.setBodySourceIsFixed`

Sobald der Benachrichtigungsauftrag vollständig konfiguriert ist, kann er mit der Methode `api.sendNotification(notificationOrder)` gesendet werden.

Exceptions:

- **IllegalStateException:** Diese Exception tritt auf, wenn der übergebene `MyTISMBenachrichtigungsauftrag` nicht neu ist oder mit einer anderen Transaktion erstellt wurde, d.h., nicht mit der Transaktion der `BenachrichtigungsScriptAPI`.

Diese neue Funktion bietet eine flexible und einfache Möglichkeit, Benachrichtigungen zu erstellen und zu versenden, und stellt sicher, dass alle notwendigen Parameter bequem über eine Fluent API gesetzt werden können.

Builder Pattern für MyTISMBenachrichtigungsauftrag:

Zusätzlich gibt es ein Builder Pattern für `MyTISMBenachrichtigungsauftrag`, das ähnlich funktioniert und intern die Fluent API im `#build`-Schritt verwendet.

Sie erstellen einen Builder vom entsprechenden Typ, z.B. `new MyTISMBenAuftragOhneVorlage.Builder()`, und konfigurieren ihn ähnlich wie mit der Fluent API. Am Ende rufen Sie auf dem Builder `.build(Transaction)` auf, um eine neue Instanz mit der übergebenen Transaktion zu erhalten.

Beispiel zur Verwendung:

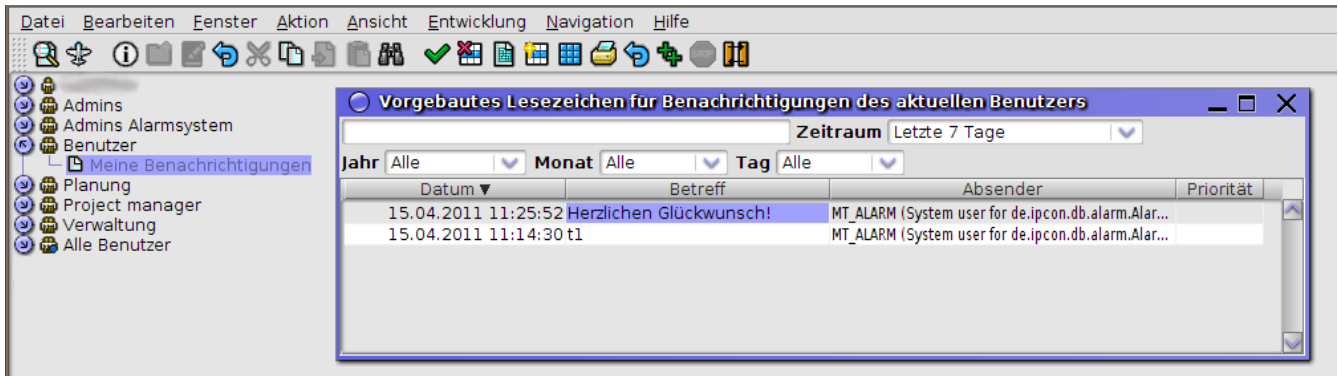
```
def tx = api.getNewTransaction()
def replyToUser = tx.getBO(123, Benutzer.class)
def notificationOrder = new MyTISMBenAuftragMitVorlage.Builder()
    .setSender(sender)
    .addRecipientEmail('develop@oashi.com')
    .addRecipientEmailBCC('team@oashi.com')
    .setReplyToAddress(replyToUser)
    .setL10nLocale(L10nLocale.byLocale(Locale.ENGLISH))
    .build(tx)
```

Benachrichtigungen einsehen



Das erforderliche Lesezeichen ist nur verfügbar, falls es von Ihrem MyTISM-Administrator für Benutzer bereit gestellt wurde.

Mitglieder der Gruppe Benutzer können alle Ihre Benachrichtigungen auch im Lesezeichen "Meine Benachrichtigungen" einsehen.



Wer kann Benachrichtigungen erhalten?

Empfänger von Benachrichtigungen können alle Objekte sein, die das Interface `de.ipcon.db.core.NotificationReceiverI` bzw. `de.ipcon.db.core.NotificationReceiverCollectionI` implementieren. Das sind im MyTISM-Grundsystem Benutzer, Gruppen und MyTISMAAdressen.

Wenn zusätzliche MyTISM-Module eingebunden sind kommen weitere Objekte dazu, z.B. Personen oder Tickets.

Applikationen können ebenfalls weitere entsprechende Objekte mitbringen.

Analyse von Problemen und Fehlerbehebung

Wenn beim Versenden einer Benachrichtigung Fehler auftreten, wird versucht diese in Abständen von einer halben Stunde erneut zu versenden. Dies gilt für alle MyTISM-Benachrichtigungsaufträge deren `Erledigt`-Flag `false` (nicht `null!`) ist.



Benachrichtigungen via Solstice-Popup, die nicht erfolgreich waren, werden *nicht* erneut versucht und zählen nicht als Fehler, wenn es darum geht, ob der zugehörige Auftrag `Erledigt` ist.

Es wird nur für die Adressen, bei denen die Versendung einer Benachrichtigung nicht funktioniert hat, ein erneuter Versuch gestartet. Empfänger, bei denen die Benachrichtigung bereits erfolgreich war werden natürlich nicht noch einmal benachrichtigt.

Wenn die Versendung für alle Empfänger einer Benachrichtigung erfolgreich war, wird automatisch das `Erledigt`-Flag des entsprechenden MyTISM-Benachrichtigungsauftrags auf `true` gesetzt. Sie können dieses Flag ebenfalls manuell auf `true` setzen, falls erneute Versendungsversuche nicht gewünscht sind.

Als *Administrator oder Benutzer mit entsprechenden Berechtigungen* können Sie kontrollieren, ob das Benachrichtigungssystem korrekt funktioniert und Benachrichtigungen wie gewünscht versendet werden bzw. wurden.

Versendung von Benachrichtigungen kontrollieren

Um zu kontrollieren, ob, wann und an wen Benachrichtigungen versendet wurden gibt es mehrere Möglichkeiten. Neben den Ausgaben im Server-Log werden auch in der MyTISM-Datenbank entsprechende Versendungen protokolliert.

Kontrolle im Server-Log

Wenn Benachrichtigungen erstellt werden, wird immer eine Meldung ins Server-Log geschrieben:

```
[...]
INFO 17:53:17.173 [r-NotificationModule] BenachrichtigungsAuftragHandler
createBenachrichtigungen(94)- Created Benachrichtigungen for
MyTISMBenAuftragMitVorlage[12345678] (scheduled at 2016-02-03 17:53:17.012) with
success for 5 recipient(s).
[...]
```



Falls Sie mehrere MyTISM-Nodes nutzen achten Sie darauf im Log des Servers nachzusehen, auf dem das Benachrichtigungssystem `aktiviert` ist.

Kontrolle in der Datenbank

Über das Lesezeichen "[Admins/MyTISM \(Vorgebaut\)/Benachrichtigungen/MyTISM-Benachrichtigungsaufträge \(Vorgebaut\)](#)" können Sie sich die gespeicherten Benachrichtigungsaufträge für einen gewünschten Zeitraum anzeigen lassen.

Durch Aktivieren des Filters "*Mit "echten" Fehlern*" können Sie sich schnell einen Überblick über aufgetretene Probleme verschaffen.

- Wählen Sie den Zeitraum, innerhalb dessen die fragliche Benachrichtigung oder Benachrichtigung versendet wurde bzw. hätten werden sollen und kontrollieren Sie, ob es entsprechende Benachrichtigungsaufträge gibt.
- Sind diese vorhanden prüfen Sie, welche Benachrichtigungen dafür erzeugt wurden. Öffnen Sie den Auftrag und wechseln Sie auf den Reiter "*Benachrichtigungen*". Kontrollieren Sie, ob für alle gewünschten Empfänger eine Benachrichtigung erstellt wurde.
- Kontrollieren Sie für alle Benachrichtigungen ob die Checkboxes in der Spalte "Erfolgreich versendet" markiert sind. Falls ja, sollte die Benachrichtigung auch erfolgreich versendet bzw. im Solstice-Client angezeigt worden sein.
- Im Fehlerfall können Sie auf dem Reiter "*Versendungen*" sehen, welche Versendungen nicht erfolgreich waren und welcher Fehler dabei aufgetreten ist. Was protokolliert wird hängt hier allerdings von der [Einstellung zur Protokollierung von Benachrichtigungsversendungen](#) ab.

Dienst zum Informieren über fehlgeschlagene Benachrichtigungsaufträge

Im `de.ipcon.db.core` Package gibt es die Klasse `NotifyAdminsOfFailedBenachrichtigungsAuftragsService`, die in einem Dienst instantiiert und gestartet werden kann, um eine Email über trotz Retries fehlgeschlagene MyTISM-Benachrichtigungsaufträge zu erhalten. Der Email-Verteiler kann über die Einstellungs-Variable `notification.recipients.mytismBenachAuftragFailed` eingestellt werden (komma-separierte Liste von Benutzer- und/oder Gruppennamen).

Eine Vorlage für einen solchen einmal täglich laufenden Dienst (BS) befindet sich in der Datei `/nrx/de/ipcon/resources/services/bs/NotifyAdminsOfFailedBenachrichtigungsAuftrags.bs.xml`.

Benachrichtigungen für spezifische Objekte finden

Oft tritt der Fall auf, dass kontrolliert werden soll, ob eine Benachrichtigung erzeugt und/oder verschickt wurde, die sich auf ein bestimmtes Objekt bezieht.

In den meisten Anwendungsfällen sollten sie hier mit einer der beiden folgenden Methoden weiterkommen:

Methode 1



Funktioniert nur, wenn das entsprechende Objekt in den Kontext des Auftrags aufgenommen oder als Anhang angegeben wurde.

1. Ermitteln Sie die MyTISM-Id des gewünschten Objekts.

2. Öffnen Sie das Lesezeichen [/Admins/MyTISM/Benachrichtigungen/MyTISM-Benachrichtigungsaufträge \(Vorgebaut\)](#)
3. Tragen sie folgendes in das Suchfeld ein: `[exists(within KontextBOs b where b.BO.Id = <objekt-id>) or exists(within AnhangBOs c where c.BO.Id = <objekt-id>) - <objekt-id>` ersetzen Sie durch die oben ermittelte Id des Objekts.
4. Schicken Sie die Suchanfrage ab.

Falls die Benachrichtigungen von einem Alarm erzeugt wurden, muss - für solche Benachrichtigungen die mit einem Core-Codestand nach dem 30.10.2019 erzeugt wurden - eine leicht andere Anfrage gestellt werden:

1. Ermitteln Sie die MyTISM-Id des gewünschten Objekts.
2. Öffnen Sie das Lesezeichen [/Admins/MyTISM/Alarmer/Ausloesung/AlarmBenachrichtigungsauftraege](#)
3. Tragen sie folgendes in das Suchfeld ein: `[a.(AlarmAusloesungFuerBO)FuerAusloesung.BO.Id = <objekt-id> or exists(within AnhangBOs c where c.BO.Id = <objekt-id>) - <objekt-id>` ersetzen Sie durch die oben ermittelte Id des Objekts
4. Schicken Sie die Suchanfrage ab.

Falls entsprechende Benachrichtigungsaufträge gefunden werden, können sie über diese die erzeugten Benachrichtigungen und deren Versendungen einsehen.

Methode 2



Funktioniert nur, wenn Daten - am besten die MyTISM-Id - des entsprechenden Objekts in den Text der Benachrichtigung aufgenommen wurden.



Normalerweise deutlich langsamer als Methode 1!

1. Ermitteln Sie die MyTISM-Id des gewünschten Objekts oder irgend einen anderen, möglichst eindeutigen/spezifischen Wert nach dem Sie im Text der Benachrichtigungen suchen können.
2. Öffnen Sie das Lesezeichen [/Admins/MyTISM/Benachrichtigungen/MyTISM-Benachrichtigungen](#)
3. Tragen sie folgendes in das Suchfeld ein: `[TextFest ilike '%<suchbegriff>%' - <suchbegriff>` ersetzen Sie durch die oben ermittelte Id oder den Suchbegriff für das Objekt.
4. Schicken Sie die Suchanfrage ab.

Falls entsprechende Benachrichtigungen gefunden werden, können sie über diese deren Versendungen einsehen.

Checkliste mögliche Fehlerquellen



Zur Kontrolle, um die entsprechenden unten genannten Log-Meldungen zu provozieren, können Sie das Benachrichtigungssystem "durchstarten". Ändern Sie dazu in der Datei `mytism.ini` den Eintrag `activateNotifications` auf "never" falls er auf "if_possible" oder "mandatory" steht und dann wieder zurück auf den ursprünglichen Wert.

Das Benachrichtigungssystem wurde nicht **aktiviert** oder bei der Aktivierung sind Fehler aufgetreten

Wenn das Logfile, in dem der Serverstart protokolliert wurde, noch verfügbar ist muss eine Meldung

```
[...]  
INFO 11:21:29.244 [t-NotificationModule] NotificationModule doStart(118)-  
Notification system successfully initialized and ready.  
[...]
```

vorhanden sein. Prüfen Sie ebenfalls die entsprechende Konfiguration in der Datei `mytism.ini`.

Das Benachrichtigungssystem wurde zur Laufzeit deaktiviert

Prüfen Sie die entsprechende Konfiguration in der Datei `mytism.ini`. `activateNotifications` muss auf "if_possible" oder "mandatory" stehen.

```
[Notifications]  
activateNotifications=if_possible
```

Ein Handler konnte nicht gestartet werden

Das Benachrichtigungssystem ist zwar aktiv, aber ein oder mehrere Handler (wie z.B. der, welcher sich um die Versendung von Benachrichtigungen per e-Mail kümmert) konnten nicht korrekt gestartet werden. Achten Sie hier ebenfalls auf entsprechende Meldungen im Server-Log, vor der Aktivierungsmeldung des Benachrichtigungssystems.

Die **e-Mail-Routing-Regeln** sind nicht richtig konfiguriert

Sowohl Benachrichtigungssystem als auch der Handler zur e-Mail-Versendung sind aktiv; aber die e-Mail-Routing-Regeln sind nicht korrekt definiert. Häufigste Ursache ist, dass "Für Server-Node" nicht korrekt gesetzt ist. Eventuell greifen auch die "Wenn..."-Bedingungen nicht, weil sie unpassend definiert wurden.

Tipps und Tricks

Testbenachrichtigungen senden

Mittels des Knopfes "*Testbenachrichtigung senden*" im [entsprechenden Reiter des Benutzer-Formulars](#) können Sie eine Testbenachrichtigung mit Standardtext an den gewählten Benutzer - also z.B. sich selbst - senden lassen.

Navigationsbaum

Der Navigationsbaum stellt eine wichtige Komponente der Solstice-Benutzeroberfläche dar, indem er für den jeweiligen Benutzer den Zugriff auf die für ihn verfügbaren Elemente strukturiert und somit eine benutzerspezifische Systemübersicht bietet.

Angezeigt werden im Navigationsbaum generell:

- Strukturelemente (Ordner, Lesezeichen, Schablonen, Formulare, Reports und Aliase darauf)
 - virtuelle Ordner
 - ein virtueller Ordner für den angemeldeten Benutzer
 - für Administratoren ein virtueller Ordner mit allen Benutzern
 - und nach dem Suchen von Strukturelementen ein virtueller Ordner mit Unterordnern für die Suchergebnisse.

Aussehen und Position von Elementen

Im Normalfall werden Elemente in Ordnern alphabetisch sortiert; es ist jedoch möglich, eine gewünschte Reihenfolge manuell festzulegen, indem man für das Element eine gewünschte Position einträgt. Elemente mit Position werden in der dadurch angegebenen Reihenfolge und vor allen Elementen ohne Position angezeigt.

Es ist außerdem möglich, Elemente durch zuweisen einer Hintergrundfarbe besonders hervorzuheben. Die Farbe muss HTML-kodiert angegeben werden.

Sichtbarkeit von Elementen

Welche Strukturelemente im Navigationsbaum für einen angemeldeten Benutzer sichtbar sind, wird von mehreren Faktoren gesteuert; u.a. im Zusammenspiel mit den von der [Rechteverwaltung](#) vergebenen Rechten.

Folgende Dinge werden in der angegebenen Reihenfolge geprüft:

Element gelöscht?

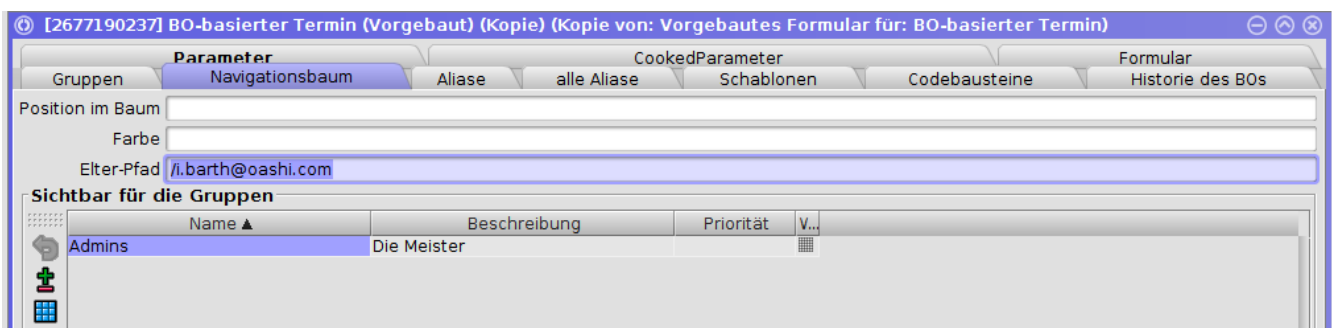
Gelöschte Elemente werden nie im Baum angezeigt.

Sichtbarkeitsskript

- Falls für das Element ein *Sichtbarkeitsskript* (Sprache *Groovy*) hinterlegt ist, wird dieses ausgeführt.
- Falls es "wahr" (`true`) für den angemeldeten Benutzer liefert, wird mit der nächsten Überprüfung fortgefahren; ansonsten ist das Element nicht sichtbar.

Element für Benutzer bzw. mindestens eine seiner Gruppen als "sichtbar" konfiguriert?

- Wenn *SichtbarFuerGruppen* (s. Screenshot weiter unten) für das Element mindestens einen Eintrag hat, wird geprüft, ob der Benutzer mindestens einer der Gruppen aus *SichtbarFuerGruppen* angehört; falls ja wird mit der nächsten Überprüfung fortgefahren; ansonsten ist das Element nicht sichtbar.
- Wenn *SichtbarFuerGruppen* keine Einträge hat oder der angemeldete Benutzer ein Administrator ist wird hier nichts geprüft.



Ausreichende Rechte vorhanden?

- Es wird geprüft, ob der Benutzer ausreichende Rechte hat, um das Element sinnvoll nutzen zu können.
- Für Schablonen oder Aliase darauf muss er Objekte der zugehörigen Klasse erstellen und schreiben dürfen; für Lesezeichen und eigenständige Reports oder Aliase darauf muss er Objekte der zugehörigen Klasse oder einer ihrer Unterklassen lesen dürfen.
- Sollte bei der Überprüfung der Rechte ein Fehler auftreten, wird das Element ebenfalls angezeigt - dieser Fall ist insb. für "kaputte" Aliase ohne Verweis auf ein Original gedacht, damit diese im Baum sichtbar sind und repariert werden können.

Element an der Wurzel des Baumes?

Elemente ohne Elter, d.h. an der Wurzel des Baumes werden nun angezeigt.

Element Kind eines bereits angezeigten Elements?

Elemente die Kinder eines bereits angezeigten Elements sind, werden ebenfalls angezeigt.

Im Normalfall reicht es, die Sichtbarkeit von Ordnern zu beschränken, da deren Kindelemente - selbst solche, die sonst im Prinzip für den angemeldeten Benutzer sichtbar wären - natürlich ebenfalls nicht angezeigt werden, wenn der Ordner an sich bereits nicht angezeigt wird. Es ist aber auch möglich, mittels der oben beschriebenen Konfigurationsmöglichkeiten einzelne Kindelemente eines Ordners auszublenden.

Durch Eintragen von `log4j.logger.de.ipcon.form.navtree.BenanntNavigationTreeNode=TRACE` in der `/.projekt/client-log.conf` können hierzu Debug-Informationen im Client-Log ausgegeben werden.



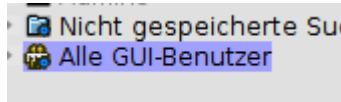
Die Eintragungen auf den Reitern "Gruppen" für Strukturelemente spielen für die Anzeige im Baum *keine* Rolle! Diese werden nur benötigt bei Anzeige im Kontextmenü und ähnlichen Stellen (beispielsweise beim Öffnen eines Strukturelements per Doppelklick). Für Lesezeichen spielen sie praktisch keine Rolle.



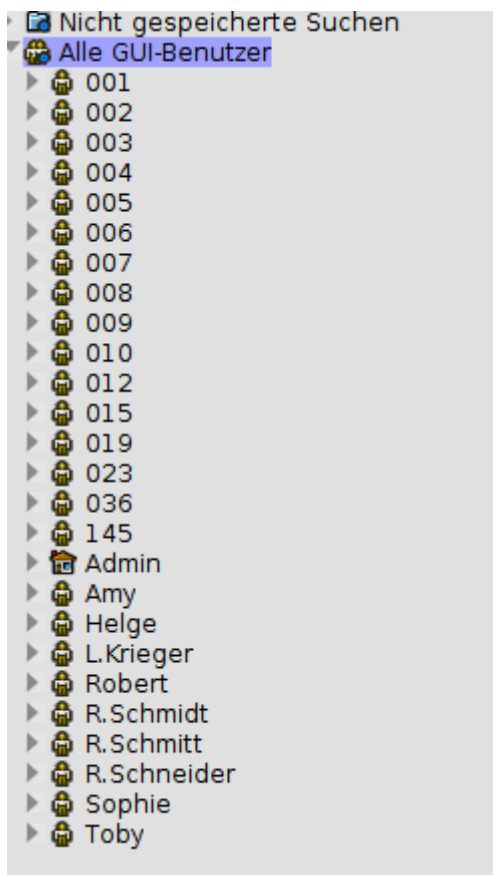
Bei generierten Strukturelementen sollten selbstverständlich keinerlei Eintragungen vorgenommen werden, da diese natürlich nicht dauerhaft verfügbar wären.

Der Ordner "Alle GUI-Benutzer" und seine Konfiguration

Am unteren Ende der Navigationsbaums befindet sich ein Ordner namens "Alle GUI-Benutzer", in dem alle Benutzer angezeigt werden, die sich über den Client in das System einloggen können.

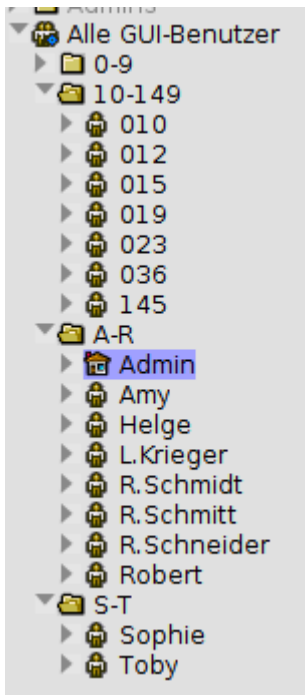


Wenn es nicht sehr viele Benutzer im System gibt, könnte der Inhalt des Ordners ungefähr so aussehen:



Überschreitet die Anzahl der User jedoch ein bestimmtes Limit, werden sie anhand ihres Namens kategorisiert. Die genauen Kriterien der Kategorisierung lassen sich über den entsprechenden Satz **EinstellungsVariablen** beeinflussen. Generell gilt, dass die Logik dahinter versucht, ähnliche Namen in einen gemeinsamen Unterordner zu packen. (Beispielsweise alle Namen, die mit "A" beginnen, oder alle Namen, die aus Zahlen bestehen)

Das obige Beispiel würde sich dann so darstellen:



Konfigurationsparameter

Benutzer können durchnummeriert sein, Kürzel oder volle Namen verwenden. Abhängig davon, welche Namenskonvention gängig ist, macht es durchaus Sinn, die Ansicht auf den entsprechenden Fall anzupassen.

"users.view.groupingStart"

- Dieser Wert beschreibt die Mindestanzahl an Benutzer, die überstiegen werden muss, damit eine Aufteilung in Unterordner ("Gruppierung") vorgenommen wird.
- **Standardwert:** 30

"users.view.enableGrouping"

- Hierüber kann die Gruppierung in der Ansicht der Benutzer ausgeschaltet werden, selbst wenn die Anzahl der Benutzer das Limit übersteigt. So werden immer alle Benutzer untereinander angezeigt
- **Standardwert:** enabled

"users.view.group.forceAlphaNumBreak"

- Ist diese Variable angeschaltet, wird bei einem Wechsel von rein numerischen zu alphabetischen (oder gemischten) Benutzernamen immer ein neuer Unterordner erstellt. Andernfalls muss vor der Erstellung zuerst die Mindestanzahl an Elementen pro Ordner erreicht werden.
- **Standardwert:** enabled

"users.view.group.maxElements"

- Dieser Wert beschreibt das absolute Maximum an Benutzern, die ein Unterordner aufnehmen darf, ehe ein weiterer Unterordner der Liste hinzugefügt wird. Gibt es also 15 Benutzer, deren Name mit "A" beginnt und "maxElements" ist auf "10" gesetzt, werden diese auf mindestens zwei Unterordner aufgeteilt.

- **Standardwert:** 15

"users.view.group.minElements"

- Dieser Wert beschreibt das Minimum an Benutzer, die ein Unterorder aufnimmt, ehe er die Aufnahme weiterer Elemente aufgrund von anderen Kriterien verweigern darf. Dies bedeutet, dass die Anzahl von Benutzern pro Unterordner zwischen "minElements" und "maxElements" liegt.
- **Standardwert:** 5

"users.view.group.numericRange"

- Der numerische Bereich definiert, wie Benutzer unterteilt werden sollen, deren Namen ausschließlich aus Zahlen bestehen. Ein Wert von "10" sorgt dafür, dass Benutzer in 10er-Schritten unterteilt werden, ein Wert von "100" in 100er-Schritten, usw.
- **Standardwert:** 10

"users.view.group.maxSubgroups"

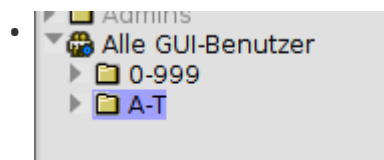
- Wenn alle Benutzer "A" und alle Benutzer "B" in einen Unterordner gesteckt werden, da das "minElement" -Limit nicht erfüllt wurde, so bilden "A" und "B" innerhalb des Unterordners sog. "Untergruppen". Auch numerische und nicht-numerische (oder gemischte) Benutzernamen bilden eigene Untergruppen. Ist dieses Limit vor "maxElements" erreicht, müssen nachfolgende Benutzer in dem nächsten Unterordner gruppiert werden.
- **Standardwert:** 4

Beispiel-Konfigurationen

"0-9,A-Z"

Gruppiere numerische und nicht-numerische Benutzernamen.

- **users.view.group.numericRange:** 100000 (sollte grob die Anzahl der numerischen Benutzernamen widerspiegeln)
- **users.view.group.maxElements:** 9999 (oder ein ähnlich hoher Wert)
- **users.view.group.minElements:** 0
- **users.view.group.maxSubgroups:** 999 (muss kleiner sein als **maxElements**)

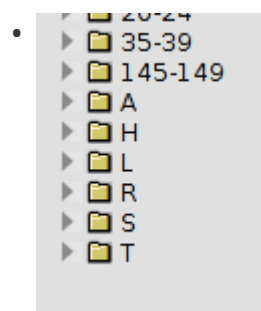


"0,1,2... A,B,C,...Z"

Gruppiere Benutzernamen nach Anfangsbuchstaben.

- **users.view.group.numericRange:** (Da Zahlen als Zahlen interpretiert werden, sollte der Wert > 1 liegen)
- **users.view.group.maxElements:** 9999 (oder ein ähnlich hoher Wert)
- **users.view.group.minElements:** 0

- `users.view.group.maxSubgroups: 1`



← Example with `numericRange = 5`

Feste Größe der Unterordner

- Dazu einfach `users.view.group.minElements = users.view.group.maxElements` setzen

Rechteverwaltung



Alle genannten Formulare, Schablonen und Lesezeichen liegen normalerweise im Ordner "*Benutzerverwaltung*".



Änderungen an Gruppen- und Rechtezuweisungen wirken sich für angemeldete Benutzer erst aus, nachdem sie sich einmal ab- und wieder angemeldet haben!

Grundlagen

Rechteverwaltung bedeutet, dass man bestimmten Personen das Lesen, Ändern, Neuanlegen oder "Löschen" (d.h. "als gelöscht Markieren") von bestimmten Objekten oder Daten (in MyTISM also BOs bzw. deren Attribute) erlauben oder verweigern will.

Hierbei sollte man sich folgende Fragen stellen:

Wer?

Welchem "Personenkreis" will ich irgendwelche Dinge erlauben oder verbieten?

Was?

Für welche Menge von Objekten/Daten will ich Sachen erlauben oder verbieten?

Wie?

Wie sollen die Rechte aussehen, d.h. was genau soll erlaubt oder verboten werden?

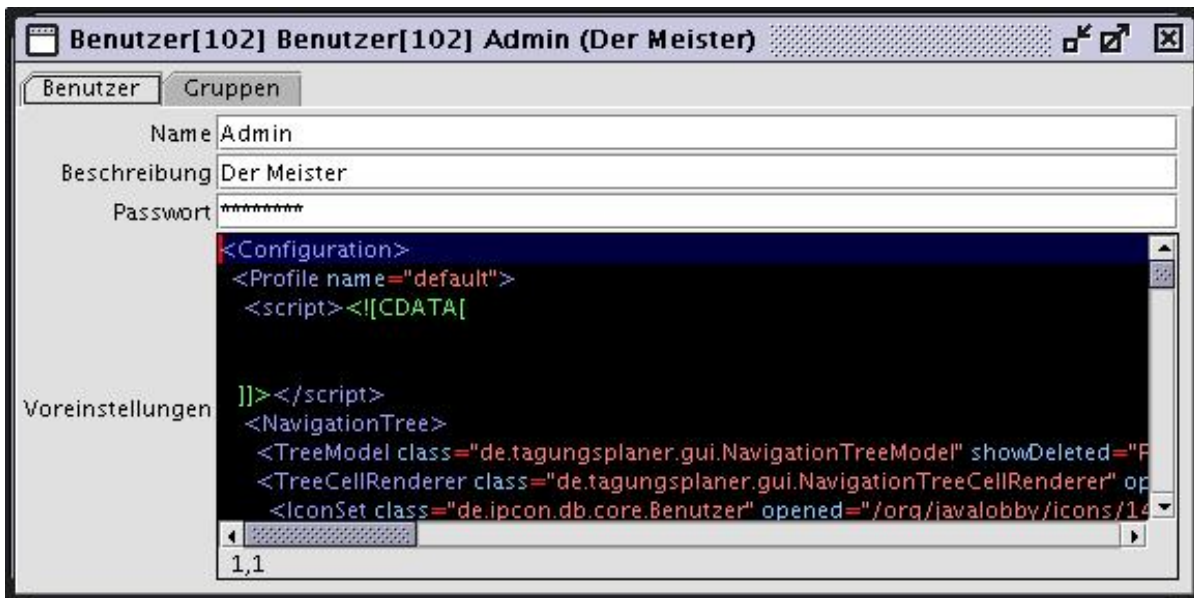
Alle Mitglieder der Geschäftsführung (*Wer?*) dürfen die Mitarbeiter-Daten (*Was?*) ansehen und ändern (*Wie?*). Alle anderen (*Wer?*) dürfen die Mitarbeiter-Daten (*Was?*) nur ansehen (*Wie?*).

Erste Voraussetzung für die Vergabe von Rechten ist natürlich, dass es auch irgend jemanden gibt, für den man irgendwelche Rechte definieren kann. In MyTISM gibt es dafür, ähnlich wie auch in vielen anderen Anwendungen, das System von Benutzern und Gruppen.

Benutzer

Ein *Benutzer* steht für eine einzelne reale oder virtuelle (z.B. externe Programme) Person, die mit einem MyTISM-System arbeiten kann, bzw. genauer eigentlich für das dieser Person zugordnete Benutzerkonto. Die definierten Benutzer bestimmen, wer überhaupt auf ein MyTISM-System zugreifen darf.

Der Benutzer "*Admin*", der alles sehen kann und alles darf, wird für jedes MyTISM-System automatisch angelegt. Weitere Benutzer können dann je nach Bedarf hinzugefügt werden, haben allerdings dann noch keinerlei Rechte.



Für dieses Tutorial nehmen wir an, dass folgende Benutzer bereits im System definiert wurden: Admin, Alice, Bob und Claire

Gruppen

Mehrere Benutzer kann man in einer *Gruppe* zusammenfassen. Da Rechte in MyTISM nur an Gruppen und nicht an einzelnen Benutzern hängen können, muss jeder Benutzer mindestens einer Gruppe angehören.



Um Rechte nur an einen einzelnen Benutzer zu vergeben kann man natürlich eine eigene Gruppe bauen und dann diesen Benutzer als deren einziges Mitglied definieren.

Eine Gruppe "Admins" wird automatisch gebaut und enthält am Anfang nur den Benutzer "Admin". Ebenso gibt es eine Gruppe "Benutzer", die alle Benutzer enthalten sollte.



Es ist vorgesehen, jedoch *nicht* verpflichtend, dass jeder Benutzer Mitglied der "Benutzer"-Gruppe ist.



Voreinstellungen für Benutzer und Gruppen

Beim Erstellen eines Benutzers oder einer Gruppe können Voreinstellungen, wie z.B. das Theme der GUI, festgelegt und als (Beanshell-)Skript hinterlegt werden. **FIXME weiter ausführen**

In diesem Tutorial benutzen wir folgende Gruppen: Admins, Benutzer (enthält Alice, Bob und Claire) und Chefs (enthält Claire)

Nachdem man mittels Benutzer und Gruppen einen "Personenkreis" definiert hat, für den man Rechte vergeben will, kann man nun nach Bedarf die einzelnen Berechtigungen definieren.

BO-Masken

Um Rechte für bestimmte Objekte festzulegen, muss man die entsprechenden Objekte natürlich irgendwie auswählen. Dies geschieht mit Hilfe der sog. *BO-Masken*; damit definiert man eine Menge von BOs, für welche man dann einer oder mehreren Gruppen bestimmte Rechte erlauben oder verweigern will. Diese Rechte gelten dann uneingeschränkt auch für die Kinder des Objektes (also auch für die Attribute des erbenden BOs, die in der Eltern-Entität nicht definiert sind).

BOMasken haben folgende Eigenschaften:

Name

Pflichtfeld - Eine passender, aber im Prinzip frei wählbarer Name für die BO-Maske. **FIXME es gibt eine Namenskonvention?**

Beschreibung

Optional - Ein kurzer Text, der in ein paar Worten "menschlesbar" erklärt, welche BOs mit dieser Maske ausgewählt werden.

...vom Typ

Pflichtfeld - Hier bestimmt man, BOs welchen Typs diese BO-Maske auswählt.

...für die dieses Skript "wahr" liefert

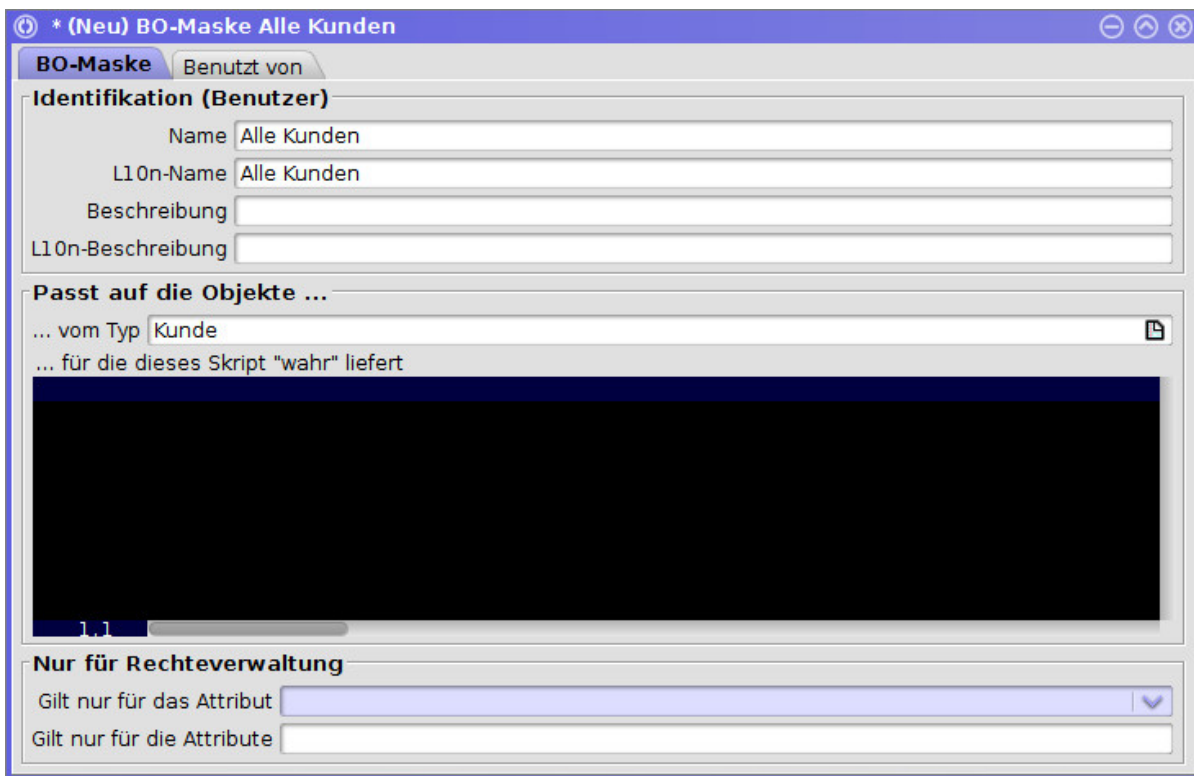
Optional - Erlaubt die Auswahl weiter einzuschränken; wird im [nächsten Abschnitt](#) genauer erklärt.

Gilt nur für das Attribut

Optional - Wenn Sie einen Wert angeben, so bezieht sich die Maske nur auf dieses Attribut der BOs, d.h. sie können hiermit Rechte für einzelne Eigenschaften von BOs vergeben. Dieses Feld ist eine vereinfachte Eingabemöglichkeit, wenn Sie hier nur ein einzelnes Attribut angeben möchten. Wenn Sie unter "*Gilt nur für die Attribute*" mehr als ein Attribut aufführen, wird dieses Feld automatisch ausgeblendet.

Gilt nur für die Attribute

Optional - Wenn Sie hier einen Wert angeben, so bezieht sich die Maske nur auf diese Attribute der BOs, d.h. sie können hiermit Rechte für einzelne Eigenschaften von BOs vergeben. Geben Sie ein oder mehrere Attribute mit ihrem Namen an, getrennt mit Komma.



Filterskript

Mittels des optionalen Filterskripts (Sprache *Groovy*) unter "...für die dieses Skript "wahr" liefert" kann die Menge der Objekte, welche von der Maske ausgewählt werden, weiter eingeschränkt werden.

Es wird für jedes BO, das von der Maske "überprüft" wird ausgewertet. Nur wenn das Skript wahr (**true**) zurückliefert, wird das entsprechende BO berücksichtigt. Im Skript stehen folgende vordefinierte Variablen zur Verfügung (definiert in der Datei */nrx/de/ipcon/db/core/BOMaske.nrx* und in der darin definierten Methode *createScript*).

Variablenname	Beschreibung
bo	Das zu überprüfende BO.
schema	Das verwendete Schema.
entity	Der in der BO-Maske angegebene Typ (<i>Entität</i>).
att	Der <i>Name</i> des Attributs, für das Rechte überprüft werden sollen. Ggf. null wenn kein spezielles Attribut sondern das gesamte BO überprüft werden soll. Auch null bei Verwendung der Maske außerhalb des Rechtesystems.
maske	Ein Verweis auf die BO-Maske selbst.
log	Ein Logger mit Hilfe dessen (Fehler)Meldungen ins Log ausgegeben werden können.
user	Der aktuelle Benutzer.



`return` kann in Groovy weggelassen werden, der Rückgabewert eines Skripts ist automatisch das Ergebnis des letzten Ausdrucks.

Example 1. Beispiele für Filterskripte:

(Eher unsinnig :-)) Maske wählt nur BOs mit einer Id > 1000 aus:

```
bo.Id.longValue() > 1000
```

Maske wählt nur BOs aus, die das Flag IstSchoen (ein Boolean-Attribut) gesetzt haben:

```
// Voraussetzung: "Entitaet" der Maske ist eine Entitaet, die dieses Flag hat.  
bo.IstSchoenNN
```

Maske wählt nur Gruppen, in denen der aktuelle Benutzer Mitglied ist:

```
// Voraussetzung: "Entitaet" der Maske ist "Gruppe".  
bo.Benutzer.containsKey(user.Id)
```



Definieren Sie nicht leichtfertig Skripte für BO-Masken. Das Skript muss für *jedes* Objekt vom Typ der Maske aufgerufen und ausgewertet werden, wenn ein entsprechendes Objekt gelesen, bearbeitet oder gelöscht werden soll. Dies kann sich - je nach Komplexität der Skripte - negativ auf die Leistung der MyTISM-Instanz auswirken.

GrooqlBOMasken

GrooqlBOMaske ist ein spezieller Untertyp von **BOMaske**. Für das Rechtesystem ist die Unterscheidung nicht wichtig aber z.B. bei Benutzung im Alarmsystem erlauben **GrooqlBOMasken** teilweise eine effizientere Abfrage was die Leistung von MyTISM in diesem Bereich verbessern kann.

GrooqlBOMasken funktionieren im Prinzip genau so wie **BOMasken** aber besitzen noch ein zusätzliches **GrooqlScript**, welches ebenfalls für die Filterung von Objekten eingesetzt wird. Informationen zur Grooql-Skriptsprache finden sich im Benutzerhandbuch im Kapitel "Grooql (Groovy Object Query Language)".

Ob ein Objekt von der Maske erfasst wird, wird für **GrooqlBOMasken** wie folgt geprüft:

1. Passt der Typ des Objekts zum in der Maske angegebenen Typ? Falls nein passt das Objekt *nicht*; falls ja, weiter mit Schritt 2.
2. Liefert das **GrooqlScript** für das Objekt "wahr" ("true")? Falls nein passt das Objekt *nicht*; falls ja, weiter mit Schritt 3.
3. Liefert das **Script** für das Objekt "wahr" ("true")? Falls nein passt das Objekt *nicht*; falls ja passt das Objekt zur Maske.

Bei Benutzung der `count()`- und `query()`- bzw. `queryBO()`-Methoden von `*BOMaske` müssen für

BOMasken mit **Script alle** zur Entitaet passenden BOs geladen und dann einzeln mit dem Script geprüft werden; bei einer größeren Menge an Objekten kann dies sehr aufwändig sein.

GrooqlBOMasken erlauben es - mittels eines aus dem **GrooqlScript** automatisch generierten OQL-Queries - die Menge der Objekte schon auf Datenbankebene vorzufiltern, was den Aufwand u.U. deutlich reduzieren kann.

OQLBOMasken

OQLBOMaske ist ein spezieller Untertyp von **BOMaske**. Für das Rechtesystem ist die Unterscheidung nicht wichtig aber z.B. bei Benutzung im Alarmsystem erlauben **OQLBOMasken** teilweise eine effizientere Abfrage was die Leistung von MyTISM in diesem Bereich verbessern kann.

OQLBOMasken funktionieren im Prinzip genau so wie **BOMasken** aber besitzen noch ein zusätzliches Attribut **WhereClauses**, welches ebenfalls für die Filterung von Objekten eingesetzt wird.

Ob ein Objekt von der Maske erfasst wird, wird für **OQLBOMasken** wie folgt geprüft:

1. Passt der Typ des Objekts zum in der Maske angegebenen Typ? Falls nein passt das Objekt *nicht*; falls ja, weiter mit Schritt 2.
2. Passt das Objekt zu den angegebenen WHERE-Klauseln? Falls nein passt das Objekt *nicht*; falls ja, weiter mit Schritt 3.
3. Liefert das **Script** für das Objekt "wahr" ("true")? Falls nein passt das Objekt *nicht*; falls ja passt das Objekt zur Maske.

Bei Benutzung der **count()**- und **query()**- bzw. **queryBO()**-Methoden von ***BOMaske** müssen für BOMasken mit **Script alle** zur Entitaet passenden BOs geladen und dann einzeln mit dem Script geprüft werden; bei einer größeren Menge an Objekten kann dies sehr aufwändig sein.

OQLBOMasken erlauben es - mittels eines mit den **WhereClauses** automatisch generierten OQL-Queries - die Menge der Objekte schon auf Datenbankebene vorzufiltern, was den Aufwand u.U. deutlich reduzieren kann.

Performance-Tipps

Script wenn möglich vermeiden

Ein **Script** für eine Maske anzugeben bedeutet *immer* und in allen Anwendungsbereichen Mehraufwand; je nach Komplexität bzw. genauer Funktionsweise des Skripts ist dieser höher oder niedriger.



Aufwändige Dinge - wie z.B. Datenbankabfragen - sollten im Skript für Masken auf jeden Fall vermieden werden!

Entitaet so "eng" wie möglich definieren

Extremes (und sehr simples) Negativbeispiel: Maske mit Entität **BO** und **Script bo.getClass() == Benutzer.class**. Hier sollte natürlich stattdessen direkt **Benutzer** als Entität der Maske gesetzt werden.

Bevorzugt OQLBOMaske oder GrooqlBOMaske verwenden

Diese beiden BOMasken-Untertypen erlauben eine effizientere Filterung was sowohl die Einzelprüfung von Objekten mit der `fits()`-Methode erleichtert als auch insb. - zum Teil erhebliche - Vorteile bei Aufruf der `count()`- und `query()`- bzw. `queryBO()`-Methoden ermöglicht. Insbesondere bei Benutzung für **Wiedervorlagen** und **BOBasierteTermine** im Alarmsystem kann das die Performance deutlich verbessern.

Zu beachten ist allerdings, dass das "free form" Groovy-Skript **Script** von BOMaske zum Teil deutlich mehr Filtermöglichkeiten bietet, als die etwas eingeschränkten Optionen via Grooql-Filtersprache oder OQL-WHERE-Klauseln. In manchen Fällen kann es also doch notwendig sein, auf ein Maske-**Script** zurückzugreifen.



Prinzipiell kann man diese Maskentypen auch ohne zusätzliche Kriterien (**WhereClauses** bzw. **GrooqlScript**), nur mit einer **Entität** - und ggf. einem **Script** - verwenden. In diesem Fall haben diese speziellen Maskentypen keinen Vorteil gegenüber einer einfachen **BOMaske**. Die Funktionalität und der entsprechende Aufwand ist dann immer derselbe, als wenn eine einfache **BOMaske** verwendet würde.

Rechtezuweisungen

In den *Rechtezuweisungen* werden alle bisher erklärten Komponenten zusammengeführt; sie definieren *wer* (via Gruppen) mit *welchen Objekten* (via BO-Maske) *wie* (via gesetzter oder nicht gesetzter Flags an der Rechtezuweisung) arbeiten darf.

Am einfachsten lassen sich die Rechtezuweisungen über den "*Rechte-Zuweisungen*"-Reiter im Gruppen-Formular bearbeiten. Im Benutzer-Formular gibt es zudem einen Reiter mit mehreren Ansichten auf die für diesen Benutzer geltenden Rechte.

Maske

Pflichtfeld - Hier wählt man die **BO-Maske** aus, die definiert, für welche Objekte man Rechte vergeben oder verweigern will.

Flags "*Lesen*", "*Schreiben*", "*Erstellen*" und "*Löschen*"

Wenn das entsprechende Flag *gesetzt* ist, wird - abhängig vom Flag "*Ablehnen*" - das entsprechende Recht für die von der Maske ausgewählten BOs gegeben oder verweigert.

Wenn das Flag *nicht gesetzt* ist, so wird das entsprechende Recht nicht "berührt".

Es muss mindestens eines der Flags *gesetzt* sein, damit die Rechtezuweisung sinnvoll ist.

Flag "*Ablehnen*"

Wenn dieses Flag *nicht gesetzt* ist, werden die durch die anderen Flags definierten Rechte *gewährt*.

Wenn dieses Flag *gesetzt* ist, werden diese Rechte allerdings *entzogen*.

Wenn für ein Objekt durch irgendeine Rechtezuweisung ein Recht entzogen wurde, so ist dies erst einmal bindend, egal durch wie viele andere Rechtezuweisungen das Recht ggf. ansonsten *gewährt* werden würde.

Flag "*Ablehnen aufheben*"

Wenn dieses Flag *gesetzt* ist, so wird jede andere Rechtezuweisung, die Rechte entzieht und deren Maske und die Flags "*Lesen*", "*Schreiben*", "*Erstellen*" und "*Löschen*" exakt mit denen dieser Rechtemaske übereinstimmen, wieder aufgehoben. Dadurch werden automatisch alle anderen Rechtezuweisungen wieder angewandt, so als ob die ablehnende Rechtezuweisung nicht existent wäre. Dies erleichtert es, eine auf Ablehnung basierende Rechteverwaltung aufzubauen, bei der man trotzdem privilegierten Benutzern leicht Rechte wieder einräumen kann, die zuvor für z.B. alle normalen Benutzer global eingeschränkt wurden.

Bemerkung

Optional - Hier kann man Erklärungen oder Erläuterungen zu dieser speziellen Rechtezuweisung unterbringen, die in der Fehlermeldung angezeigt werden, wenn eine Aktion aufgrund dieser Rechtezuweisung nicht ausgeführt werden kann.

Example 2. Benutzer dürfen Kundendaten sehen/lesen

Wir wollen in unserem Beispiel allen Benutzern etwas erlauben, also tragen wir unter *Gruppe* die bereits definierte Gruppe "Benutzer" ein.

Wir wollen Rechte für alle Kunden-BOs setzen, also tragen wir unter *Maske* unsere eben definierte BOMaske "Kunden" ein, die ja alle vorhandenen Kunden-BOs auswählt.

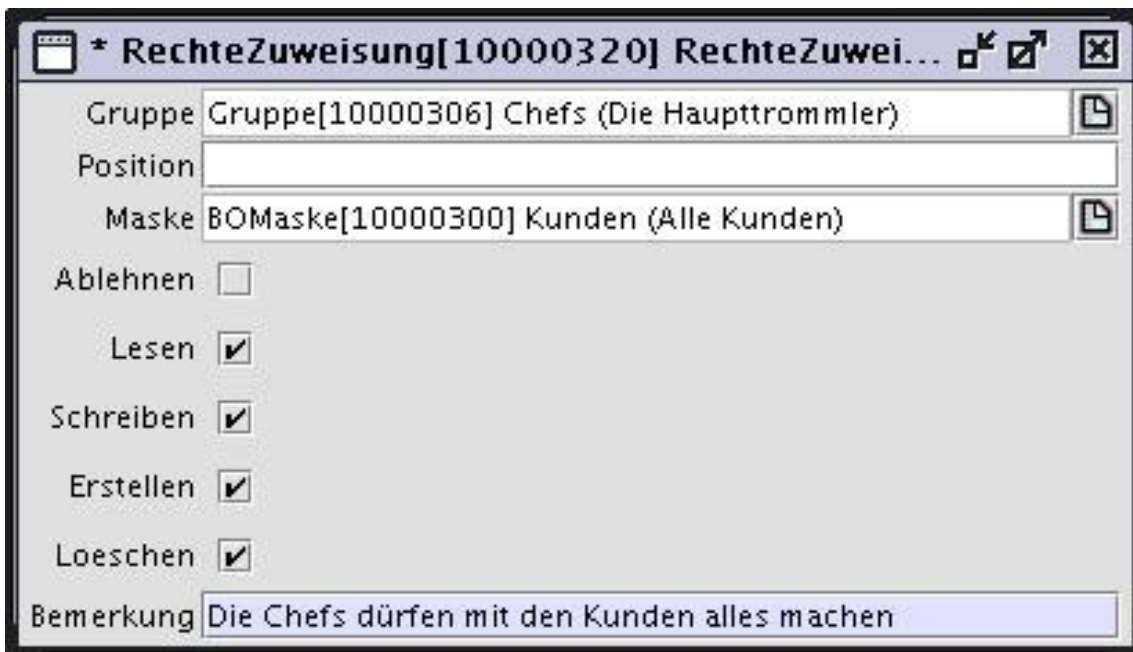
Jetzt können wir auch bestimmen, was die "Benutzer" mit allen "Kunden" machen dürfen. Da Änderungen an den Kundendaten Chefsache sind, erlauben wir nur das "Lesen".

Zu guter Letzt sollte man auch hier wieder einen kurzen "menschenslesbaren" Kommentar eintragen, der aussagt, was diese Rechtezuweisung genau bewirkt oder warum sie angelegt wurde. Dieser im Feld *Bemerkung* hinterlegte Kommentar wird auch in der Meldung angezeigt, die dem Anwender angezeigt wird, wenn er für die jeweilige Aktion keine Rechte besitzt.

Nachdem wir diese Rechtezuweisung abgespeichert haben, dürfen fortan alle Benutzer die Kundendaten einsehen!

Example 3. Chef darf Kundendaten bearbeiten

Soweit, so gut, aber irgend jemand muss ja auch die Kundendaten auf dem aktuellen Stand halten. Da das Sache der Chefs ist, benötigen diese natürlich auch entsprechende Rechte d.h. wir bauen noch eine weitere RechteZuweisung:



The screenshot shows a dialog box titled '* RechteZuweisung[10000320] RechteZuwei...'. It contains the following fields and options:

- Gruppe: Gruppe[10000306] Chefs (Die Haupttrommler)
- Position: (empty)
- Maske: BOMaske[10000300] Kunden (Alle Kunden)
- Ablehnen:
- Lesen:
- Schreiben:
- Erstellen:
- Loeschen:
- Bemerkung: Die Chefs dürfen mit den Kunden alles machen

Unter *Gruppe* wählen wir logischerweise "Chefs".

Da wir auch hier wieder etwas für alle Kunden-BOs definieren wollen, tragen wir unter *Maske* wieder unsere eben definierte BO-Maske "Kunden" ein.

An Rechten vergeben wir jetzt aber wesentlich mehr, nämlich alles was möglich ist:

- Lesen - Die Chefs dürfen natürlich auch Lesen (dieses Recht hier nochmal zu vergeben ist rein theoretisch nicht nötig, da wir das Lesen ja bereits vorher für alle Benutzer, also auch die Chefs, die ja ebenfalls auch in der Gruppe "Benutzer" sein sollten, erlaubt haben).
- Schreiben - Die Chefs dürfen bestehende Kunden-BOs bearbeiten und wieder abspeichern.
- Erstellen - Die Chefs dürfen auch vollkommen neue Kunden(-BOs) anlegen.
- Löschen - Die Chefs dürfen vorhandene Kunden(-BOs) in der Datenbank als "gelöscht" markieren.
- Ablehnen - Alle oben genannten Rechte bekommen eine umgekehrte Bedeutung, das heißt, würden Lesen, Schreiben, Erstellen oder Löschen *verbieten* - das setzen wir natürlich nicht.

Rechte vergeben

Es gibt prinzipiell zwei Methoden, um Rechte in einem MyTISM-System zu vergeben.

Negativliste

Standardmäßig alles erlauben und einiges selektiv via Negativliste verbieten.

Positivliste

Standardmäßig alles verbieten und einiges selektiv via Positivliste erlauben.

Negativliste - Selektiv verbieten

Bei dieser Methode werden einer Gruppe erst einmal *alle Rechte* für *alle Objekte* gegeben. Danach werden diese Rechte selektiv eingeschränkt um bestimmte Dinge zu verbieten.

Vorteil

Schnell und einfach einzurichten; Benutzer bekommen auf einfache Weise alle Rechte, die sie benötigen, um mit einem System zu arbeiten.

Im Minimalfall reichen eine Gruppe, eine BO-Maske und eine Rechtezuweisung.

Nachteil

Benutzer bekommen standardmäßig viele Rechte, die sie nicht benötigen oder die sogar gefährlich sind.

Wenn vergessen wird, vertrauliche Daten mit einer das Lesen verbietenden Rechtezuweisung zu schützen, können diese von allen Benutzern eingesehen werden.
Wenn vergessen wird, wichtige Daten mit mit einer das Löschen verbietenden Rechtezuweisung zu schützen, können diese von allen Benutzern absichtlich oder versehentlich als gelöscht markiert werden.

Durch die Einführung eines neuen Features im Rechtesystem (vgl. Ticket 188879251 „Verweigern / DENY Rechtezuweisungen wieder aufheben können“) ist diese Art der Rechtevergabe nun einfacher und flexibler geworden und bietet sich daher als Standardvorgehen an.

Das „Ablehnen aufheben“-Flag wurde neu eingeführt. Wenn eine Rechtezuweisung dieses Flag gesetzt hat **und** die *gleiche* BO-Maske sowie die *gleichen* Rechte wie eine „Ablehnen“-Rechtezuweisung hat, hebt diese Rechtezuweisung die ablehnende auf, sofern einem Benutzer über seine Gruppenmitgliedschaften beide Rechtezuweisungen zugeteilt sind. Die übrigen Rechte greifen dann wieder „normal“ für diesen Benutzer.

So können ablehnende Rechte für Admins oder andere spezielle Gruppen wieder aufgehoben und durch weniger restriktive Verweigerungsrechte (oder gar keine) ersetzt werden, ohne dass diese Benutzer aus den normalen Gruppen, die diese ablehnenden Rechte haben, entfernt werden müssen.

Dies entspricht der Realität besser, da Benutzer in der Regel viele Berechtigungen haben und nur wenige Einschränkungen benötigen. Es erspart die Notwendigkeit, eine große Anzahl von Rechten ermitteln, vergeben und einrichten zu müssen, damit Benutzer im System arbeiten können (s. „Positivliste“). In der Vergangenheit wurde das Rechtssystem oft nur mangelhaft eingerichtet, weil die Konfiguration schwierig war und ablehnende Rechte bindend und nicht mehr aufhebbar waren.

Hinweis: Es kann ratsam sein, einen Alarm auf die Erzeugung einer Rechtezuweisung oder die Änderung des „Ablehnen aufheben“-Flags anzusetzen, da diese Konfiguration sicherheitskritisch sein kann.

Positivliste - Selektiv erlauben

Bei dieser Methode erhalten Gruppen *nur für genau die Objekte, die sie für ihre Arbeit benötigen nur genau die erforderlichen Rechte*.

Vorteil

Benutzer bekommen nur die passenden Rechte.

Vertrauliche Daten sind standardmäßig vor Einsicht geschützt.
Wichtige Daten können nicht von jedem gelöscht oder verändert werden.

Nachteil

Je nach System komplex und aufwändig einzurichten. Es müssen u.U. viele Gruppen, BO-Masken und Rechtezuweisungen angelegt und zugewiesen werden. Welche Gruppen man anlegen soll und welche Rechte sie bekommen sollen kann schwierig zu entscheiden sein.

Allein die Standardgruppe "RG_Solstice_Login" (Benutzer mit minimalen Rechten zum Login mit Solstice) hat 15 Rechtezuweisungen mit ebensovielen BO-Masken.

Zusammenhang mit der Methode `BO#isReadOnly(AttributeI)`

Die Methode `BO#isReadOnly(AttributeI)` ist genau wie `BO#isMandatory(AttributeI)` lediglich für die UI sowie den Zugriff via Schema-Attribute gedacht.

Die default-Implementierung in der Klasse BO, die von Subklassen überschrieben werden kann, fragt zunächst, ob das `AttributeI#isReadOnly` ist und anschließend das Rechtesystem, das intern über den (an dieser Stelle client-seitigen) `PermissionHandler` abgewickelt wird.



Leider ist die Schreibweise der Methoden `BO#isReadOnly(AttributeI)` und `AttributeI#isReadOnly` (noch) nicht konsistent.

`CBOAttribute` ist aktuell die einzige relevante Implementierung von `AttributeI`, neben dem `ScriptedAttribute`, das aber nur in der UI via `<virtualProperty>` definiert wird, daher konzentrieren wir uns hier auf das `CBOAttribute`.

Ein `CBOAttribute` gibt in zwei Fällen `true` in `#isReadOnly` zurück: 1. Wenn es im Schema explizit via `readonly="true"` als read-only definiert wurde, was aber in der Praxis nie vorkommt, da es ziemlich unsinnig ist, ein Attribut global gegen das Schreiben zu sperren; das `readonly`-Attribut wird eigentlich nur zum Definieren eines vattrr als `readonly="false"` verwendet, um dieses virtuell schreibbar zu machen. 2. Wenn es virtuell ist, und es nicht explizit auf `readonly="false"` gesetzt wurde.

Falls das `AttributeI#isReadOnly` nicht `true` zurückgegeben hat, wird anschließend das Rechtesystem über den client-seitigen `PermissionHandlerI`, d.h. via `BO#isWritable(AttributeI)`, befragt und dessen Rückgabewert natürlich negiert. In dieser Abfrage kommen dann auch die Rechtemasken für die Gruppen zur Anwendung, in denen der aktuelle Benutzer ist.

Das heißt effektiv, dass das Ergebnis von `#isReadOnly(AttributeI)` normalerweise nie von den definierten Rechten abweichen sollte, sofern überall immer korrekt das Resultat von `super.isReadOnly`, also im Endeffekt von `#isReadOnly(AttributeI)` mitberücksichtigt und nicht durch eigenmächtig mit `false` überschrieben wird, was spätestens beim Speichern dann aber eh zum Scheitern führen würde, da die Rechte im server-seitigen `PermissionHandlerI` nochmal abgeprüft werden.

Allerdings wird in der Praxis oftmals das Leserecht nicht immer über die Rechte eingeschränkt, sondern programmatisch im Code des BOs über eine überschriebene `BO#isReadOnly(AttributeI)` Methode. Der Sinn davon ist, dass die Logik des Programms es erfordern kann, dass bestimmte Änderungen in bestimmten Situationen oder Zuständen verweigert werden müssen.

Das Abprüfen der Mitgliedschaft in bestimmten Gruppen in der überschriebenen `BO#isReadOnly(AttributeI)` Methode ist ganz genau genommen eigentlich falsch, sofern nicht noch weitere Bedingungen dazu kommen, und sollte insofern eigentlich besser direkt über das Rechtesystem abgehandelt werden. Aus historischen Gründen ist das jedoch fast nie der Fall bzw. so gemacht worden.

Zudem ist die Konfiguration von BO-Masken für einzelne Attribute und die Verknüpfung dieser

Masken mit Rechtemasken und Gruppen zwar möglich, aber eher unüblich, da sehr aufwändig (falls in der `#isReadOnly(AttributeI)` das AttributeI gar nicht geprüft wird und es eine "globale" read-only Einstellung ist, ist das natürlich etwas einfacher).

Insofern passieren diese read-only Checks sehr oft im Code der entsprechenden BO-Klasse in der `#isReadOnly(AttributeI)` Methode.

Plugins - Zuschaltbare Komponenten im Client

Eingabefeld für Befehle in der Aktionsleiste einblenden

Dieses Eingabefeld ist im Grunde ein Plugin der Solstice GUI um Befehle an den Client abzusetzen. Die nachfolgenden Liste zeigt alle Befehlsmöglichkeiten, welche in diesem Eingabefeld durchgeführt werden können.

- [0-9]+ öffnet das Objekt mit der entsprechenden Id. Zum Öffnen wird das Standardformular benutzt.
- :[0-9] öffnet das Objekt mit der entsprechenden Id. Zum Öffnen wird das Formular mit der hinter dem : stehenden Id genutzt (Falls die Berechtigungen dies erlauben).
- [0-9]+:auto öffnet das Objekt mit der entsprechenden Id. Zum Öffnen wird das automatisch generierte Formular genutzt.
- #[0-9]+ öffnet das Lesezeichen mit der entsprechenden Id
- #[0-9]+:.* öffnet das Lesezeichen mit der entsprechenden Id und führt direkt die angegebene Query aus.

Die Befehlseingabe befindet sich am rechten Rand der Aktionsleiste (Leiste unter der Menüleiste). Um die Befehlseingabe zu aktivieren, muss die folgende Skriptzeile im Voreinstellungsskript des Benutzers im Tag Configuration → Profile angegeben werden. Nach dem Hinzufügen dieser Zeile ist ein Neuanmelden nötig.

```
<Plugin class="de.ipcon.form.ClientToolBarManager" name="Main"
orientation="HORIZONTAL" position="NORTH" rollover="true" floatable="true"
cli="true"/>
```

Server-Gesundheitsanzeige einblenden

Zur Aktivierung des Plugins fügt man folgende Zeile im Voreinstellungsskript des Benutzers im Tag Configuration → Profile ein. Nach dem Hinzufügen dieser Zeile ist ein Neuanmelden nötig.

```
<Plugin class="de.ipcon.form.ServerHealthMonitor"/>
```

Das Plugin wird unten rechts in der Statusleiste eingeblendet. Das Plugin zeigt auf an synchronisierenden Servern angemeldeten Clients als Beschriftung den aktuellen Synchronisations-Verzug in Minuten an. Auf autoritativen Servern wird als Beschriftung ein statischer Text angezeigt. Klickt man den Knopf des Plugins an, so öffnet sich ein Dialogfeld mit weiteren Kenndaten des Servers, an dem man angemeldet ist. Die Informationen über den Server

werden standardmäßig alle 60 Sekunden an die Clients, die das Plugin aktiviert haben, per DBManServerHealthEvent verteilt.

Fehlerbehebung

Der Server startet nicht wegen angeblich fehlenden Hostnamens

Situation

Der Server bricht den Startvorgang mit folgender Fehlermeldung ab:

```
[DBMan.boot] DBMan          main(203)- Problem during DBMan initialization.  
java.lang.IllegalArgumentException: No usable hostname found, is the network  
connection ok?
```

Fehlerbehebung

Die Datei `/etc/hosts` sieht vermutlich so oder ähnlich aus:

```
127.0.0.1    localhost  
127.0.1.1    ha19000
```

Damit MyTISM korrekt laufen kann, muss die IP-Adresse des Servers zu etwas anderem als `localhost` auflösbar sein. Oft ist die Lösung dazu einfach, den Eintrag für die `127.0.0.1` in der `/etc/hosts` noch um den Hostnamen zu ergänzen:

```
127.0.0.1    ha19000 localhost  
127.0.1.1    ha19000
```

Danach sollte der MyTISM-Server korrekt hochfahren.

Der Integrity-Check bricht ab wegen fehlender Ressourcen

Situation: Der Integrity-Check bricht ab mit der Meldung, dass keine Dateien mehr geöffnet werden können oder der shared memory aufgebraucht sei.

Fehlerbehebung

Der Integrity-Check läuft parallel in vielen Threads und mit vielen gleichzeitigen Verbindungen zur Datenbank. Deswegen braucht die Datenbank beim Startup genug Ressourcen. Es ist allgemein aber auch gut, wenn man das Betriebssystem und PostgreSQL tuned, damit auch beim laufenden Betrieb nicht irgendwann unter hoher Last die Ressourcen fehlen.

PostgreSQL-Konfiguration

Die `shared_buffers` sollten vom Standard 32 MB auf mindestens 512 MB (bis hin zu 25 % des RAM) erhöht werden. Datei `/etc/postgresql/10/main/postgresql.conf` (oder entsprechende Version) ändern. Anschließend PostgreSQL neu starten via `service postgresql restart`.

Falls dem Betriebssystem der Speicher ausgeht und der OOM-Killer zuschlägt, kann man noch schauen ob man die Anzahl von `max_connections` z.B. von 100 auf 50 reduziert.

Linux-Konfiguration

Anzahl gleichzeitig offener Dateien bzw. File-Descriptors sollte erhöht werden vom Standard 65536 auf z.B. das Zehnfache. Einstellung `fs.file-max` in Datei `/etc/sysctl.conf` auf 655360 erhöhen. Anschließend Kommando `sysctl -p` rufen.

Doppelte IDs in der Datenbank korrigieren

Situation

Der Integrity-Check beim Serverstart meldet doppelte IDs bzw. es ist anderweitig aufgefallen, dass in der Datenbank gleiche IDs für verschiedene Objekte mehrfach vergeben wurden (kann z.B. passieren, wenn ein Backup einer Datenbank eingespielt wurde, aber vergessen wurde, vor dem darauffolgenden Serverstart die Dateien `.init-keygen` und `.checked-firstnodestart` zu löschen).

Fehlerbehebung

Die folgenden Schritte sind bei synchronisierenden Systemen alle auszuführen. Wenn nur ein einzelner Server zu fixen ist, können die "Synchronisierender Server"-Schritte weggelassen werden. Alle Kommandos müssen im jeweiligen Projektverzeichnis stehend ausgeführt werden. Ggfs. heißt das Kommando statt `./mytism` auch `PROJEKTNAME`, also z.B. `./oashi`.

1. Synchronisierender Server: Server stoppen `./mytism stop_mytism`
2. Autoritativer Server: Server stoppen `./mytism stop_mytism`
3. Autoritativer Server: Backup ziehen `./mytism backup` (Nur zur Sicherheit)
4. Autoritativer Server: Fixer laufen lassen `./mytism run de.ipcon.db.tools.DoubleIdFixer PROJEKTINSTANZ --repairAll fix_double_ids.sql` (PROJEKTINSTANZ z.B. `.oashi`)
5. Autoritativer Server: checked-*-Dateien löschen `rm .checked-*`
6. Autoritativer Server: Generiertes SQL-Script einspielen `psql -U postgres DATENBANKNAME < fix_double_ids.sql`
7. Synchronisierender Server: Backup ziehen (Nur zur Sicherheit)
8. Autoritativer Server: bi-tabelle leeren `psql -U postgres DATENBANKNAME`, dann dort `delete from bi`
9. Autoritativer Server: Server starten `./mytism start_mytism` (Diverse Aufräumarbeiten werden gemacht)
10. Autoritativer Server: Server stoppen `./mytism stop_mytism`
11. Autoritativer Server: Backup ziehen `./mytism backup`
12. Backup zum synchronisierenden Server kopieren
13. Autoritativer Server: Server starten `./mytism start_mytism`
14. Falls dort ein Grails läuft, dieses durchstarten `./mytism stop_grails`, dann `./mytism start_grails`
15. Synchronisierender Server: Backup einspielen `./mytism restore BACKUPDATEINAME`
16. Synchronisierender Server: `.checked-*-Dateien` löschen `rm .checked-*`
17. Synchronisierender Server: `.init-keygen` löschen
18. Synchronisierender Server: `.checked-firstnodestart` löschen
19. Synchronisierender Server: Server starten `./mytism start_mytism`

Beide Server sollten danach ohne Fehlermeldungen starten:

- auf der Status-Webseite kontrollieren, ob auf dem autoritativen Server der Sync-Account des synchronisierenden Servers angemeldet ist
- im logs/daily.log des synchronisierenden Servers kontrollieren, ob die Synchronisation läuft, d.h. nach Meldungen wie `INFO 10:15:59.355 [shi oashi.oashi.com] SyncService logSyncLocalToRemote(637)- >>>> BT[12345678] from 11-12-12 10:15:57 SrvCrea 11-12-12 10:15:57` bzw. `logSyncRemoteToLocal()` Ausschau halten.
- Ggfs. selbst mal eine Änderung an einem Objekt auf einem der Server machen. = FAQ - Immer wiederkehrende Fragen und deren Beantwortung

Wie starte ich MyTISM durch, ohne dass der Benutzer es mitbekommt (restart fast / restart silent)?

Situation

Wurde für einen Bugfix z.B. eine jar ausgetauscht, reicht es evtl., dass sich die Clients ab und wieder anmelden. Wenn es aber ein Server-seitiges Problem ist, MUSS der MyTISM-Server durchgestartet werden, damit der neue Code/Fix zur Anwendung kommt.

Sofern die Clients von dem Problem nicht direkt betroffen sind, kann man in Betracht ziehen den MyTISM-Server "silent" durchzustarten (still und leise, ohne dass den angemeldeten Clients eine unnötige Meldung über den Server-Shutdown angezeigt wird).

Lösung

Situation 1: Der MyTISM-Server wird als systemd.service gestartet

Üblicherweise sollte ein solcher MyTISM-Service so konfiguriert sein, dass er sich immer automatisch neustartet (**Restart=always**), falls er unerwartet endet (also nicht explizit mit 'service PROJEKTKUERZEL stop' angehalten wurde). Dies passiert nach spätestens 100ms (laut "man page" der Defaultwert), sofern nicht in der Konfiguration was anderes angegeben ist (z.B. **RestartSec=1800s**).

Die systemd-Konfiguration für den MyTISM-Dienst liegt auf dem jeweiligen Server üblicherweise im Ordner `/etc/systemd/system` und kann wie folgt angesehen werden:

```
less /etc/systemd/system/[mytism-]PROJEKTKUERZEL.service
```

Somit genügt es den Service mit folgendem Kommando "abzuschiessen" und nach 100ms wird er automatisch neugestartet:

```
./mytism stop fast
```

Situation 2: Der MyTISM-Server wird mittels mytism-Skript gestartet

```
./mytism restart fast
```