



NMS: BENUTZERHANDBUCH

S T A H L
netzmanagement



Martin Eckardt, Robert Kais, C. Beimler

Stahl Netzmanagement GmbH
Veröffentlicht: 18.08.2026
Version: 1.63.0

Inhaltsverzeichnis

Einleitung	5
1. Schritte.....	6
Anmeldung.....	6
Admin Rolle.....	6
Lizenz einspielen.....	6
Benutzer anlegen.....	6
Admin Passwort ändern	6
Grundkonfiguration	6
Aufgaben in der Aufgabenverwaltung anlegen.....	6
Gerätemodelle pflegen.....	6
Hersteller-MAC-Adressen-Prefixe importieren.....	6
Gruppenstruktur anlegen	7
1. Gerät anbinden	7
Allgemein	8
Benutzerverwaltung/Rollen	8
Berechtigungen.....	8
Zweifaktorauthentifizierung für Benutzer aktivieren	9
Zweifaktorauthentifizierung erzwingen	9
Gruppen.....	10
Dashboards.....	10
Folgende Dashboard-Widgets stehen zur Verfügung:.....	11
Einstellungen	14
Backup	14
Wiederherstellung von Backups (Restore)	15
Globale Suchfunktion (optional)	15
Benutzerdefinierte CSS-Datei zur Anpassung des NMS-Designs.....	16
Auswertungen.....	17
Übersicht	17
Auswertung anzeigen.....	17
Parameter (Variablen).....	17
Export	17
Importieren.....	17
Automatische Aktualisierung	17
Versand per Zeitplan	18
Inventarisierung.....	18
Geräte.....	18

Wartungsmodus	18
Import	19
Export	19
Unzugewiesene Geräte	19
Einbauobjekte	19
Gerätevorlagen	19
SIM-Karten	20
Import	20
Export	20
Geräteverwaltung	21
OpenVPN Konfigurationen	21
SSH TerminalServer Funktion	21
Unterstützte Verwaltungsprotokolle	22
Konfiguration von Protokolloptionen	22
Liste der Protokolloptionen	23
Gerätetreiber	24
Allgemeine Hinweise zur Treibernutzung	24
Konfiguration von Treiberoptionen	24
Verfügbare Gerätetreiber	27
APS-R-PoE	27
CD9-X	27
Cisco (IE-3x00 Series)	28
Generisches Linux-OS	29
Generischer Ping	29
Hirschmann HIOS	30
Hirschmann OCTOPUS	30
LANCOM (LCOS SX v4)	31
Moxa (EDS Series)	32
Netmodule (alle Modelle)	33
Netmodule (FW ≤4.0)	38
OpenWrt	41
Ping über ein Zwischengerät	43
SES-imagotag ePaper Services	43
Stahl OpenVPN-Dienstanbieter	44
Stahl OpenVPN-Instanz	45
Stahl Zertifizierungsdienste (EasyRSA v2/3)	46
Teltonika RUT	47

Trapeze IBIS-IP	48
Tronteq (10-port ES Series JSON-API).....	49
Tronteq (10-port ES Series OpenAPI)	49
Wavecom	50
Westermo (RT SW 6).....	51
Westermo (WeOS 4).....	52
Konfigurationsverteilung	53
Funktionsbeschreibung	53
Konfiguration.....	53
Konfigurationsdatei.....	53
Template	53
Template-Schnipsel	53
Skriptausdrücke.....	54
Objekte und Funktionen	55
Zugangsdaten.....	55
Kunden / Gruppen	58
Gerät	60
Gerätemodell	62
IP-Adresse.....	64
Objekt (ehem. „Fahrzeug“).....	65
Optionsverweis	67
Skript	68
SIM-Karten	70
Zeichenketten	72
Subnetz	78
Template	85
Time.....	88
Variablen	89
Kommentare	90
Beispiele.....	91
Verwaltung.....	94
Firmware	94
Passwörter.....	94
SIM-Karten	94
Health Check	94
IP Address Management (IPAM)	94
Administration.....	95

Aufgabenverwaltung.....	95
E-Mail Benachrichtigungen.....	95
CLI-Interface	96
Benutzerverwaltung	96
Queue Management.....	96
Migrations	97
Events.....	98
Anlagen.....	99
ER-Diagramm / Datenbankschema	100

Einleitung



1. Schritte

Anmeldung

1. URL in einem Browser aufrufen (unterstützt: Mozilla Firefox, Google Chrome)
2. Erstanmeldung mit Benutzername **admin** und Passwort **nmsadmin**

Admin Rolle

Sie sollten unter **Rollen & Berechtigungen** der **Admin** Rolle sämtliche Berechtigungen zuweisen.

Lizenz einspielen

Navigieren Sie auf die Seite Support und dort auf den Tab Lizenz. Übermitteln Sie die ID an die Stahl Netzmanagement GmbH und lassen Sie sich eine Lizenz zusenden. Laden Sie anschließend hier Ihre Lizenz in das System.

Benutzer anlegen

Legen Sie unter Verwaltung -> Benutzer einen neuen Benutzer an und weisen Sie diesem eine entsprechende Rolle zu.

Admin Passwort ändern

Ändern Sie unter „Mein Account“ ihr Passwort für den per default angelegten Admin Benutzer.

Grundkonfiguration

Navigieren Sie zu Verwaltung -> Einstellungen und hinterlegen Sie dort Ihre E-Mail Adresse.

Hinterlegen Sie einen E-Mail Server in der Konfigurationsdatei common/config/main-local.php

Aufgaben in der Aufgabenverwaltung anlegen

Navigieren Sie zu Verwaltung -> Aufgabenverwaltung und legen Sie z.B. eine Aufgabe für den Abruf neuer Versionen von einer Herstellerseite an.

Gerätemodelle pflegen

Navigieren Sie zu Verwaltung -> Gerätemodelle und legen Sie hier Ihre verwendeten Gerätetypen an.

Hersteller-MAC-Adressen-Prefixe importieren

Auf die Seite /devicetype/import-mac-prefixes navigieren und anschließend von der [Webseite des IEEE](#) die Datei **oui.csv** herunterladen und einspielen:

⬇️ DOWNLOAD

In order to download the entire public listing for a registry, please select either the text or CSV file below.

Note: the CSV files below use the UTF-8 character set. Please open the files in an application that supports UTF-8.

1. **MAC Address Block Large (MA-L)**   
2. **MAC Address Block Medium (MA-M)**  
3. **MAC Address Block Small (MA-S)**  
4. **Company ID**  
5. **EtherType™**  
6. **ManufacturerID**   [XDL](#)
7. **IEEE 802.16 Operator ID**  

Gruppenstruktur anlegen

Navigieren Sie zu Gruppen und legen Sie hier eine Gruppenhierarchie an. Machen Sie sich sorgfältig Gedanken über die Struktur. Diese hilft Ihnen Objekte zu strukturieren und schnell wieder zu finden.

Außerdem können Sie Benutzer-Berechtigungen an diese Gruppen knüpfen und somit den Zugriff auf Elemente im System einschränken.

1. Gerät anbinden

Gehen Sie zur Geräteübersicht und legen Sie ihr erstes Gerät an.

Allgemein

Benutzerverwaltung/Rollen

Einem Benutzer können eine oder mehrere Rollen zugewiesen werden. Die Berechtigungen der Benutzer definieren sich ausschließlich über die zugewiesenen Rollen.

Die Benutzerverwaltung steht unter dem Menüpunkt „Verwaltung“ -> „Benutzer“ zur Verfügung. Hier können Benutzer erstellt, bearbeitet oder gelöscht werden.

Die Rollen und Berechtigungen stehen unter dem Menüpunkt „Verwaltung“ -> „Rollen & Berechtigungen“ zur Verfügung. Hier können über den Button „Neue Rolle“ neue Rollen angelegt und Ihnen eine oder mehrere Berechtigungen zugewiesen werden.

Hinweis: Mit einem Doppelklick auf die entsprechende Rolle kann diese geöffnet und bearbeitet werden.

Hinweis: Sollte eine Rolle gelöscht werden, welche bereits einem Benutzer zugewiesen wurde, so wird dem Benutzer diese Rolle entzogen.

Das Feld „Dashboard“ definiert das nach dem Anmelden standardmäßig angezeigte Dashboard. Sollte das Feld leergelassen werden, wird das globale **Standard**-Dashboard verwendet.

Berechtigungen

Berechtigung (Suffix)	Beschreibung
-Admin	Globaler administrativer Zugriff
-Assign	Dokumente zuweisen
-BetaAccess	Zugriff auf Beta-Funktionen
-ContentRead	Dokumentinhalt lesen
-ContentWrite	Dokumentinhalt schreiben
-Create	Erstellen
-Delete	Löschen
-Download	Firmware-Images herunterladen
-Duplicate	Duplizieren
-GenerateFile	Generierung von Dateien
-HistoryList	Auflistung der Historie
-HistoryRead	Lesen der Daten von Historien
-Install	Erweiterungen installieren
-LicenseRead	Einsicht in Lizenzinformationen
-LicenseUpload	Hochladen einer Lizenzdatei
-List	Auflistung
-ListMap	Kartenansicht mit Standorten von Geräten
-LoginDownload	Login in der Download-Applikation
-LoginFrontend	Login in der Frontend-Applikation
-LoginRest	Login in der Rest-Applikation
-Logout	Angemeldeten Benutzer abmelden
-Maintenance	Wartungsmodus bei Geräten ein/ausschalten
-Manage	Management-GUI von Geräten öffnen
-MetaRead	Metadaten lesen
-MetaWrite	Metadaten bearbeiten
-ModuleRead	Informationen über Hardware-Module von Geräten anzeigen
-Occupancy	Anzeige von Besetztgradinformationen
-PhpConfig	Einsicht über die PHP-Konfiguration
-Push	Aufgaben manuell ausführen
-Query	Suche aller zugänglichen Datenobjekte

-Read	Lesen von Daten
-Reboot	Neustarten von Geräten
-RemoveAuthenticator	Entfernen des Authentifikators von anderen Benutzern
-ReplaceDevice	Erlaubt das Austauschen von Geräten
-Rollback	Rollback durchführen
-ServerInfo	Informationen über den Server einsehen
-SnippetAssign	Snippets an Templates/Snippets zuordnen
-SnippetHistoryRead	Lesen der Schnipselzuweisungshistorien von Vorlagen
-SnippetHistoryRollback	Rollback der Schnipselzuweisungen von Vorlagen durchführen
-SupportInfo	Support-Info von Geräten herunterladen
-Terminal	Öffnen von Terminalsitzungen im Webbrowser
-TrafficReset	Datenvolumen von SIM-Karten zurücksetzen
-Uninstall	Erweiterungen deinstallieren
-UpdateConfig	Konfigurationen von Geräten aktualisieren
-UpdateFirmware	Firmware von Geräten aktualisieren
-VendorPrefixImport	MAC-Präfixe zur Identifizierung von Herstellern importieren
-Write	Bearbeiten von Daten
-WriteCustomer	Einstellungen für zugeordnete Gruppen bearbeiten
-WriteGlobal	Einstellungen global bearbeiten
-WriteUser	Einstellungen für eigenen Benutzer bearbeiten

Zweifaktorauthentifizierung für Benutzer aktivieren

Um bei der Authentifizierung für den aktuellen Benutzer einen OTP-Code einzurichten navigieren Sie zu Ihrer „Mein Account“ Seite und klicken Sie auf den Button „Zweifaktorauthentifizierung einrichten“:

Mein Account

 **Zweifaktorauthentifizierung einrichten**

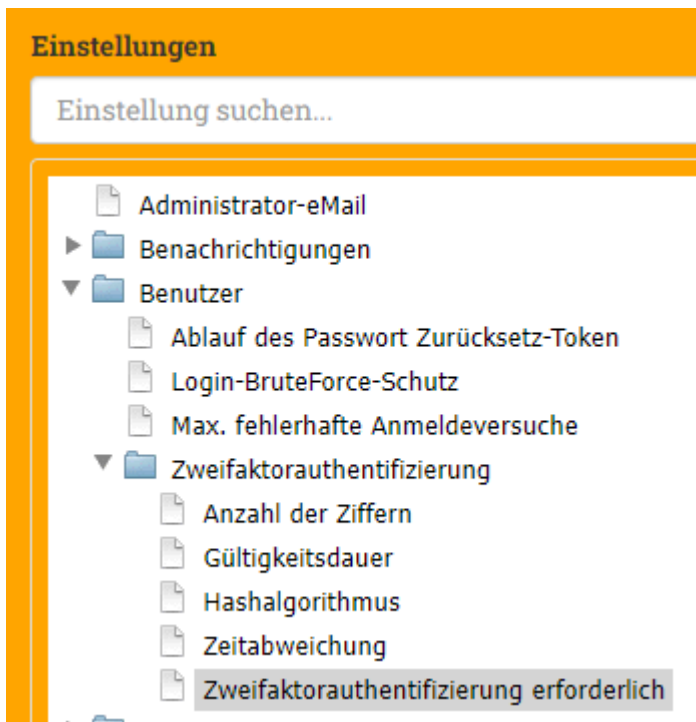
Scannen Sie anschließend den QR-Code mit einer Authenticator App Ihrer Wahl und geben Sie als Bestätigung den Code in das Eingabefeld ein.

Um den Code für sich zu entfernen, navigieren Sie zu Ihrem Konto und klicken Sie auf „Zweifaktorauthentifizierung entfernen“.

Um den 2. Faktor für einen anderen Benutzer zu entfernen benötigen Sie das Recht „UserRemoveAuthenticator“, navigieren Sie anschließend zu dem Benutzer in der Benutzerverwaltung und klicken Sie hier auf „Zweifaktorauthentifizierung entfernen“.

Zweifaktorauthentifizierung erzwingen

Wenn Sie die 2-Faktor Authentifizierung für **alle** Benutzer verpflichtend machen möchten, können Sie diese in den globalen Einstellungen unter Benutzer -> Zweifaktorauthentifizierung -> Zweifaktorauthentifizierung erforderlich aktivieren:



Gruppen

Gruppen sind für eine Strukturierung der weiteren Objekte im System gedacht und können zur Gruppierung von Informationen verwendet werden. Dieser Bereich wurde im Verlauf der Produktentwicklung von „Kunden“ in „Gruppen“ umbenannt, da die Strukturierung universell ist und nicht nur zur Verwaltung einer Kundenzuordnung dienen kann.

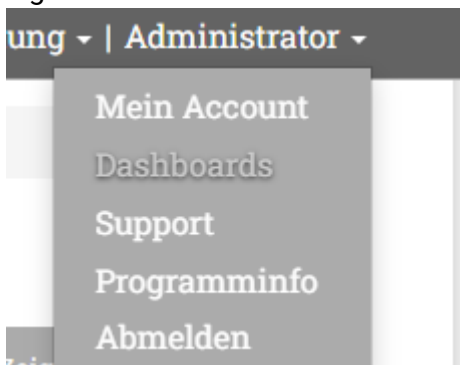
Zeitgleich mit der Umbenennung wurde die Darstellung von einer flachen Liste in eine Baumstruktur gewandelt und hinsichtlich Sicht- und Bearbeitbarkeit der den Gruppen zugeordneten Unterobjekte an das Rechte- und Rollenmodell gekoppelt.

Dashboards

Sie können das allgemeine Dashboard bearbeiten oder sich ein persönliches Dashboard anlegen.

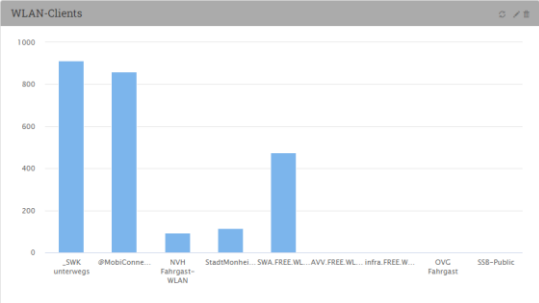
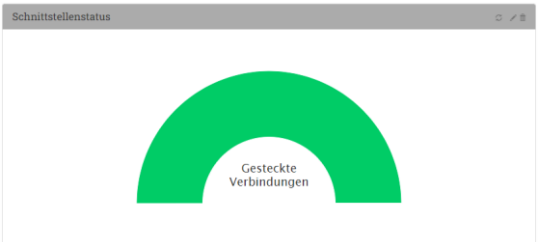
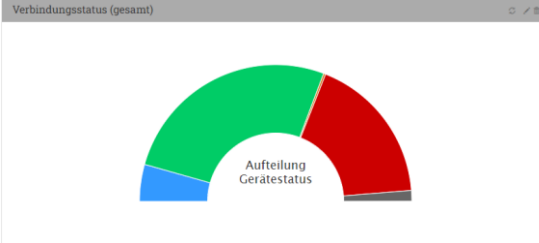
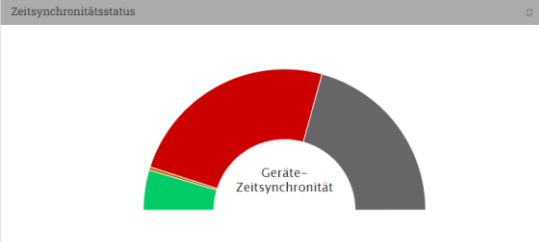
Auf dem Dashboard können Sie über vordefinierte Widgets ihre eigene Übersichtsseite flexibel selbst gestalten.

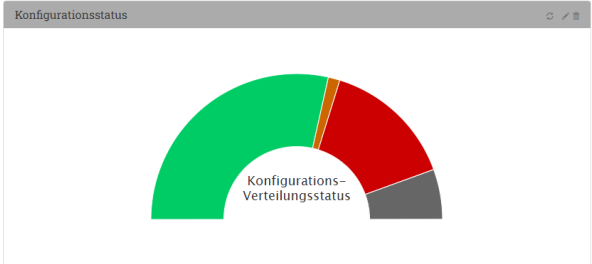
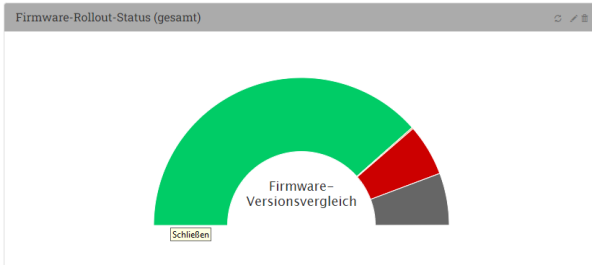
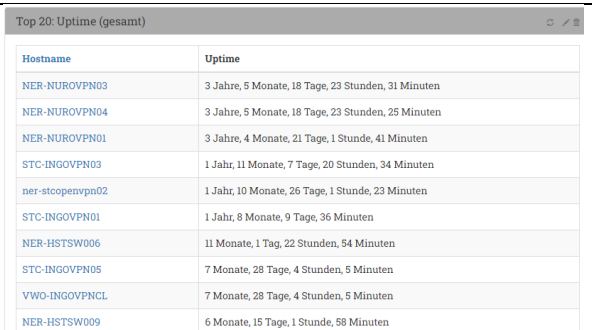
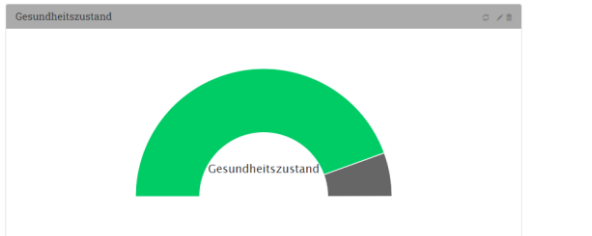
Der Zugriff auf die Dashboardverwaltung erfolgt über den Menüpunkt Dashboards unter dem eigenen Benutzernamen:

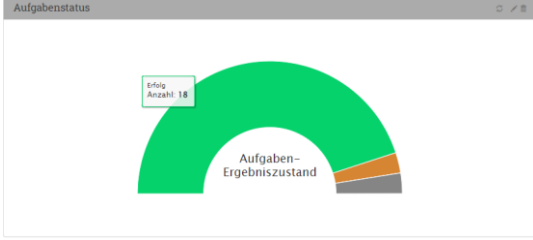


Sie benötigen hierfür das Recht DashboardList und DashboardCreate.

Folgende Dashboard-Widgets stehen zur Verfügung:

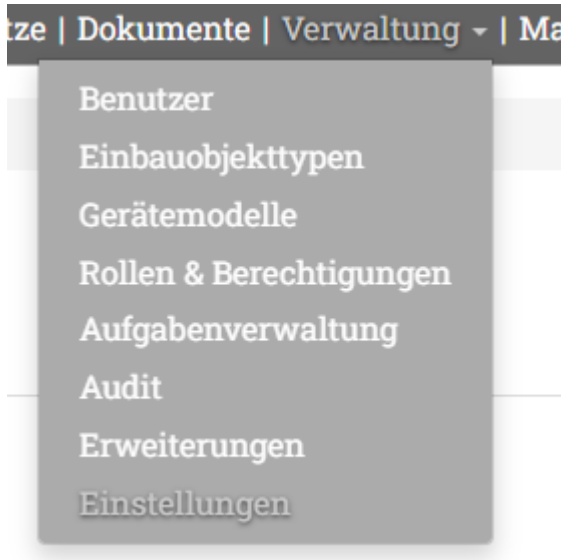
Name	Beschreibung	Parameter	Beispiel
Zusammenfassung	Eine Übersicht der Gruppen und deren Status	Gruppe, Gruppierung, Anzahl Elemente, Status-Spalten	
WLAN-Clients	Zeigt eine Übersicht der Anzahl an verbundenen WLAN-Clients pro SSID	Gruppe, SSID	
Schnittstellenstatus	Zeigt den Status der Interfaces an. In der Konfiguration kann der Soll-Zustand angegeben werden.	Gruppe, Konfigurationsvorlage, Zwingend aktive Schnittstellen, Zwingend deaktivierte Schnittstellen	
Verbindungsstatus	Zeigt eine Grafik zur Übersicht der Gerätestati.	Gruppe, Konfigurationsvorlage	
Zeitsynchronitätsstatus	Zeigt eine Übersicht wieviele Geräte eine aktuelle Uhrzeit gesetzt haben.	Gruppe, Konfigurationsvorlage	

Konfigurationsstatus	Zeigt eine Übersicht wieviele Geräte einen bestimmten Konfigurationsstatus haben.	Gruppe, Konfigurationsvorlage																							
Firmwarestatus	Zeigt eine Übersicht wieviele Geräte welchen Firmware-Status haben.	Gruppe, Konfigurationsvorlage																							
„Top-X“ Geräteliste	Zeigt eine Liste mit Geräten anhand einer Spalte, z.B. Uptime oder Temperatur	Gruppe, Datenspalte, Anzahl Elemente	 <table><thead><tr><th>Hostname</th><th>Uptime</th></tr></thead><tbody><tr><td>NER-NUROVPN03</td><td>3 Jahre, 5 Monate, 18 Tage, 23 Stunden, 31 Minuten</td></tr><tr><td>NER-NUROVPN04</td><td>3 Jahre, 5 Monate, 18 Tage, 23 Stunden, 25 Minuten</td></tr><tr><td>NER-NUROVPN01</td><td>3 Jahre, 4 Monate, 21 Tage, 1 Stunde, 41 Minuten</td></tr><tr><td>STC-INGOVPN03</td><td>1 Jahr, 11 Monate, 7 Tage, 20 Stunden, 34 Minuten</td></tr><tr><td>ner-stcopenvpn02</td><td>1 Jahr, 10 Monate, 26 Tage, 1 Stunde, 23 Minuten</td></tr><tr><td>STC-INGOVPN01</td><td>1 Jahr, 8 Monate, 9 Tage, 36 Minuten</td></tr><tr><td>NER-HSTSW006</td><td>11 Monate, 1 Tag, 22 Stunden, 54 Minuten</td></tr><tr><td>STC-INGOVPN05</td><td>7 Monate, 28 Tage, 4 Stunden, 5 Minuten</td></tr><tr><td>VWO-INGOVPNCL</td><td>7 Monate, 28 Tage, 4 Stunden, 5 Minuten</td></tr><tr><td>NER-HSTSW009</td><td>6 Monate, 15 Tage, 1 Stunde, 58 Minuten</td></tr></tbody></table>	Hostname	Uptime	NER-NUROVPN03	3 Jahre, 5 Monate, 18 Tage, 23 Stunden, 31 Minuten	NER-NUROVPN04	3 Jahre, 5 Monate, 18 Tage, 23 Stunden, 25 Minuten	NER-NUROVPN01	3 Jahre, 4 Monate, 21 Tage, 1 Stunde, 41 Minuten	STC-INGOVPN03	1 Jahr, 11 Monate, 7 Tage, 20 Stunden, 34 Minuten	ner-stcopenvpn02	1 Jahr, 10 Monate, 26 Tage, 1 Stunde, 23 Minuten	STC-INGOVPN01	1 Jahr, 8 Monate, 9 Tage, 36 Minuten	NER-HSTSW006	11 Monate, 1 Tag, 22 Stunden, 54 Minuten	STC-INGOVPN05	7 Monate, 28 Tage, 4 Stunden, 5 Minuten	VWO-INGOVPNCL	7 Monate, 28 Tage, 4 Stunden, 5 Minuten	NER-HSTSW009	6 Monate, 15 Tage, 1 Stunde, 58 Minuten
Hostname	Uptime																								
NER-NUROVPN03	3 Jahre, 5 Monate, 18 Tage, 23 Stunden, 31 Minuten																								
NER-NUROVPN04	3 Jahre, 5 Monate, 18 Tage, 23 Stunden, 25 Minuten																								
NER-NUROVPN01	3 Jahre, 4 Monate, 21 Tage, 1 Stunde, 41 Minuten																								
STC-INGOVPN03	1 Jahr, 11 Monate, 7 Tage, 20 Stunden, 34 Minuten																								
ner-stcopenvpn02	1 Jahr, 10 Monate, 26 Tage, 1 Stunde, 23 Minuten																								
STC-INGOVPN01	1 Jahr, 8 Monate, 9 Tage, 36 Minuten																								
NER-HSTSW006	11 Monate, 1 Tag, 22 Stunden, 54 Minuten																								
STC-INGOVPN05	7 Monate, 28 Tage, 4 Stunden, 5 Minuten																								
VWO-INGOVPNCL	7 Monate, 28 Tage, 4 Stunden, 5 Minuten																								
NER-HSTSW009	6 Monate, 15 Tage, 1 Stunde, 58 Minuten																								
Gesundheitszustand	Zeigt eine Übersicht wieviele Geräte einen bestimmten Gesundheitszustand haben	Gruppe, Konfigurationsvorlage																							

SIM-Datenverbrauch	Zeigt eine Tabelle mit SIM-Karten und deren Verbrauch an.	Gruppe, Datenspalte, Nur SIM-Karten mit Datenlimit anzeigen, Anzahl Elemente	
Aufgaben mit Problemen	Zeigt eine Tabelle von Aufgaben an, welche auf dem Status Warnung oder Fehler stehen.		
Aufgabenstatus	Zeigt eine Übersicht der Stati von Aufgaben an.		
Indexseite	Zeigt eine Tabelle mit einer definierten Anzahl von Objekten an (z.B. Geräte, SIM-Karten, Benutzer, usw.)	Typ, Anzahl Elemente	

Einstellungen

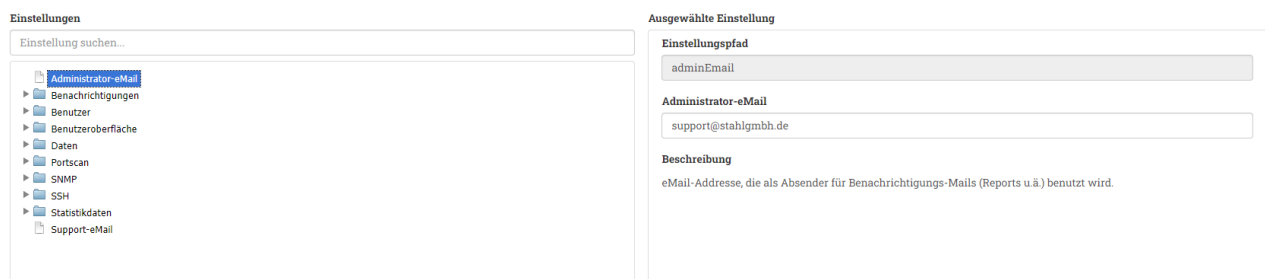
Über den Menüpunkt Verwaltung -> Einstellungen können Allgemeine Einstellungen konfiguriert werden.



Wichtig: Diese Parameter überschreiben evtl. gesetzte Einstellungen in einer params-local.php Datei.

Bedienung: im linken Bereich kann die gewünschte Einstellung gesucht und ausgewählt werden. Anschließend erscheint im rechten Fenster der Parameter und dessen Bedeutung.

Einstellungen



Backup

Ein Backup der Datenbank und des Systems kann mit dem Skript backupDB.sh erstellt werden. Dieses Skript sichert die Datenbank und die Dateien im Dateisystem inkl. aller hochgeladenen Dateien in den Ordner /backups/ auf dem lokalen System.

Bei einem manuellen Backup oder einer Übertragung auf ein anderes System ist darauf zu achten, dass der sog. „secret key“, welcher für die Ver- und Entschlüsselung der Passwörter zuständig ist, mit gesichert oder übertragen wird. Dieser befindet sich in der Datei „common/config/params-local.php“.

Bei der Standardinstallation wird das Skript backupDB.sh per crontab täglich aufgerufen.

Wiederherstellung von Backups (Restore)

Eine Wiederherstellung von Backups kann über das Skript restoreBackup.sh erfolgen.

Das Skript kann folgendermaßen verwendet werden:

```
Usage: /usr/sbin/restoreBackup.sh -h <hostname> [-d <date>] [-s
<source_directory>] [-u <ssh_username>] [-b] [-f]

-h <hostname>          Hostname des Quellsystems
-d <date>              Optional: Datum des Backups im Format YYYY-MM-DD
-s <source_directory>  Optional: Verzeichnis, in dem die Backups lokal
liegen
-u <ssh_username>      Optional: SSH-Benutzername für die Verbindung
-b                    Optional: Nur die Datenbank wiederherstellen
-f                    Optional: Zusätzlich Dateien wie Images und
Dokumente wiederherstellen
```

Also eine Wiederherstellung des letzten Backupstandes vom lokalen System erfolgt somit folgendermaßen:

```
restoreBackup.sh -h localhost
```

Globale Suchfunktion (optional)

Damit die globale Suche über einen externen Suchindex funktioniert, muss ein OpenSearch oder Elasticsearch installiert und entsprechend konfiguriert sein. (siehe Installationshandbuch)

Über eine geplante Aufgabe können bestimmte Datenobjekte einmalig indiziert werden. Zusätzlich werden die Objekte nach dem Anlegen oder Speichern entsprechend im Hintergrund indiziert.

Sofern die Suche konfiguriert ist und der Benutzer die Berechtigung „SearchQuery“ hat, wird in der Menuleiste ein entsprechendes Suchfeld eingeblendet:



Dort kann man Abfragen gegenüber den indizierten Objekten starten und bekommt diese als Ergebnisse in einer Tabelle aufgelistet. Die Reihenfolge wird über eine Relevanz bestimmt welche die externe Suchmaschine übermittelt.

Wichtig: Es werden pro Datenobjekttyp maximal 10 Suchergebnisse angezeigt. (werden unter dem eingeben Suchbegriff also 20 Geräte und 30 Vorlagen gefunden, erscheinen 10 Geräte und 10 Vorlagen im Ergebnis)

Referenz:

Die Suche über das Suchfeld findet über eine „`simple_query_string`“ Abfrage statt, Referenz: <https://opensearch.org/docs/latest/query-dsl/full-text/simple-query-string/>

Folgende Operatoren werden unterstützt:

Operator	Beschreibung
<code>+</code>	Wirkt wie der <code>AND</code> Operator.

Operator Beschreibung

	Wirkt wie der OR Operator.
*	Sofern am Ende einer Abfrage genutzt symbolisiert der Operator eine Präfixabfrage.
"	Umschließt mehrere Begriffe zu einer Phrase (z.B. "wind rises").
(,)	Umschließt eine Klausel zur Priorisierung (z.B. wind + (rises rising)).
~n	Wenn nach einem Begriff verwendet (zum Beispiel, wind~3), legt dies eine Unschärfe (fuzziness) fest. Wenn nach einer Phrase verwendet, legt dies eine Toleranz fest.
-	Negiert den Ausdruck.

Alle vorherigen Operatoren sind reservierte Zeichen. Um sie als Rohzeichen zu referenzieren und nicht als Operatoren, muss jedes von ihnen mit einem Backslash escapet werden.

Benutzerdefinierte CSS-Datei zur Anpassung des NMS-Designs

Das NMS unterstützt die Möglichkeit, eine benutzerdefinierte CSS-Datei zu verwenden, um das Design der Benutzeroberfläche individuell anzupassen. Diese Funktion bietet Administratoren die Flexibilität, das Aussehen des Systems an unternehmensspezifische Anforderungen oder persönliche Vorlieben anzupassen.

Die benutzerdefinierte CSS-Datei muss unter folgendem Pfad abgelegt werden:

„custom/css/custom.css“

Auswertungen

Das Auswertungsmodul stellt vordefinierte Auswertungen („Reports“) bereit. Eine Auswertung ist eine gespeicherte Datenbankabfrage mit optionalen Variablen. Die Auswertungen werden von Stahl erstellt und als signierte Datei ausgeliefert; sie werden im NMS importiert, als Kacheln angezeigt, als PDF oder CSV exportiert und können per Zeitplan automatisch per E-Mail versendet werden.

Übersicht

Der Menüpunkt „Auswertungen“ zeigt alle aktiven Auswertungen als Kacheln. Jede Kachel zeigt den Namen, eine kurze Beschreibung sowie eine Vorschau der Ergebnistabelle.

Oben rechts im übergeordneten Panel befinden sich zwei Symbole:

- Aktualisieren (alle): erzeugt die Ergebnisse aller Auswertungen neu. Dies läuft im Hintergrund nacheinander ab, sodass die Oberfläche bedienbar bleibt.
- Importieren: öffnet den Dialog zum Hochladen einer Auswertungs-Datei (.rep).

Jede einzelne Kachel besitzt eigene Symbole:

- Aktualisieren: berechnet das Ergebnis dieser Auswertung neu.
- Anzeigen: öffnet die Einzelansicht mit der vollständigen Tabelle.
- Löschen: entfernt die Auswertung (nur mit entsprechender Berechtigung).

Auswertung anzeigen

Ein Klick auf das Anzeigen-Symbol öffnet die Einzelansicht. Dort wird das vollständige Ergebnis als Tabelle dargestellt. Die Tabelle unterstützt Sortierung, Filterung und das Ein-/Ausblenden von Spalten (Personalisierung wie bei den übrigen NMS-Tabellen).

Oberhalb der Tabelle werden links der Zeitpunkt der letzten Aktualisierung sowie die Anzahl der Datensätze angezeigt. Rechts befinden sich die Schaltflächen „Parameter“ und „Export“.

Parameter (Variablen)

Hat eine Auswertung Variablen (z. B. ein Suchmuster oder ein Mindestwert), erscheint die Schaltfläche „Parameter“. Ein Klick öffnet ein Fenster mit den Eingabefeldern. Nach „Anzeigen“ wird die Auswertung mit den gewählten Werten neu ausgeführt.

Export

Über die Schaltfläche „Export“ kann das aktuell angezeigte Ergebnis als PDF oder CSV heruntergeladen werden. Die CSV-Datei nutzt Semikolon als Trennzeichen und ist für das Öffnen in Excel (deutsche Einstellungen) vorbereitet.

Importieren

Neue Auswertungen werden als Datei mit der Endung .rep ausgeliefert. Über das Import-Symbol wird die Datei hochgeladen. Das NMS prüft die Signatur der Datei; veränderte oder nicht signierte Dateien werden abgelehnt. Ein erneuter Import derselben Auswertung aktualisiert die vorhandene (kein Duplikat).

Automatische Aktualisierung

Unter Einstellungen → Auswertungen lässt sich das „Aktualisierungsintervall“ (in Minuten) einstellen. Ist ein Wert größer 0 gesetzt, werden die Kacheln der Übersicht automatisch im Hintergrund aktualisiert und veraltete Ergebnisse neu berechnet. Beim Wert 0 erfolgt die Aktualisierung nur manuell über die Aktualisieren-Symbole.

Versand per Zeitplan

Auswertungen können über die Aufgabenverwaltung automatisch per E-Mail versendet werden. Dazu wird ein Zeitplan mit dem Benachrichtigungstyp „Auswertung versenden“ angelegt. Dort werden eine oder mehrere Auswertungen, das Format (PDF oder CSV) sowie die Empfänger-Adressen ausgewählt. Zum eingestellten Zeitpunkt werden die Auswertungen erzeugt und als Anhang versendet.

Inventarisierung

Geräte

Geräte können über den Menüpunkt „Geräte“ -> „Geräte“ aufgerufen werden.

Unter einem Gerät wird im NMS die objektorientierte Abbildung eines physikalischen Netzwerkgertes verstanden. Es dient der Steuerung allgemeiner Geräteparameter im NMS (Hostname, Management-IP, Serial, ...) und erlaubt mittels administrativem Zugriff das Ausführen von Systemfunktionen auf dem physikalischen Gerät (Reboot, FW-Update, Konfigurations-Update, ...).

Ein Gerät ist dabei zur Wahrung der Mandantenfähigkeit immer genau einer Gruppe zugeordnet.

Wartungsmodus

Ein Gerät kann für einen bestimmten Zeitraum oder auf unbegrenzte Zeit in den Wartungsmodus gesetzt werden. Im Wartungsmodus wird die Statusabfrage angehalten und das System stellt beim Abruf keine Firmwaredateien und Konfigurationen mehr zur Verfügung. Die Anfrage nach einer Konfiguration oder Firmware wird mit dem HTTP-Fehlercode 503 beantwortet.

Eingeschaltet werden kann der Wartungsmodus beim Gerät auf dem Tab „Status“:

Details	Status	Module	Interfaces	Subnetze	IP-Adressen	SIM-Karten	Passwörter	Firmware	Konfigurationen
Status									
Wartungsmodus		Einschalten							
Uptime		4 Stunden, 39 Minuten							
Dienste		SSH, HTTP, HTTPS							

Hier kann eine zeitliche Beschränkung des Wartungsmodus stattfinden:

Wartungsmodus einschalten

Zeitliche Beschränkung

☐ Unbeschränkt
 ☒ Zeitgesteuert

Wartungsbeginn

Sofort

Wartungsende

Unbeschränkt

Einschalten

Abbrechen

Alternativ kann der Wartungsmodus auch über die Massенbearbeitung an- oder ausgeschaltet werden. Eine zeitliche Begrenzung ist über die Massенbearbeitung nicht möglich.

Import

Ein Import von Geräten ist über den Menüpunkt „Importieren“ erreichbar. Die CSV Datei sollte kommasetrennt sein (Excel verwendet Semikolon). Außerdem sollten die IP-Adresse und die Seriennummer eindeutig sein. Die Angabe von Kunde, Busnummer, Seriennummer, IP, Passwort und SNMP-Zugangsdaten sind optional.

Export

Ein Export aller Geräte als CSV-Datei ist über den Button „Exportieren“ möglich.

Unzugewiesene Geräte

Unter dem Menüpunkt „Unzugewiesene Geräte“ werden Geräte aufgelistet, die Konfigurationen, Firmware, etc. von der Download-Applikation angefragt haben, aber keinem angelegten Gerät zugeordnet werden können. Die aufgenommenen Informationen dieser Geräte enthalten die IP-Adresse, das Datum der ersten und letzten Verbindung und, wenn verfügbar, den Hostnamen und die Seriennummer. Aus diesen Einträgen können Geräte mit den bekannten Informationen erstellt werden.

Einbauobjekte

Unter dem Menüpunkt Einbauobjekte verbirgt sich die Inventarisierung von Einbauobjekten. Einbauobjekte können im Anschluss Geräte zugeordnet werden.

Ein Einbauobjekt besteht aus einem eindeutigen Namen(Nummer), dem Typ (Bus, Bahn, usw.) und dem Kunden.

Einem Objekt können mehrere Geräte zugeordnet werden.

Die Objekttypen können mit der Rolle „admin“ unter dem Menüpunkt „Verwaltung“ -> „Einbauobjekttypen“ verwaltet werden.

Gerätevorlagen

Gerätevorlagen ermöglichen es, alle Geräte für ein Fahrzeug automatisch anhand von Konfigurationsvorlagen zu erstellen.

Zuerst muss eine Gerätevorlagen Sammlung erstellt werden, welche mehrere Gerätevorlagen pro Gruppe definiert.

Gerätevorlagen werden in einer Gerätevorlagen Sammlung definiert.

Eine Gerätevorlage besteht aus folgenden Angaben:

- Name – zur Identifizierung der Vorlage
- Einbauobjekttyp – der Typ, für den die Geräte erstellt werden
- Konfigurationsvorlage – die Vorlage für das zu erzeugende Gerät
- Parameter – optionale Feldzuweisungen aus anderen Geräten

Parameter können verwendet werden, um Werte eines erzeugten Geräts mit Werten eines anderen Geräts zu ersetzen, das ebenfalls über eine Gerätevorlage erstellt wurde.

Es muss mindestens eine Gerätevorlage ohne Parameter existieren.

Beispiel für Parametereinstellungen:

```
{
  "parent_id": {                                <- Name des Gerätefeldes, das geändert werden soll
    "device_template_id": 1, <- ID der Gerätevorlage, die das Quellgerät erzeugt
    "format": "{%id%}"        <- Neuer Wert für das neue Gerät. Kann Platzhalter für Werte
                                aus dem Quellgerät verwenden
  },
  "ip": {
    "device_template_id": 1,
    "format ": "{%ip%}"
  }
}
```

In diesem Beispiel werden beim erzeugten Gerät folgende Werte übernommen:

- parent_id erhält die id des Geräts, das durch die Gerätevorlage mit der ID 1 erstellt wurde.
- ip erhält die ip-Adresse dieses Geräts.

Im Einbauobjekt können die Geräte auf Basis der definierten Gerätevorlagen erzeugt werden.

Existiert im Einbauobjekt bereits ein Gerät mit derselben Konfigurationsvorlage wie in einer Gerätevorlage, wird kein neues Gerät für diese Gerätevorlage angelegt.

Jede Gerätevorlage kann pro Einbauobjekt nur ein Gerät erzeugen.

SIM-Karten

Die SIM-Karten werden normalerweise automatisch aus den verwalteten Geräten ausgelesen, können aber auch aus einer CSV-Datei importiert werden.

Import

Ein Import von SIM-Karten ist über den Menüpunkt „Importieren“ erreichbar. Die CSV Datei sollte kommagetrennt sein (Excel verwendet Semikolon).

Außerdem sollten die Kartennummer und die Telefonnummer eindeutig sein.

Pflichtfelder sind: Kartennummer und Active

Die Angabe von Kunde, DeviceHostname, DeviceSerial, Busnummer, PhoneNumber, Provider, PIN und PUK sind optional.

Export

Ein Export aller SIM-Karten als CSV-Datei ist über den Button „exportieren“ möglich.

Geräteverwaltung

Das NMS unterstützt die Verwaltung von Geräten verschiedener Hersteller und Modellreihen mit diversen Konfigurationsformaten und -protokollen. Realisiert wurde dies mithilfe einer Gerätetreiber- und Protokollschicht, welche die Kommunikation mit dem verwalteten Gerät übernimmt und auf diesen die benötigten Funktionen ausführt (z.B. Konfigurations- und Firmware-Update, Statusabfragen, usw.).

OpenVPN Konfigurations

1. OpenVPN-Zertifikatsmanagement

Für die Erstellung von Zertifikaten wird ein PKI-Server benötigt (Gerätetreiber: Stahl Zertifizierungsdienste (EasyRSA v2/v3)).

Dort muss im Tab „Instanzen“ eine neue PKI-Instanz angelegt werden, die alle Zertifikatsinformationen enthält.

Optional kann hier ein bestehendes CA-Zertifikat angegeben werden, sofern bereits ein passendes vorhanden ist. Das Feld kann jedoch auch leer gelassen werden. In diesem Fall wird beim erstmaligen Erstellen eines neuen Zertifikats automatisch ein CA-Zertifikat mitgeneriert.

Im Tab „OpenVPN-Instanzen“ können OpenVPN-Instanzen mit der jeweiligen PKI-Instanz verknüpft werden.

2. OpenVPN-Clientkonfiguration

Für die Verbindung von Geräten mit einem OpenVPN-Server muss zunächst ein OpenVPN-Server als Gerät angelegt werden (Gerätetreiber: Stahl OpenVPN-Dienstanbieter).

Zusätzlich müssen OpenVPN-Instanzen als Geräte angelegt werden (Gerätetreiber: Stahl OpenVPN-Instanz).

OpenVPN-Server und OpenVPN-Instanzen können jeweils in den Tabs „Instanzen“ bzw. „OpenVPN-Server“ in einer m:n-Beziehung miteinander verknüpft werden.

Darüber hinaus müssen OpenVPN-Instanzen mindestens einer PKI-Instanz zugeordnet sein.

3. Verbindung von Geräten mit einer OpenVPN-Instanz

Für die Verbindung eines Geräts mit einer OpenVPN-Instanz muss in der entsprechenden Vorlage die folgende Konfigurationszeile hinzugefügt werden:

\$openvpn.tunnel.[X]=[ID]

- **[X]**: Der Index des VPN-Tunnels.
- **[ID]**: Die Geräte-ID der OpenVPN-Instanz, die zugeordnet werden soll.

SSH TerminalServer Funktion

Über den integrierten SSH TerminalServer kann bequem über den Browser per SSH auf ein Gerät zugegriffen werden. Dafür muss dem Benutzer eine Rolle mit der Berechtigung „DeviceTerminal“ zugewiesen sein. Als weitere Voraussetzung gilt, dass auf dem Server der TerminalServer konfiguriert und gestartet wurde.

Sind alle Anforderungen erfüllt, erscheint bei einem Gerät der Button „Terminalsitzung starten“.

Nach Auswahl der entsprechenden Zugangsdaten für die Verbindung öffnet sich ein Terminal Fenster und die Verbindung zum Gerät wird über den Server als Jump-Host aufgebaut.

Hinweis: Copy & Paste im Terminalsitzungsfenster ist ausschließlich über die Maus möglich, da sämtliche Tastatureingaben zum Gerät weitergeleitet werden.

Unterstützte Verwaltungsprotokolle

Derzeit stehen die folgenden Protokolle für Kommunikation der Gerätetreiber mit dem Endgerät zur Verfügung:

- HTTP/S (Hypertext Transfer Protocol / mit oder ohne Verschlüsselung)
- SSH (Secure Shell)
- SNMP (Simple Network Management Protocol / Version 1, 2 und 3)

Welches Protokoll letztendlich zur Kommunikation benutzt werden soll, lässt sich über die Benutzeroberfläche mittels Auswahlfeld und per REST-API individuell pro Gerätedatensatz festlegen. Zu beachten ist, dass der vom Gerätetreiber unterstützte Funktionsumfang je nach verwendetem Kommunikationsprotokoll variieren kann.

Konfiguration von Protokolloptionen

Einige der verfügbaren Kommunikationsprotokolle ermöglichen eine gezielte Beeinflussung der Datenübertragung über sog. Protokolloptionen. Die Konfiguration dieser Optionen erfolgt analog zu den im nachfolgenden Kapitel beschriebenen Gerätetreiberoptionen in Form einer „Fallback-Kette“ (= spezifischste Definition gewinnt, sonst Nutzungsversuch einer allgemeineren Definition) und kann an den folgenden Stellen im System definiert werden:

- In der globalen Anwendungskonfiguration („*common/config/params-local.php*“) durch Definition des Schlüssels „DeviceProtocol“,
- über die Eigenschaft „DeviceProtocol“ in den Metadaten des jeweiligen Gerätemodell-Datensatzes, und/oder
- in den Metadaten eines Gerätes durch Definition der Eigenschaft „DeviceProtocol“.
- Jede dieser Einstellungen erfordert die Angabe einer Untereigenschaft, die den technischen Namen des Kommunikationsprotokolls trägt (z.B. „Http“, „Snmp“, „Ssh“). Unterhalb dieser können dann die jeweils für das Protokoll zu verwendenden Optionen hinterlegt werden.

Achtung: Manche Gerätetreiber ignorieren absichtlich die für das Kommunikationsprotokoll definierten Optionen. Dies geschieht insbesondere dann, wenn veränderte Optionen einen Fehlschlag des Verbindungsaufbaus oder ähnliche Funktions-/Kommunikationsstörungen zur Folge hätten.

Tipp: Protokolloptionen, die nur für bestimmte gleichartige Geräte gedacht sind, lassen sich für die einzelnen Geräte auch über Konfigurationsvorlagen bzw. -schnipsel (siehe Folgekapitel) zuweisen. Dazu kann – hier am Beispiel einer Zeile aus einer NetModule-Konfigurationsvorlage – der Umstand genutzt werden, dass sich Metadaten eines Geräts in den Vorlagen und Schnipseln definieren lassen, z.B. mittels:

```
$dms.meta.DeviceProtocol.<Techn_Protokollname>.<Optionsname>.<Unteroption>=<Optionswert>
```

Die in der Dokumentation nachfolgend verwendete Schreibweise von Optionsbezeichnungen mit einer Punkttrennung (z.B. „Security.VerifyHost“) ist als hierarchische Folge von Unteroptionen zu verstehen, die unterhalb der Startebene („DeviceProtocol.<Technischer Protokollname>“) zu definieren ist. Bei der Angabe von technischen Protokollnamen sowie Protokolloptionsnamen ist auf korrekte Groß- und Kleinschreibung zu achten.

Liste der Protokolloptionen

HTTP

Technischer Protokollname: Http

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Redirect.Follow	Wahrheitswert	Folgt Umleitungen durch „Location:“-Header (Standard: aktiv).
Port	Ganzzahl	Ermöglicht den Verbindungsaufbau der HTTP(S)-Sitzung über einen alternativen TCP-Port (Wertebereich: 1-65535, Standards: 80 für unverschlüsseltes HTTP, 443 für verschlüsseltes HTTPS).
Security.Force	Wahrheitswert	Erzwingt den Verbindungsaufbau über das HTTPS-Protokoll auch dann, wenn der Gerätetreiber eine unverschlüsselte Verbindung zum Gerät herzustellen versucht (Standard: aktiv).
Security.VerifyPeer	Wahrheitswert	Legt fest, ob das SSL-/TLS-Zertifikat des angesprochenen Geräts beim Verbindungsaufbau validiert werden soll (Standard: aktiv). Bei Fehlschlag der Validierung wird der Verbindungsaufbau abgebrochen.
Security.VerifyHost	Wahrheitswert	Legt fest, ob beim Verbindungsaufbau die Übereinstimmung eines Common Name-Feldes oder eines Subject Alternate Name-Feldes im SSL-/TLS-Peer-Zertifikat mit dem angegebenen Hostnamen geprüft werden soll (Standard: aktiv). Bei Fehlschlag der Prüfung wird der Verbindungsaufbau abgebrochen.
Timeout	Ganzzahl	Gibt die Zeitdauer (in Sekunden) an, die bis zum Abschluss einer abgesetzten Anfrage verstreichen darf (Standard: 10 Sekunden).

SNMP

Technischer Protokollname: Snmp

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Timeout	Ganzzahl	Gibt die Zeitdauer (in Sekunden) an, die bis zum erfolgreichen Aufbau einer Verbindung verstreichen darf (Standard: 1 Sekunde).

SSH

Technischer Protokollname: Ssh

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Port	Ganzzahl	Ermöglicht den Verbindungsaufbau der SSH-Sitzung über einen alternativen TCP-Port (Wertebereich: 1-65535, Standard: 22).
Timeout	Ganzzahl	Gibt die Zeitdauer (in Sekunden) an, die bis zum erfolgreichen Aufbau einer Verbindung verstreichen darf (Standard: 10 Sekunden).

ICMP

Technischer Protokollname: Icmp

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
TTL	Ganzzahl	Der TTL-Wert (Time to Live) in einem Ping-Befehl gibt die maximale Anzahl an Routern oder Hops an, die ein Paket durchlaufen darf, bevor es verworfen wird. Er verhindert, dass Pakete endlos im Netzwerk zirkulieren, indem er bei jedem Hop um eins verringert wird. (Standard: 128)
Timeout	Ganzzahl	Gibt die Zeitdauer (in Sekunden) an bis eine Ping-Antwort zurückkommen muss (Standard: 3 Sekunden).

Gerätetreiber

Allgemeine Hinweise zur Treibernutzung

Über den Bereich „Verwaltung“ → „Gerätemodelle“ können die vom NMS zu verwaltenden Gerätemodelle angelegt werden. Bei der Anlage eines neuen Gerätemodells muss jeweils ein dafür geeigneter Treiber (auch „Kommando-Prozessor“ genannt) ausgewählt werden, der mit den zu verwaltenden Geräten kompatibel sein muss und den jeweilig nutzbaren Funktionsumfang bestimmt.

Konfiguration von Treiberoptionen

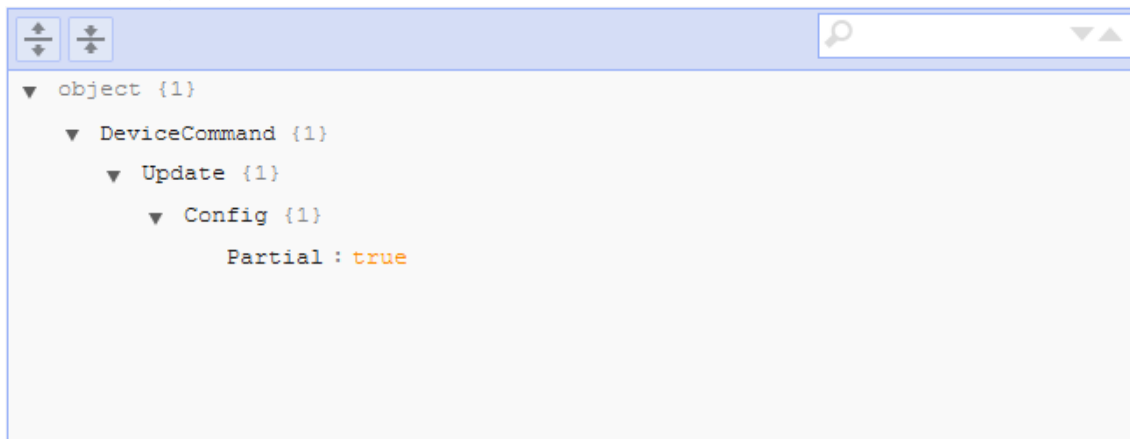
Manche Treiber verfügen über eigene Konfigurationsmöglichkeiten („Treiber-Optionen“) und bieten hierüber die Möglichkeit einer anwendungsglobalen, modellabhängigen und sogar gerätespezifischen Feinjustierung der vom Treiber bereitgestellten Funktionen.

Treiberoptionen können in Form einer „Fallback-Kette“ (= spezifischste Definition gewinnt, sonst Nutzungsversuch einer allgemeineren Definition) an den folgenden Stellen im System definiert werden:

- In der globalen Anwendungskonfiguration („*common/config/params-local.php*“) durch Definition des Schlüssels „DeviceCommand“:

```
'DeviceCommand' => [
    '<Technischer Treibername>' => [
        '<Optionsname1>' => '<Optionswert1>',
        '<Optionsname2>' => [
            'Unteroption1' => [
                '<Optionsname3>' => '<Optionswert3>',
                'Unter-Unteroption1' => [
                    '<Optionsname4>' => '<Optionswert4>',
                    '<Optionsname5>' => '<Optionswert5>',
                ],
            ],
        ],
    ],
],
```

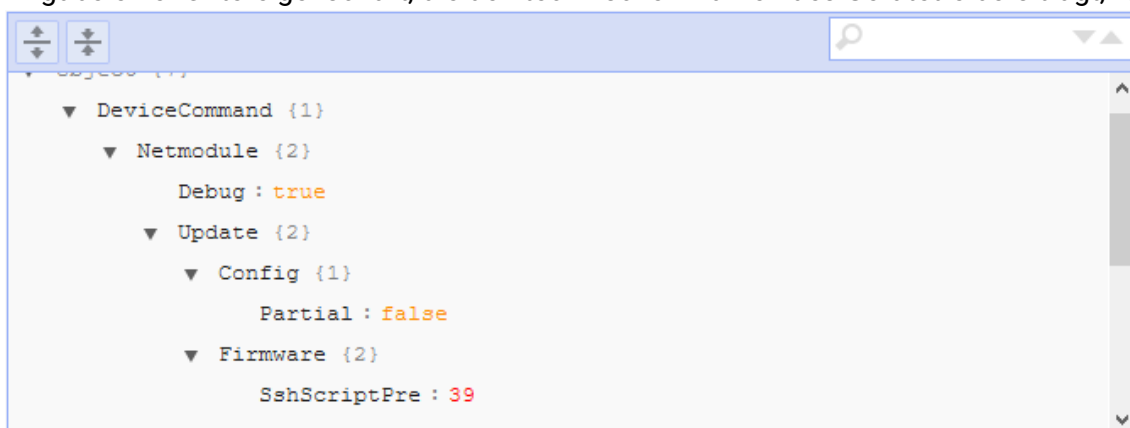
- Über die Eigenschaft „DeviceCommand“ in den Metadaten des jeweiligen Gerätemodell-Datensatzes, z.B.:



- In den Metadaten einer Gruppe durch Definition der Eigenschaft „DeviceCommand“ mit Angabe einer Untereigenschaft, die den technischen Namen des Gerätetreibers trägt, z.B.:



- In den Metadaten eines Gerätes durch Definition der Eigenschaft „DeviceCommand“ mit Angabe einer Untereigenschaft, die den technischen Namen des Gerätetreibers trägt, z.B.:



Tipp: Gerätetreiberoptionen, die nur für bestimmte gleichartige Geräte gedacht sind, lassen sich für die einzelnen Geräte auch über Konfigurationsvorlagen bzw. -schnipsel (siehe Folgekapitel) zuweisen. Dazu kann – hier am Beispiel einer Zeile aus einer NetModule-Konfigurationsvorlage – der Umstand genutzt werden, dass sich Metadaten eines Geräts in den Vorlagen und Schnipseln definieren lassen, z.B. mittels:

```
$dms.meta.DeviceCommand.<Techn_Treibername>.<Optionsname>.<Unteroption>=<Optionswert>
```

Die in der Dokumentation nachfolgend verwendete Schreibweise von Optionsbezeichnungen mit einer Punkttrennung (z.B. „Update.Config.Partial“) ist als hierarchische Folge von Unteroptionen zu verstehen, die unterhalb der Startebene („DeviceCommand.<Technischer Treibername>“ bzw. „Options“) zu definieren ist.

Bei der Angabe von technischen Treibernamen sowie Treiberoptionsnamen ist auf korrekte Groß- und Kleinschreibung zu achten.

Verfügbare Gerätetreiber

APS-R-PoE

Gerätetreiber zur Steuerung von Fahrgastzählern des Herstellers Hella Aglaia.

Technischer Treibername

APS-R-PoE

Unterstützte Kommunikationsprotokolle

- HTTP/S

Funktionsumfang

- Gerätereustart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Seriennummer, Temperatur, Netzwerkkonnektivität, Firmware-Zustände)
- Initiieren von Firmware-Updates, die das Endgerät vom NMS beziehen kann
- Abfrage von Fahrgast Daten

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

CD9-X

Gerätetreiber zur Steuerung von IP-Kameras des Herstellers r2p.

Technischer Treibername

CD9-X

Unterstützte Kommunikationsprotokolle

- HTTP/S

Funktionsumfang

- Gerätereustart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Seriennummer, Netzwerkkonnektivität, Firmware-Zustände)
- Abfrage des aktuellen Kamerabildes

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

Cisco (IE-3x00 Series)

Gerätetreiber zur Steuerung von Switches der IE-Serie des Herstellers Cisco.

Technischer Treibername

CiscoSwitch

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Geräteneustart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit, Netzwerkkonnektivität, Konfigurations- und Firmware-Zustände)
- Initiieren von Konfigurationsupdates, die das Endgerät vom NMS beziehen kann
- Initiieren von Firmware-Updates, die das Endgerät vom NMS beziehen kann

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
Neighbors.Interface.Whitelist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche an die aufgelisteten Schnittstellen angeschlossen sind.
Neighbors.Interface.Blacklist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche NICHT an die aufgelisteten Schnittstellen angeschlossen sind.

Generisches Linux-OS

Gerätetreiber zur Steuerung von Geräten, Servern und Client-Computern, die mit einem Linux-basierten Betriebssystem arbeiten.

Technischer Treibername

GenericLinux

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Geräteneustart
- Abfrage grundlegender Gerätestatusdaten (z.B. Distribution, Version, Verfügbarkeit, Systemgesundheit, Netzwerkkonnektivität, lokale Systembenutzer)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
PingBeforeStatusQuery	Wahrheitswert	Vor der eigentlichen Statusabfrage wird das Gerät mittels Ping auf Verfügbarkeit geprüft. Ist es per Ping nicht erreichbar findet keine weitere Abfrage statt.

Generischer Ping

Einfacher Gerätetreiber zur Prüfung der Verfügbarkeit eines Gerätes mittels PING ohne Zugangsdaten.

Technischer Treibername

Ping

Unterstützte Kommunikationsprotokolle

- ICMP

Funktionsumfang

- Abfrage der Verfügbarkeit mittels PING

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

Hirschmann HIOS

Gerätetreiber zur Steuerung von Switches mit der HiOS Software des Herstellers Hirschmann.

Technischer Treibername

HirschmannHiOs

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Gerätereustart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Seriennummer, Konfigurations- und Firmware-Zustände)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

Hirschmann OCTOPUS

Gerätetreiber zur Steuerung von Switches mit der OCTUPUS Software des Herstellers Hirschmann.

Technischer Treibername

HirschmannOctopus

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Gerätereustart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Seriennummer, Konfigurations- und Firmware-Zustände)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

LANCOM (LCOS SX v4)

Gerätetreiber zur Steuerung von Switches der LCOS SX-Serie des Herstellers Lancom.

Technischer Treibername

LancomLcosSx4

Unterstützte Kommunikationsprotokolle

- HTTP/S

Funktionsumfang

- Geräteneustart
- Initiieren von Konfigurationsupdates, die das Endgerät vom NMS beziehen kann
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit, Netzwerkkonnektivität, Konfigurations- und Firmware-Zustände)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

Moxa (EDS Series)

Gerätetreiber zur Steuerung von Switches der EDS-Serie des Herstellers Moxa.

Wichtig: Damit der Treiber funktioniert muss die Option „login mode cli“ aktiviert werden.

Technischer Treibername

MoxaEdsSwitch

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Gerätereuestart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Firmware-Zustände)
- Abfrage der auf dem Gerät gespeicherten Konfiguration
- Abfrage von verbundenen Geräten
- Vergleich der Uhrzeit

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
Neighbors.Interface.Whitelist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche an die aufgelisteten Schnittstellen angeschlossen sind.
Neighbors.Interface.Blacklist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche NICHT an die aufgelisteten Schnittstellen angeschlossen sind.

Netmodule (alle Modelle)

Generischer Treiber zur Steuerung aller Router der NB-Serie des Herstellers NetModule.

Technischer Treibername

Netmodule

Unterstützte Kommunikationsprotokolle

- SSH
- SNMP v1/v2/v3

Funktionsumfang

- Gerätereustart
- Initiieren von Konfigurationsupdates, die das Endgerät vom NMS beziehen kann
- Initiieren von Firmware-Updates, die das Endgerät vom NMS beziehen kann
- Beeinflussung des Standardwertverhaltens beim Konfigurationsupdate
- Ausführen von im NMS gelagerten Shell-Skripten vor und nach den o.g. Update-Prozessen, um mittels Bedingungen und zusätzlichen Aktionen die Updateprozesse bedarfsgerecht steuern und anpassen zu können (nur via SSH-Protokoll)
- Download von Support-/Debug-Informationen, die das Gerät bereitstellt (nur via SSH-Protokoll)
- Abfrage von detaillierten Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit, Systemkomponenten, Konfigurations- und Firmware-Zustände, Netzwerkkonnektivität, lokale Systembenutzer, Standortdaten, Volumina des Datenverkehrs und -verbrauchs)
- Abfrage von verbundenen Geräten

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
PingBeforeStatusQuery	Wahrheitswert	Vor der eigentlichen Statusabfrage wird das Gerät mittels Ping auf Verfügbarkeit geprüft. Ist es per Ping nicht erreichbar findet keine weitere Abfrage statt.
Update.Config.Upload	Wahrheitswert	Die Konfiguration wird vor dem einspielen per SCP auf das Gerät geladen anstatt per http(s) heruntergeladen.
Update.Config.Partial	Wahrheitswert	Wenn aktiviert (Standard), werden in der vom NMS ausgelieferten Gerätekonfiguration die nicht definierten Werte bei der Übernahme in das Gerät ignoriert; andernfalls werden die undefinierten Werte mit Werksvorgaben des Herstellers belegt.

Update.Config.SshScriptPre <i>(nur möglich mit SSH als Kommunikationsprotokoll)</i>	Ganzzahl	NMS-ID eines (Shell-)Skriptes, das zur Ausführung vor einem Konfigurationsupdate vorgesehen ist. Das Skript kann mittels Exit-Code (0 = Erfolg; sonst Fehlschlag) signalisieren, ob der Updatevorgang fortgesetzt werden soll oder nicht.
Update.Config.SshScriptPost <i>(nur möglich mit SSH als Kommunikationsprotokoll)</i>	Ganzzahl	NMS-ID eines (Shell-)Skriptes, das zur Ausführung nach einem erfolgreichen Konfigurationsupdate vorgesehen ist.
Update.Config.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für ein Konfigurationsupdate auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird i.d.R. nur bei Verwendung von Pre-/Post-Update-Skripten benötigt, da insbesondere für die Ausführung des Post-Skriptes der Abschluss des Updates abgewartet werden muss. Auf Geräten mit schwächerer CPU kann dieser Prozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Update.Firmware.Upload	Wahrheitswert	Das Softwareimage wird vor dem einspielen per SCP auf das Gerät geladen anstatt per http(s) heruntergeladen.
Update.Firmware.SshScriptPre <i>(nur möglich mit SSH als Kommunikationsprotokoll)</i>	Ganzzahl	NMS-ID eines (Shell-)Skriptes, das zur Ausführung vor einem Firmware-Update vorgesehen ist. Das Skript kann mittels Exit-Code (0 = Erfolg; sonst Fehlschlag) signalisieren, ob der Updatevorgang fortgesetzt werden soll oder nicht.
Update.Firmware.SshScriptPost <i>(nur möglich mit SSH als Kommunikationsprotokoll)</i>	Ganzzahl	NMS-ID eines (Shell-)Skriptes, das zur Ausführung nach einem Firmware-Update vorgesehen ist. Der Exit-Code des Firmware-Updateprozesses (0 = Erfolg; sonst Fehlschlag) wird als erster Kommandozeilenparameter an das Skript übergeben, sodass in weitere Aktionen in Abhängigkeit von dessen Erfolg ausführbar sind. Das Skript selbst kann mittels Exit-Code (0 = Erfolg; sonst Fehlschlag) signalisieren, ob der im Anschluss an ein erfolgreiches Firmware-Update erforderliche Gerätereustart ausgeführt werden soll oder nicht. Im Falle eines fehlgeschlagenen Updates sollte kein Neustart erfolgen.

Update.Firmware.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für ein Firmware-Update auf die angegebene Anzahl von Sekunden (Standard: 600 = 10 Minuten). Ein höherer Timeout wird i.d.R. nur bei Verwendung von Pre-/Post-Update-Skripten benötigt, da insbesondere für die Ausführung des Post-Skriptes der Abschluss des Updates abgewartet werden muss. Auf Geräten mit schwächerer CPU kann dieser Prozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Status.Comparison.Jitter	Ganzzahl	Definiert eine Karenzzeit in Sekunden, die vom gerätelokalen Änderungszeitstempel der Konfiguration abgezogen werden soll (Standard: 30 Sekunden). Diese Einstellung erlaubt den Ausgleich des Zeitunterschieds beim Vergleich zwischen Zeitstempeln der Konfiguration vom Gerät und des Konfigurations-Downloads vom NMS. Sie besagt in etwa, wieviel Zeit das Gerät zwischen Download und Anwendung der Konfiguration benötigen darf.
Status.History.IgnoreOptions	Array von Zeichenketten	Ignoriert die angegebenen Konfigurationsoptionen bei der Entscheidung, ob die bei der Statusabfrage vom Gerät heruntergeladene Konfiguration als geändert betrachtet werden soll. Dadurch kann das Anlegen eines neuen Eintrags in der Konfigurationshistorie verhindert werden, um bei häufig wechselnden Optionen (z.B. „custom.var.*“) das exzessive Anlegen unnötiger Versionshistorieneinträge zu verhindern. Die zu ignorierenden Optionen müssen als Liste Perl-kompatibler regulärer Ausdrücke (PCRE) angegeben werden und ermöglichen dadurch auch das Ignorieren mehrerer gleichartiger Optionen (z.B. mit fortlaufender Nummerierung). Beispielsweise unterdrückt der Eintrag „custom\.var\d+“ die Berücksichtigung aller frei definierbaren Variablen in einer NetModule-Konfiguration. Die PCRE-Syntax entspricht dem Standard der PHP-Dokumentation, die unter

		https://www.php.net/manual/de/reference.pcre.pattern.syntax.php verfügbar ist.
Support.TechFile.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für den Download eines Support-TechFile-Paketes auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird ggf. benötigt, um langsamere Verbindungen auszugleichen, damit der Download sicher abgeschlossen werden kann. Auf Geräten mit schwächerer CPU kann der Erstellungsprozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Neighbors.Interface.Whitelist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche an die aufgelisteten Schnittstellen angeschlossen sind.
Neighbors.Interface.Blacklist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche NICHT an die aufgelisteten Schnittstellen angeschlossen sind.
Neighbors.IbisIp.Enabled	Wahrheitswert	Zum Aktivieren der Abfragen von IBIS-IP Diensten als verbundene Geräte.
Neighbors.IbisIp.ContainerName	Zeichenkette	Der Name des LXC-Containers auf dem avahi-browse installiert ist. Anleitung zum erstellen des Containers: https://wiki.netmodule.com/app-notes/avahi-on-lxc (Standard: guest0)
Neighbors.IbisIp.Protocol	Zeichenkette	Über welches Protokoll die Befehle vom Zugehörigen Gerät auf den Trapeze Gerät ausgeführt werden sollen. Gültige Werte: http und https (Standard: http)
Neighbors.IbisIp.Subnet.Allowed	Array von Zeichenketten	Liste von Subnetzen/IP-Adressen, an die sich das Zugehörige Gerät mit dem Trapeze Gerät verbinden darf. Gültige Werte: Subnetze z.B. "192.168.0.0/16" IP-Adressen, z.B. "192.168.10.24" Folgende Aliasse: "any" ("0.0.0.0/0", "::/0") "*" (Selbes wie "any") "private" ("10.0.0.0/8", "172.16.0.0/12", "192.168.0.0/16", "fd00::/8") "multicast" ("224.0.0.0/4", "ff00::/8") "linklocal" ("169.254.0.0/16", "fe80::/10")

		"localhost" ("127.0.0.0/8", "::1") "documentation" ("192.0.2.0/24", "198.51.100.0/24", "203.0.113.0/24", "2001:db8::/32") "system" (Alle Werte von "multicast", "linklocal", "localhost" und "documentation")
Neighbors.IbisIp.Retries	Ganzzahl	Wie oft IBIS-IP Abfragen versucht werden sollen (Standard: 5)
Neighbors.IbisIp.Timeout	Ganzzahl	Wie lange auf eine IBIS-IP Abfrage gewartet werden soll in Sekunden (Standard: 5)
Neighbors.IbisIp.IgnoreSSL	Wahrheitswert	Ob SSL bei IBIS-IP Abfragen ignoriert werden soll

Netmodule (FW ≤4.0)

Abwärtskompatibler Treiber für Router der NB-Serie des Herstellers NetModule, die eine Firmware-Version bis einschließlich 4.0 verwenden. Dieser unterscheidet sich vom Treiber „Netmodule (alle Modelle)“ nur darin, dass die Option „Update.Config.Partial“ gemäß früherer Gerätesemantik stets deaktiviert ist und sämtliche Unteroptionen von „Update.Config“ daher nicht greifen.

Mit Einführung der administrativ definierbaren Treiberoptionen in NMS v1.22.0 wurde dieser Treiber obsolet, da er durch den o.g. Standardtreiber mit entsprechend definierter Treiberoption ersetzt werden kann.

Technischer Treibername

NetmoduleCcm

Unterstützte Kommunikationsprotokolle

- SSH
- SNMP v1/v2/v3

Funktionsumfang

- Gerätereustart
- Initiieren von Konfigurationsupdates, die das Endgerät vom NMS beziehen kann
- Initiieren von Firmware-Updates, die das Endgerät vom NMS beziehen kann
- Beeinflussung des Standardwertverhaltens beim Konfigurationsupdate
- Ausführen von im NMS gelagerten Shell-Skripten vor und nach den o.g. Update-Prozessen, um mittels Bedingungen und zusätzlichen Aktionen die Updateprozesse bedarfsgerecht steuern und anpassen zu können (nur via SSH-Protokoll)
- Download von Support-/Debug-Informationen, die das Gerät bereitstellt (nur via SSH-Protokoll)
- Abfrage von detaillierten Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit, Systemkomponenten, Konfigurations- und Firmware-Zustände, Netzwerkkonnektivität, lokale Systembenutzer, Standortdaten, Volumina des Datenverkehrs und -verbrauchs)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
PingBeforeStatusQuery	Wahrheitswert	Vor der eigentlichen Statusabfrage wird das Gerät mittels Ping auf Verfügbarkeit geprüft. Ist es per Ping nicht erreichbar findet keine weitere Abfrage statt.
Update.Config.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für ein Konfigurationsupdate auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird i.d.R. nur bei Verwendung von Pre-/Post-Update-Skripten benötigt, da insbesondere für die Ausführung des Post-Skriptes der Abschluss des Updates abgewartet werden muss. Auf Geräten mit schwächerer CPU

		kann dieser Prozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Update.Firmware.SshScriptPre <i>(nur möglich mit SSH als Kommunikationsprotokoll)</i>	Ganzzahl	NMS-ID eines (Shell-)Skriptes, das zur Ausführung vor einem Firmware-Update vorgesehen ist. Das Skript kann mittels Exit-Code (0 = Erfolg; sonst Fehlschlag) signalisieren, ob der Updatevorgang fortgesetzt werden soll oder nicht.
Update.Firmware.SshScriptPost <i>(nur möglich mit SSH als Kommunikationsprotokoll)</i>	Ganzzahl	NMS-ID eines (Shell-)Skriptes, das zur Ausführung nach einem Firmware-Update vorgesehen ist. Der Exit-Code des Firmware-Updateprozesses (0 = Erfolg; sonst Fehlschlag) wird als erster Kommandozeilenparameter an das Skript übergeben, sodass in weitere Aktionen in Abhängigkeit von dessen Erfolg ausführbar sind. Das Skript selbst kann mittels Exit-Code (0 = Erfolg; sonst Fehlschlag) signalisieren, ob der im Anschluss an ein erfolgreiches Firmware-Update erforderliche Gerätereustart ausgeführt werden soll oder nicht. Im Falle eines fehlgeschlagenen Updates sollte kein Neustart erfolgen.
Update.Firmware.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für ein Firmware-Update auf die angegebene Anzahl von Sekunden (Standard: 600 = 10 Minuten). Ein höherer Timeout wird i.d.R. nur bei Verwendung von Pre-/Post-Update-Skripten benötigt, da insbesondere für die Ausführung des Post-Skriptes der Abschluss des Updates abgewartet werden muss. Auf Geräten mit schwächerer CPU kann dieser Prozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Status.Comparison.Jitter	Ganzzahl	Definiert eine Karenzzeit in Sekunden, die vom gerätelokalen Änderungszeitstempel der Konfiguration abgezogen werden soll (Standard: 30 Sekunden). Diese Einstellung erlaubt den Ausgleich des Zeitunterschieds beim Vergleich zwischen Zeitstempeln der Konfiguration vom Gerät und des Konfigurations-Downloads vom NMS. Sie besagt in etwa, wieviel Zeit das Gerät zwischen Download und Anwendung der Konfiguration benötigen darf.

Status.History.IgnoreOptions	Array von Zeichenketten	Ignoriert die angegebenen Konfigurationsoptionen bei der Entscheidung, ob die bei der Statusabfrage vom Gerät heruntergeladene Konfiguration als geändert betrachtet werden soll. Dadurch kann das Anlegen eines neuen Eintrags in der Konfigurationshistorie verhindert werden, um bei häufig wechselnden Optionen (z.B. „custom.var.*“) das exzessive Anlegen unnötiger Versionshistorieneinträge zu verhindern. Die zu ignorierenden Optionen müssen als Liste Perl-kompatibler regulärer Ausdrücke (PCRE) angegeben werden und ermöglichen dadurch auch das Ignorieren mehrerer gleichartiger Optionen (z.B. mit fortlaufender Nummerierung). Beispielsweise unterdrückt der Eintrag „custom\.var\d+“ die Berücksichtigung aller frei definierbaren Variablen in einer NetModule-Konfiguration. Die PCRE-Syntax entspricht dem Standard der PHP-Dokumentation, die unter https://www.php.net/manual/de/reference.pcre.pattern.syntax.php verfügbar ist.
Support.TechFile.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für den Download eines Support-TechFile-Paketes auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird ggf. benötigt, um langsamere Verbindungen auszugleichen, damit der Download sicher abgeschlossen werden kann. Auf Geräten mit schwächerer CPU kann der Erstellungsprozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.

OpenWrt

Allgemeiner Gerätetreiber für alle Geräte mit OpenWrt als Betriebssystem.

Technischer Treibername

OpenWrt

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Geräteneustart
- Initiieren von Konfigurationsupdates, die das Endgerät vom NMS beziehen kann
- Initiieren von Firmware-Updates, die das Endgerät vom NMS beziehen kann
- Download von Support-/Debug-Informationen, die das Gerät bereitstellt (nur via SSH-Protokoll)
- Abfrage von detaillierten Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit, Konfigurations- und Firmware-Zustände, Netzwerkkonnektivität, lokale Systembenutzer)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
Update.Config.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für ein Konfigurationsupdate auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird i.d.R. nur bei Verwendung von Pre-/Post-Update-Skripten benötigt, da insbesondere für die Ausführung des Post-Skriptes der Abschluss des Updates abgewartet werden muss. Auf Geräten mit schwächerer CPU kann dieser Prozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Update.Firmware.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für ein Firmware-Update auf die angegebene Anzahl von Sekunden (Standard: 600 = 10 Minuten). Ein höherer Timeout wird i.d.R. nur bei Verwendung von Pre-/Post-Update-Skripten benötigt, da insbesondere für die Ausführung des Post-Skriptes der Abschluss des Updates abgewartet werden muss. Auf Geräten mit schwächerer CPU kann dieser Prozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.

Support.TechFile.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für den Download eines Support-TechFile-Paketes auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird ggf. benötigt, um langsamere Verbindungen auszugleichen, damit der Download sicher abgeschlossen werden kann. Auf Geräten mit schwächerer CPU kann der Erstellungsprozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Status.History.IgnoreOptions	Array von Zeichenketten	Ignoriert die angegebenen Konfigurationsoptionen bei der Entscheidung, ob die bei der Statusabfrage vom Gerät heruntergeladene Konfiguration als geändert betrachtet werden soll. Dadurch kann das Anlegen eines neuen Eintrags in der Konfigurationshistorie verhindert werden, um bei häufig wechselnden Optionen (z.B. „custom.var.*“) das exzessive Anlegen unnötiger Versionshistorieneinträge zu verhindern. Die zu ignorierenden Optionen müssen als Liste Perl-kompatibler regulärer Ausdrücke (PCRE) angegeben werden und ermöglichen dadurch auch das Ignorieren mehrerer gleichartiger Optionen (z.B. mit fortlaufender Nummerierung). Beispielsweise unterdrückt der Eintrag „custom\.var\d+“ die Berücksichtigung aller frei definierbaren Variablen in einer NetModule-Konfiguration. Die PCRE-Syntax entspricht dem Standard der PHP-Dokumentation, die unter https://www.php.net/manual/de/reference.pcre.pattern.syntax.php verfügbar ist.
Neighbors.Interface.Whitelist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche an die aufgelisteten Schnittstellen angeschlossen sind.
Neighbors.Interface.Blacklist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche NICHT an die aufgelisteten Schnittstellen angeschlossen sind.

Ping über ein Zwischengerät

Einfacher Gerätetreiber zur Prüfung der Verfügbarkeit eines Gerätes mittels PING über ein anderes Gerät ohne Zugangsdaten.

Technischer Treibername

PingViaIntermediarySwitch

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Abfrage der Verfügbarkeit mittels PING über ein Zwischengerät

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

SES-imagotag ePaper Services

Steuert ePaper-/eInk-Displays des Herstellers „SES-imagotag“, die z.B. von der Firma LANCOM als Teil einer Digital-Signage-Lösung vertrieben werden.

Technischer Treibername

ImagotagEpaper

Unterstützte Kommunikationsprotokolle

- HTTP/S

Funktionsumfang

- Rendering von Anzeigeinhalten
- Abrufen von Vorschaubildern
- Abfrage von grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Batteriestatus, Firmwareversion, Netzwerkkonnektivität / Empfangssignalpegel)

Treiberoptionen

Dieser Treiber stellt keine konfigurierbaren Treiberoptionen bereit.

Stahl OpenVPN-Dienstanbieter

Steuert Linux-basierte Maschinen, die OpenVPN-Dienste nach Vorgaben der Stahl Netzmanagement GmbH bereitstellen.

Technischer Treibername

StahlOpenvpn

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Geräteneustart
- Diagnosepaket herunterladen
- Ablage/Installation sowie Löschen/Zurückziehen von Zertifikaten
- Ablage/Abholung/Synchronisation von Zertifikaten, die von einer Stahl-konformen Zertifizierungsbehörde erzeugt wurden bzw. verwaltet werden (siehe u.g. EasyRSA-Treiber)
- Lesen und Schreiben von OpenVPN-Konfigurationsdateien für den Serverdienst und die sich hiermit verbindenden Clientgeräte
- Statusabfrage (Verfügbarkeit, Uptime, Netzwerkkonnektivität, lokale Systembenutzer)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
PathToClientConfig	Zeichenkette	Der Pfad zu den ClientConfigs (CCD) %s wird durch den Dateinamen ersetzt. Default: „/etc/openvpn/client/%s“
PathToServerConfig	Zeichenkette	Der Pfad zur Serverkonfiguration, %s wird durch den Dateinamen ersetzt. Default: „/etc/openvpn/server/%s.conf“
PingBeforeStatusQuery	Wahrheitswert	Vor der eigentlichen Statusabfrage wird das Gerät mittels Ping auf Verfügbarkeit geprüft. Ist es per Ping nicht erreichbar findet keine weitere Abfrage statt.
Support.TechFile.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für den Download eines Support-TechFile-Paketes auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird ggf. benötigt, um langsamere Verbindungen auszugleichen, damit der Download sicher abgeschlossen werden kann. Auf Geräten mit schwächerer CPU kann der Erstellungsprozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.

Stahl OpenVPN-Instanz

Steuert einzelne Instanzen eines OpenVPN-Dienstes.

Technischer Treibername

StahlOpenVpnInstance

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
ConfigFilePrefix	Zeichenkette	Bestimmt einen Prefix für die VPN Client Konfigurationsdatei z.B. um eine Gruppen-basierte Ordnerstruktur zu schaffen.
ConfigOptionMap	Objekt	Eine Abbildung um NMS Konfigurationsoptionen mit OpenVPN Konfigurationsoptionen zu verknüpfen. z.B.: "\$ovpn.iroute": "iroute", "\$ovpn.push_route": "push route", "\$ovpn.ifconfig_push": "ifconfig-push"
ServiceName	Zeichenkette	Name des open vpn Service vor dem '@' Zeichen. Standard: 'openvpn-server'
UseSudo	Wahrheitswert	Ob die Befehle zur Steuerung dieses Treibers über 'sudo' ausgeführt werden sollen. Standard: inaktiv

Stahl Zertifizierungsdienste (EasyRSA v2/3)

Steuert Linux-basierte Maschinen, die mittels des Drittsoftware-Paketes „EasyRSA“ (Version 2.x / 3.x) Dienste einer PKI (Public-Key-Infrastruktur) nach Vorgaben der Stahl Netzmanagement GmbH bereitstellen.

Technischer Treibername

StahlErsa2Ca / StahlErsa3Ca

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Geräteneustart
- Erzeugung von PKI-Zertifikatsbehörden („CA-Zertifikate“)
- Erzeugung von Client-Zertifikaten für CAs, die auf der verwalteten Maschine erzeugt oder hinterlegt wurden
- Auslesen von CA- und Client-Zertifikaten, die auf der verwalteten Maschine erzeugt oder hinterlegt wurden
- Zurückziehen bzw. Löschen von Client-Zertifikaten verwalteter PKI-Zertifikatsbehörden sowie Aktualisierung der zugehörigen Zertifikatssperrlisten
- Statusabfrage (Verfügbarkeit, Uptime, Netzwerkkonnektivität, lokale Systembenutzer)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
PingBeforeStatusQuery	Wahrheitswert	Vor der eigentlichen Statusabfrage wird das Gerät mittels Ping auf Verfügbarkeit geprüft. Ist es per Ping nicht erreichbar findet keine weitere Abfrage statt.
PathToPKI	Zeichenkette	Pfad zum Verzeichnis in welchem die PKI abgelegt wird. Dabei wird %s durch die PKI ersetzt. Der Standardwert ist /etc/pki/%s/
PathToEasyRSA	Zeichenkette	Pfad zum easy-rsa Tool. z.B.: Debian: /usr/share/easy-rsa/easyrsa RedHat: /usr/share/easy-rsa/3/easyrsa

Teltonika RUT

Gerätetreiber zur Steuerung von Routern der RUT-Serie des Herstellers Teltonika.

Technischer Treibername

TeltonikaRut

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Gerätereustart
- Diagnosepaket herunterladen
- Initiieren von Konfigurationsupdates, die das Endgerät vom NMS beziehen kann
- Initiieren von Firmware-Updates, die das Endgerät vom NMS beziehen kann
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Standortdaten, Systemgesundheit, Konfigurations- und Firmware-Zustände)
- Abfrage von verbundenen Geräten

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
PingBeforeStatusQuery	Wahrheitswert	Vor der eigentlichen Statusabfrage wird das Gerät mittels Ping auf Verfügbarkeit geprüft. Ist es per Ping nicht erreichbar findet keine weitere Abfrage statt.
Neighbors.Interface.Whitelist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche an die aufgelisteten Schnittstellen angeschlossen sind.
Neighbors.Interface.Blacklist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche NICHT an die aufgelisteten Schnittstellen angeschlossen sind.
Support.TechFile.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für den Download eines Support-TechFile-Paketes auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird ggf. benötigt, um langsamere Verbindungen auszugleichen, damit der Download sicher abgeschlossen werden kann. Auf Geräten mit schwächerer CPU kann der Erstellungsprozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.

Trapeze IBIS-IP

Gerätetreiber zur Steuerung Geräte des Herstellers Trapeze über IBIS-IP.

Dieser Gerätetreiber benötigt einen Netmodule Router als Zugehöriges Gerät. Dieser muss einen LXC-Container bereitstellen auf dem avahi-browse installiert ist. Dafür kann dieser Anleitung von Netmodule bis einschließlich Schritt 3 gefolgt werden: <https://wiki.netmodule.com/app-notes/avahi-on-lxc>

Technischer Treibername

Trapezelbisl

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit)
- Abfrage von verbundenen Geräten

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
Parent.ContainerName	Zeichenkette	Der Name des LXC-Containers (Standard: guest0)
Parent.MinUptimeRequired	Ganzzahl	Wie hoch die Uptime des zugehörigen Geräts mindestens sein muss, bevor für dieses Gerät eine Statusabfrage durchgeführt wird.
Protocol	Zeichenkette	Über welches Protokoll die Befehle vom Zugehörigen Gerät auf den Trapeze Gerät ausgeführt werden sollen. Gültige Werte: http und https (Standard: http)
Subnet.Allowed	Array von Zeichenketten	Liste von Subnetzen/IP-Adressen, an die sich das Zugehörige Gerät mit dem Trapeze Gerät verbinden darf. Gültige Werte: Subnetze z.B. "192.168.0.0/16" IP-Adressen, z.B. "192.168.10.24" Folgende Aliasse: "any" ("0.0.0.0/0", "::/0") "*" (Selbes wie "any") "private" ("10.0.0.0/8", "172.16.0.0/12", "192.168.0.0/16", "fd00::/8") "multicast" ("224.0.0.0/4", "ff00::/8") "linklocal" ("169.254.0.0/16", "fe80::/10") "localhost" ("127.0.0.0/8", "::1") "documentation" ("192.0.2.0/24", "198.51.100.0/24", "203.0.113.0/24", "2001:db8::/32") "system" (Alle Werte von "multicast", "linklocal", "localhost" und "documentation")

Tronteq (10-port ES Series JSON-API)

Für diesen Gerätetreiber liegt aufgrund der umfangreichen Konfigurationsmöglichkeiten ein eigenständiges Referenzhandbuch vor. Achtung: Die JSON-API ist ab der Firmware Version 2.4.0 nicht mehr unterstützt!

Tronteq (10-port ES Series OpenAPI)

Gerätetreiber zur Steuerung von Routern der ES Serie des Herstellers Tronteq ab der Firmware Version 2.4.0.

Technischer Treibername

TronteqEsSwitchOpenApi

Unterstützte Kommunikationsprotokolle

- HTTP/S

Funktionsumfang

- Gerätereustart
- Initiieren von Firmware-Updates, die das Endgerät vom NMS beziehen kann
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Standortdaten, Systemgesundheit, Firmware-Zustände)
- Abfrage von verbundenen Geräten

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
Update.Firmware.Timeout	Ganzzahl	Definiert die Zeitspanne in Sekunden, die dem Firmware Update auf dem Gerät gewährt wird (Standard: 600)
Neighbors.Interface.Whitelist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche an die aufgelisteten Schnittstellen angeschlossen sind.
Neighbors.Interface.Blacklist	Array von Zeichenketten	Es werden nur verbundene Geräte abgefragt, welche NICHT an die aufgelisteten Schnittstellen angeschlossen sind.

Wavecom

Gerätetreiber zur Steuerung von Geräten des Herstellers Wavecom.

Technischer Treibername

Wavecom

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Geräteneustart
- Initiieren von Konfigurationsupdates, die das Endgerät vom NMS beziehen kann
- Abfrage von detaillierten Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit, Konfigurations- und Firmware-Zustände, Netzwerkkonnektivität, Volumina des Datenverkehrs und -verbrauchs)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
Update.Config.Timeout	Ganzzahl	Erhöht temporär den Verbindungs-Timeout für ein Konfigurationsupdate auf die angegebene Anzahl von Sekunden (Standard: 180 = 3 Minuten). Ein höherer Timeout wird i.d.R. nur bei Verwendung von Pre-/Post-Update-Skripten benötigt, da insbesondere für die Ausführung des Post-Skriptes der Abschluss des Updates abgewartet werden muss. Auf Geräten mit schwächerer CPU kann dieser Prozess mehr Zeit in Anspruch nehmen, sodass der genaue Timeout vorab ausgelotet werden sollte.
Status.History.IgnoreOptions	Array von Zeichenketten	Ignoriert die angegebenen Konfigurationsoptionen bei der Entscheidung, ob die bei der Statusabfrage vom Gerät heruntergeladene Konfiguration als geändert betrachtet werden soll. Dadurch kann das Anlegen eines neuen Eintrags in der Konfigurationshistorie verhindert werden, um bei häufig wechselnden Optionen (z.B. „custom.var.*“) das exzessive Anlegen unnötiger Versionshistorieneinträge zu verhindern. Die zu ignorierenden Optionen müssen als Liste Perl-kompatibler regulärer Ausdrücke (PCRE) angegeben werden und ermöglichen dadurch auch das Ignorieren mehrerer gleichartiger Optionen (z.B. mit fortlaufender Nummerierung). Beispielsweise unterdrückt der Eintrag „custom\.var\d+“ die Berücksichtigung aller frei definierbaren Variablen in einer NetModule-Konfiguration.

Die PCRE-Syntax entspricht dem Standard der PHP-Dokumentation, die unter <https://www.php.net/manual/de/reference.pcre.pattern.syntax.php> verfügbar ist.

Westermo (RT SW 6)

Gerätetreiber zur Steuerung von Routern des Herstellers Westermo, die mit der Firmware „RT SW“ in der Version 6 ausgestattet sind.

Technischer Treibername

WestermoRtsw6

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Gerätereuestart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Uptime, Systemgesundheit, Konfigurations- und Firmware-Zustände)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)
Status.Config.Timeout	Ganzzahl	Definiert die Zeitspanne in Sekunden, die dem Konfigurationsdienst zum Abfragen des Inhalts der Konfiguration auf dem Gerät gewährt wird (Standard: 10). Die ideale Zeitspanne hängt i.d.R. von der CPU-Leistung des Geräts und der Datenmenge des Konfigurationsinhalts ab. Ist diese zu niedrig, kann u.U. keine Konfiguration ausgelesen werden. Ist sie zu hoch, entstehen meist unnötige Verzögerungen, die sich negativ auf die Gesamtsystemperformance auswirken können. Der Standardwert wurde so gewählt, dass er für die meisten Geräte als idealer Kompromiss gilt. Sollte die eigene Beobachtung ergeben, dass keine Konfigurationen in der Historie abgelegt werden, sollte der Timeout testweise erhöht werden, um einen idealen Wert auszuloten.

Westermo (WeOS 4)

Gerätetreiber zur Steuerung WeOS-4-kompatibler Geräte des Herstellers Westermo.

Technischer Treibername

WestermoOs4

Unterstützte Kommunikationsprotokolle

- SSH

Funktionsumfang

- Geräteneustart
- Abfrage grundlegender Gerätestatusdaten (z.B. Verfügbarkeit, Systemgesundheit, Konfigurations- und Firmware-Zustände)

Treiberoptionen

OPTIONSNAME / -PFAD	TYP	BESCHREIBUNG
Debug	Wahrheitswert	Aktiviert den treiberinternen Debug-Modus (Standard: inaktiv). Bei Aktivierung werden z.B. die Ergebnisse ausgeführter SSH-Kommandos in der Anwendungs-Logdatei protokolliert. (Vorsicht: Datenmenge! Performance!)

Konfigurationsverteilung

Funktionsbeschreibung

Eine der Kernfunktionen des NMS ist die Bereitstellung von Gerätekonfigurationen. Diese werden aus Vorlagen – sog. „Templates“ – erzeugt, die sich im „Baukastenprinzip“ aus einzelnen Teilkonfigurationen („Template-Schnipseln“) zusammensetzen lassen.

In den Templates und Template-Schnipseln können zur Parametrisierung bzw. zum Auslösen von Seiteneffekten Ausdrücke einer NMS-spezifischen Skriptsprache benutzt werden. Diese erlauben z.B. die Einbettung von dynamisch erzeugten Geräteinformationen, gerätebezogenen Objekten wie Passwörtern o.ä. in die finale Konfiguration und können damit verbundene Prozesse automatisieren, beispielsweise die Allokation von Netzwerken und darin verfügbaren IP-Adressen.

Mittels eines Templates können mit Massenfunktionen Geräte erzeugt sowie die für sie generierten Konfigurationen bei Änderungen am Template aktualisiert und anschließend auf die Geräte ausgerollt werden.

Dabei spielen die folgenden Bestandteile eine zentrale Rolle:

Konfiguration

- Definition gerätespezifischer Betriebsparameter („Konfigurationsoptionen“)
- Zuordnung zu genau einem Gerät
- Mehrfache Instanzen durch Erzeugung neuer Versionen per Aktualisierung
- Generierung im NMS aus genau einem Template
- Konfigurationsoptionen liegen als Objektstruktur vor
- Objektstruktur enthält Bezüge zu NMS-Objekten, die zum Rendern einer Konfigurationsdatei herangezogen wurden

Konfigurationsdatei

- Liegt i.d.R. in hersteller- / gerätespezifischem Format vor
- Enthält Konfigurationsoptionen in Form von menschenlesbaren Zeichenketten oder einem maschinenlesbaren, gerätespezifischen Binärformat
- Kein Rückbezug zu NMS-Objekten, die zur Generierung herangezogen wurden
- Wird zu benutzerdefinierten Zeitpunkten aus objektbasierter Konfiguration gerendert
- Gerendeter Dateiinhalte liegt immer zusammen mit der objektbasierten Konfiguration vor

Template

- Definition von Konfigurationsoptionen, i.d.R. für mehrere Geräte identisch
- Vorlage zur Erstellung von Konfigurationen für i.d.R. mehrere gleichartige Geräte
- Konfigurationsoptionen werden entweder mit endgültigen Werten oder mit wohlgeformten Ausdrücken einer NMS-internen Skriptsprache belegt
- Änderungen am Template sind auf daraus bereits erzeugte Konfigurationen übertragbar
- Werden bei Geräteanlage zu einer konkreten Konfiguration ausgeleitet
- Können vorhandene, bereits aus demselben Template erzeugte Konfigurationen durch Ausleitung einer neuen Konfigurationsversion aktualisieren

Template-Schnipsel

- Steuern bei der Ausleitung einen eigenen Satz von Konfigurationsoptionen bei
- Zuordnung zu einem oder mehreren Templates bzw. Schnipsel(n)
- Unterstützung additiver Vererbung (Überschreiben und Ergänzen) von Konfigurationsoptionen durch hierarchische Organisation der Schnipsel
- Direkte Zuweisung zu einem Gerät ist nicht erlaubt
- Aufbau und Verwendung entsprechen einem regulären Template

Skriptausrücke

- Dienen der Parametrisierung der Werte von Konfigurationsoptionen
- Erlauben den Zugriff auf im NMS verwaltete Sekundärobjekte (Zugangsdaten, Netzwerke / IP-Adressen, Zertifikate, ...)
- Ermöglichen das Einbinden variabler Information, z.B. fortlaufender Gerätenummern oder an Sekundärobjekten gespeicherter Metadaten
- Basieren auf einem Satz vordefinierter und für Spezialanforderungen erweiterbarer Tokens
- Können zur Abdeckung komplexerer Szenarien verschachtelt werden

Verwendung

Skriptausrücke können ausschließlich als Bestandteil des Wertes einer Konfigurationsoption benutzt werden. Die Verwendung im Optionsnamen ist nicht zulässig und wird bei der Generierung ignoriert.

Skriptausrücke dürfen an beliebiger Stelle innerhalb eines Konfigurationswertes vorkommen und können bei Einhaltung der korrekten Syntax sowohl verkettet als auch verschachtelt werden. Bei der Auswertung von Objekt- und Funktionsnamen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Skriptausrücke verhalten sich ähnlich wie die aus Microsoft Excel ggf. bekannten Formeln. Sie verwenden Objekte (sog. „Tokens“) und Funktionen, deren Zustand bei der Generierung einer Konfiguration ausgewertet und in die Konfiguration eingebettet wird. Das Ergebnis einer Objekt- oder Funktionsauswertung ist stets wieder ein Objekt, auf das weitere Funktionen angewendet werden können („Verkettung“).

Besonderheiten

Manche Skriptausrücke haben Seiteneffekte, mit denen der Datenbestand verändert wird. Werden beispielsweise mittels Skriptausrücken IP-Adressen in einem Subnetz allokiert, legt das System die zugehörigen Adressdatensätze an und bildet die notwendigen Verknüpfungen, d.h. es wird schreibend auf den Datenbestand zugegriffen.

Textänderungen an Skriptausrücken führen zur Neuberechnung des Optionswertes, zu dem sie notiert sind. Ebenso wird eine Neuberechnung auch für sämtliche Optionen durchgeführt, die von der geänderten Option abhängen (z.B. bei der Verwendung von Optionsverweisen). Dabei ist zu beachten, dass durch die Neuberechnung auch die o.g. Seiteneffekte erneut greifen. Damit das System die Datenintegrität gewährleisten kann, ist dieser Prozess unumgänglich.

Formatbeschreibung

- **Ausdruck** := {%= <Objekt>[.<Funktion>[.<Funktion>[...]]] %}
- **Objekt** := <Objektname>(<Parameterliste>)
- **Funktion** := <Funktionsname>(<Parameterliste>)
- **Parameterliste** := [<Parameter>[, <Parameter>[, ...]]]
- **Parameter** := "Zeichenkette" | "<Ausdruck>" | Ziffernfolge
| NULL | TRUE | FALSE

Hinweis: Um in einer Zeichenkette das sonst zur Begrenzung verwendete doppelte Anführungszeichen (") auszugeben, muss dieses durch einen umgekehrten Schrägstrich („Backslash“) als \" maskiert werden.

Dies ist auch dann notwendig, wenn die Zeichenkette einen verschachtelten Ausdruck enthält, der selbst eine Zeichenkette als Parameter erwartet. In den Beispielen „Abfrage von GUI-Geräteparametern (z.B. Seriennummer)“ und „Automatische Anlage von Subnetzen / IP-Adressen“ wird die Verwendung der Maskierung demonstriert.

Objekte und Funktionen

Zugangsdaten

Objekt	<code>Credentials ([Objekt-ID])</code>	
	Liefert ein Benutzer-/Passwort-Objekt zur Abfrage oder Erstellung von Zugangsdaten.	
Parameter	Objekt-ID (Ganzzahl, optional)	Die NMS-ID des abzufragenden Zugangsdaten-Objektes. Wird diese ausgelassen, wird ein leeres Zugangsdaten-Objekt erzeugt, wie es z.B. zur Generierung eines neuen Kennworts benötigt wird.
Ergebnis	Credentials-Objekt	Die neue Instanz des angefragten Zugangsdaten-Objekts.

Funktion	<code>id()</code>	
	Gibt die eindeutige NMS-ID des Zugangsdaten-Objekts zurück.	
Ergebnis	Ganzzahl	Die eindeutige ID des Zugangsdaten-Objekts.

Funktion	<code>user()</code>	
	Gibt den am Zugangsdaten-Objekt gespeicherten Benutzernamen zurück.	
Ergebnis	Zeichenkette	Der gespeicherte Benutzername.

Funktion	<code>password()</code>	
	Gibt das am Zugangsdaten-Objekt gespeicherte Passwort zurück.	
Ergebnis	Zeichenkette	Das gespeicherte Passwort.

Funktion	<code>createdat()</code>	
	Gibt das am Zugangsdaten-Objekt gespeicherte Erstelldatum zurück.	
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.

Funktion	<code>updatedat()</code>	
	Gibt das am Zugangsdaten-Objekt gespeicherte Bearbeitungsdatum zurück.	
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	<code>hash ([Hash-Algorithmus])</code>	
	Gibt einen „salted hash“ des am Zugangsdaten-Objekt gespeicherten Passwortes zurück.	
Parameter	Hash-Algorithmus (Zeichenkette, optional)	Einer der folgenden Hash-Algorithmen: "md5", "sha256", "sha512" Wurde beim Aufruf kein Algorithmus angegeben, so wird der SHA256-Algorithmus als Standard verwendet (kompatibel zu aktuellen Firmwares des Herstellers „NetModule“). Hinweis: Für den zurückgegebenen Hash wird bei jedem Aufruf ein neues, zufälliges Salt generiert. Wird der Hash mit identischem Salt an mehreren Stellen benötigt, sollte diese Funktion nur einmalig aufgerufen und das Ergebnis zur mehrfachen Verwendung in einer Variablen gespeichert werden.
Ergebnis	Zeichenkette	Ein „versalzener Hash“ des gespeicherten Passwortes.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	<code>vardata(<Attributname>, [Standardwert])</code>	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, dem/der das Zugangsdaten-Objekt zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugeordnete Gruppe.

Funktion	generate(<Passwort-Typ>, [Benutzername], [Secret])	
Generiert (ggf. für einen bestimmten Benutzernamen) ein neues Passwort des angegebenen Typs. Sofern für das aktuell zu generierende bzw. zu aktualisierende Gerät die Kombination aus Gerät, Passwort-Typ und Benutzername noch nicht existiert, wird das generierte Kennwort mit den genannten Merkmalen in der NMS-Datenbank gespeichert. Andernfalls das zur bestehenden Kombination gehörige Passwort zurückgegeben, um permanente Passwortänderungen zu unterbinden.		
Parameter	Passwort-Typ (Zeichenkette, erforderlich)	Eine der folgenden Passwort-Arten: "device", "pin", "puk", "snmpv1", "snmpv2", "snmpv3", "user", "radius", "apn", "keypair"
	Benutzername (Zeichenkette, optional)	Der zum Kennwort zu speichernde Benutzername. Wird dieser nicht angegeben, wird der für den Passwort-Typ vordefinierte Standard verwendet. Manche Typen geben einen festen Benutzernamen vor. Der Benutzername kann auch in Form eines Token-Ausdrucks angegeben werden, der auf ein anderes Zugangsdatenobjekt zeigt. Bei vorhandener Leseberechtigung auf dieses Objekt wird der Benutzername aus dem referenzierten Datensatz übernommen (kopiert).
	Secret (Zeichenkette, optional)	Wenn angegeben, wird dieses hartkodierte Secret als Passwort am Datensatz anstelle eines neu generierten Passworts gespeichert. Dies erlaubt z.B. das Setzen einheitlicher, vorgenerierter Passwörter. Das Secret kann auch in Form eines Token-Ausdrucks angegeben werden, der auf ein anderes Zugangsdatenobjekt zeigt. Bei vorhandener Leseberechtigung auf dieses Objekt werden die zum neu zu generierenden Datensatz passenden Daten (Passwort bzw. Schlüsselpaar) aus dem referenzierten Datensatz übernommen (kopiert). Hinweis: Die Übernahme von Schlüsselpaaren ist ausschließlich mittels Referenz auf einen anderen Zugangsdatensatz möglich, da einerseits mehrere Datensatzteile übernommen werden müssen und diese andererseits in einem Format vorliegen, das nicht in jedem Konfigurationsformat darstellbar ist.
Ergebnis	Credentials-Objekt	Eine Instanz des generierten oder bereits gespeicherten Zugangsdatensatzes.

Kunden / Gruppen

Objekt	Customer ([Objekt-ID])	
Liefert ein Objekt zur Abfrage von Kunden- / Gruppeninformationen.		
Parameter	Objekt-ID (Ganzzahl, optional)	Die NMS-ID des abzufragenden Geräte-Objektes. Wird diese ausgelassen, bezieht sich das Objekt auf die Gruppe des aktuell zu generierenden bzw. zu aktualisierenden Gerätes.
Ergebnis	Customer-Objekt	Die neue Instanz des angefragten Gruppen-Objekts.

Funktion	id()	
Gibt die eindeutige NMS-ID des Gruppenobjekts zurück.		
Ergebnis	Ganzzahl	Die eindeutige ID des Gruppenobjekts.

Funktion	name ()	
Gibt den Namen des Kunden bzw. der Gruppe zurück.		
Ergebnis	Zeichenkette	Der Gruppenname (ohne Hierarchie / Pfadangabe).

Funktion	createdat()	
Gibt das am Gruppen-Objekt gespeicherte Erstelldatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.

Funktion	updatedat ()	
Gibt das am Gruppen-Objekt gespeicherte Bearbeitungsdatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON- kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	vardata(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	parent()	
Ermittelt den übergeordneten Kunden bzw. die übergeordnete Gruppe.		
Ergebnis	Customer-Objekt	Der übergeordnete Kunde bzw. die übergeordnete Gruppe.

Funktion	<code>path([Pfadtrenner])</code>	
Gibt den vollständigen Pfadnamen des Gruppenobjekts zurück.		
Parameter	Pfadtrenner (Zeichenkette, optional)	Die Zeichenkette, die als Trennzeichen zwischen den Pfadelementen (Gruppennamen) verwendet werden soll (Standard: „/“, falls der Parameter nicht angegeben wird).
Ergebnis	Zeichenkette	Der Name der Gruppe inklusive aller übergeordneten Gruppen, jeweils getrennt durch die Pfadtrenner-Zeichenkette.

Gerät

Objekt	Device ([Objekt-ID])	
Liefert ein Objekt zur Abfrage gerätespezifischer Daten.		
Parameter	Objekt-ID (Ganzzahl, optional)	Die NMS-ID des abzufragenden Geräte-Objektes. Wird diese ausgelassen, bezieht sich das Objekt auf das aktuell zu generierende bzw. zu aktualisierende Gerät.
Ergebnis	Device-Objekt	Die neue Instanz des angefragten Geräte-Objekts.

Funktion	id()	
Gibt die NMS-interne ID des Gerätes zurück.		
Ergebnis	Ganzzahl	Die NMS-ID, mit der der Gerätedatensatz eindeutig identifiziert wird.

Funktion	hostname()	
Gibt den am Gerät hinterlegten Hostnamen zurück.		
Ergebnis	Zeichenkette	Der in der NMS-Datenbank gespeicherte Hostname.

Funktion	mgmtAddr ()	
Gibt die zum Gerät hinterlegte Management-IP-Adresse zurück.		
Ergebnis	Zeichenkette	Die in der NMS-Datenbank gespeicherte Management-IP-Adresse.

Funktion	description()	
Gibt die am Gerät hinterlegte ausführliche Beschreibung zurück.		
Ergebnis	Zeichenkette	Der in der NMS-Datenbank gespeicherte Beschreibungstext.

Funktion	serial()	
Gibt die für das Gerät ausgelesene bzw. vergebene Seriennummer zurück.		
Ergebnis	Zeichenkette	Die in der NMS-Datenbank gespeicherte Seriennummer des Geräts.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, dem/der das Gerät zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugordnete Gruppe.

Funktion	model()	
Gibt das Gerätemodell des Gerätes (= Gerätetyp) zurück.		
Ergebnis	Modell-Objekt	Der zugeordnete Gerätetyp.

Funktion	object()	
Gibt das Objekt (z.B. Gebäude, Fahrzeug, Automat, usw.) zurück, in welches das Gerät eingebaut wurde.		
Ergebnis	Objekts-Objekt	Das zugeordnete Einbauobjekt (ehem. „Fahrzeug“).

Funktion	createdat()	
Gibt das am Gerät gespeicherte Erstelldatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.

Funktion	updatedat()	
Gibt das am Gerät gespeicherte Bearbeitungsdatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	credentials(<Passwort-Typ>, [Benutzername], [Beschreibung], [Index])	
Ermittelt für einen gegebenen Passwort-Typ ein Zugangsdaten-Objekt, das mit dem Gerät verknüpft ist.		
Parameter	Passwort-Typ (Zeichenkette, erforderlich)	Eine der folgenden Passwort-Arten: "device", "pin", "puk", "snmpv1", "snmpv2", "snmpv3", "user", "radius", "apn", "keypair", "openvpn", "voice"
	Benutzername (Zeichenkette, optional)	Falls angegeben, werden nur verknüpfte Zugangsdaten zurückgegeben, deren Feld „Benutzername“ den hier übergebenen Wert enthält. Das Filtern nach einer leeren Zeichenkette hat keinen Effekt.
	Beschreibung (Zeichenkette, optional)	Falls angegeben, werden nur verknüpfte Zugangsdaten zurückgegeben, deren Feld „Beschreibung“ den hier übergebenen Wert enthält. Das Filtern nach einer leeren Zeichenkette hat keinen Effekt. Hinweis: Werden beide optionale Filter angegeben, wirken diese zusammen als UND-Verknüpfung.
	Index (Ganzzahl, optional)	Falls angegeben, wird nur ein Zugangsdatensatz zurückgegeben, der dem Index entspricht sofern mehrere gleiche Zugangsdatensätze vorhanden sind. Beispiel: voice.password1={%=device().credentials("voice", null, null, 0).password()%} Gib das erste Passwort des Typs voice aus.
Ergebnis	Credentials-Objekt	Eine Instanz des angefragten Zugangsdaten-Objekts, falls für die Suchparameter ein passendes Objekt gefunden wurde. Andernfalls wird ein Fehlertoken zurückgegeben. Existieren mehrere Treffer für die angegebenen Kriterien/Parameter, wird das erste passende Zugangsdaten-Objekt zurückgegeben.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON- kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Gerätemodell

Objekt	Model ([Objekt-ID])	
Liefert ein Gerätetyp-Objekt.		
Parameter	Objekt-ID (Ganzzahl, erforderlich)	Die NMS-ID des abzufragenden Gerätetyp-Objektes.
Ergebnis	Model-Objekt	Die neue Instanz des angefragten Gerätetyp-Objekts.

Funktion	id()	
Gibt die NMS-interne ID des Gerätemodells zurück.		
Ergebnis	Ganzzahl	Die NMS-ID, mit der der Gerätedatensatz eindeutig identifiziert wird.

Funktion	name ()	
Gibt den Namen des referenzierten Gerätemodells zurück.		
Ergebnis	Zeichenkette	Der in der NMS-Datenbank gespeicherte Name des Modells, z.B. „NB2800“ Hinweis: Da die Gerätetypen / Modellbezeichnungen vom Systemadministrator gepflegt werden können, kann (z.B. beim Vergleich von Zeichenketten) nicht von „echten“ Herstellerbezeichnungen ausgegangen werden.

Funktion	meta ()	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON- kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	vardata(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	<code>internalNumber([Formatdefinition])</code>	
Liefert die bei der Geräteerzeugung vom NMS zugewiesene fortlaufende Nummer des Geräts.		
Parameter	Formatdefinition (Zeichenkette, optional)	Optionale Formatangabe, um die Darstellung der fortlaufenden Zahl zu beeinflussen. Eine Beschreibung der möglichen Formatvarianten kann dem Parameter „format“ der folgenden PHP-Dokumentation entnommen werden: http://de.php.net/manual/de/function.sprintf.php Wird diese Angabe ausgelassen, wird die Zahl als vorzeichen-behaftete Ganzzahl ausgegeben (entspricht der Formatangabe "%d").
Ergebnis	Zeichenkette	Die formatierte, fortlaufende Gerätenummer.

Funktion	simcards(<Geräte-Slot>, [Aktiv], [Provider])	
Ermittelt für einen gegebenen Karten-Slot ein Simkarten-Objekt, das mit dem Gerät verknüpft ist.		
Parameter	Geräte-Slot (Nummer, erforderlich)	Der Geräte-Slot (z.B. 1)
	Aktiv (Wahr/Falsch, optional)	Falls angegeben und mit true definiert werden nur aktive Karten zurückgegeben.
	Provider (Zeichenkette, optional)	Falls angegeben, werden nur SIM-Karten dieses Providers zurückgegeben. Hinweis: Werden beide optionale Filter angegeben, wirken diese zusammen als UND-Verknüpfung.
Ergebnis	Simcard-Objekt	Eine Instanz des angefragten Simcard-Objekts, falls für die Suchparameter ein passendes Objekt gefunden wurde. Andernfalls wird ein Fehlertoken zurückgegeben. Existieren mehrere Treffer für die angegebenen Kriterien/Parameter, wird das erste passende Zugangsdaten-Objekt zurückgegeben.

IP-Adresse

Objekt	Ipaddress (<Objekt-ID>)	
Liefert ein IP-Adressobjekt zur Abfrage seiner Eigenschaften.		
Parameter	Objekt-ID (Ganzzahl, erforderlich)	Die NMS-ID des abzufragenden IP-Adressobjektes.
Ergebnis	Ipaddress-Objekt	Die neue Instanz des angefragten IP-Adressobjekts.

Funktion	network()	
Ermittelt das zur Adresse gehörende übergeordnete Subnetz.		
Ergebnis	Subnet-Objekt	Eine neue Instanz des Subnetzes, in dem die Adresse definiert ist.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, dem/der das IP-Adressobjekt zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugeordnete Gruppe.

Funktion	createdat()		
Gibt das am IP-Adressobjekt gespeicherte Erstelldatum zurück.			
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.	

Funktion	updatedat ()	
Gibt das am IP-Adressobjekt gespeicherte Bearbeitungsdatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON- kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	vardata(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Objekt (ehem. „Fahrzeug“)

Objekt	Object (<Objekt-ID>)	
Liefert ein Objekts*-Objekt zur Abfrage seiner Eigenschaften.) Einbauobjekt im Sinne eines Standortes bzw. Containers wie z.B. Gebäude, Fahrzeug, Automat, usw.		
Parameter	Objekt-ID (Ganzzahl, erforderlich)	Die NMS-ID des abzufragenden Objektes.
Ergebnis	Object-Objekt	Die neue Instanz des angefragten Objekts.

Funktion	id()	
Gibt die NMS-interne ID des Objektes zurück.		
Ergebnis	Ganzzahl	Die NMS-ID, mit der der Objektdatensatz eindeutig identifiziert wird.

Funktion	name ()	
Gibt die Kennzeichnung des Objektes zurück (z.B. Gebäudename, Fz.-Kennzeichen, o.ä.).		
Ergebnis	Zeichenkette	Die im System hinterlegte Kennzeichnung des Objekts.

Funktion	type()	
Gibt den Objekttyp zurück (z.B. Gebäude, Bus, Automat, o.ä.).		
Ergebnis	Zeichenkette	Die Textrepräsentation des Objekttyps.

Funktion	description()	
Gibt die am Objekt hinterlegte ausführliche Beschreibung zurück.		
Ergebnis	Zeichenkette	Der in der NMS-Datenbank gespeicherte Beschreibungstext.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, dem/der das Objekt zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugeordnete Gruppe.

Funktion	createdat()	
Gibt das am Objekt gespeicherte Erstelldatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.

Funktion	updatedat()	
Gibt das am Objekt gespeicherte Bearbeitungsdatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON- kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	vardata(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON- kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Optionsverweis

Objekt	<code>Option([Optionsname])</code>	
Liefert ein Options-Objekt, mit dem auf Werte anderer Konfigurationsoptionen zugegriffen werden kann.		
Parameter	Optionsname (Zeichenkette, optional)	Der vollständige Name der Option, auf die das zu erstellende Objekt Bezug nehmen soll. Wird dieser ausgelassen, bezieht sich das Objekt auf die gerade gerenderte Option, die diesen Aufruf enthält.
Ergebnis	Option-Objekt	Die neue Instanz des angefragten Options-Objekts. Wird mittels Angabe des Optionsnamens eine andere Option referenziert, enthält das zurückgegebene Option-Objekt einen Bezug zum ersten Vorkommen einer Option mit diesem Namen innerhalb der gesamten Template-/Schnipsel-Baumstruktur. (Dies kann zu unerwarteten Effekten bei der gleichzeitigen Nutzung von Vererbung führen.)

Funktion	value()	
Ermittelt den Wert der referenzierten Option.		
Ergebnis	Ergebnistyp der referenzierten Option	Der berechnete Wert der durch das Option-Objekt referenzierten Option. Durch diesen Aufruf wird auf der referenzierten Option ggf. ein zusätzlicher Auswertungsvorgang ausgeführt, der mit Seiteneffekten (z.B. der Anlage von Sekundärobjekten) behaftet sein kann.

Funktion	template()	
Ermittelt das Template bzw. den Schnipsel, durch den die referenzierte Option erzeugt wurde.		
Ergebnis	Template-Objekt	Eine neue Instanz des Template-Objektes.

Funktion	findByName (<Optionsname>)	
Liefert ein Options-Objekt, mit dem auf Werte anderer Konfigurationsoptionen zugegriffen werden kann.		
Parameter	Optionsname (Zeichenkette, erforderlich)	Der vollständige Name der Option, auf die das zu erstellende Objekt Bezug nehmen soll.
Ergebnis	Option-Objekt	Die neue Instanz des angefragten Options-Objekts.

Skript

Objekt	Script (<Objekt-ID>)	
Liefert ein Skript-Objekt zur Abfrage von Eigenschaften.		
Parameter	Objekt-ID (Ganzzahl, erforderlich)	Die NMS-ID des abzufragenden Skript-Objektes.
Ergebnis	Script-Objekt	Die neue Instanz des angefragten Skript-Objekts.

Funktion	name ()	
Gibt den Namen des referenzierten Skripts zurück.		
Ergebnis	Zeichenkette	Der in der NMS-Datenbank gespeicherte Name des Skripts.

Funktion	filename()	
Gibt den Dateinamen des referenzierten Skripts zurück.		
Ergebnis	Zeichenkette	Der in der NMS-Datenbank gespeicherte Dateiname des Skripts.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, dem/der das Skript zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugeordnete Gruppe.

Funktion	createdat()	
Gibt das am Skript-Objekt gespeicherte Erstelldatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.

Funktion	updatedat ()	
Gibt das am Skript-Objekt gespeicherte Bearbeitungsdatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	<code>meta(<Attributname>, [Standardwert])</code>	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	<code>vardata(<Attributname>, [Standardwert])</code>	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

SIM-Karten

Objekt	Simcard(<Objekt-ID>)	
Liefert ein SIM-Karten Objekt zur Abfrage von Eigenschaften.		
Parameter	Objekt-ID (Ganzzahl, erforderlich)	Die NMS-ID des abzufragenden Simcard-Objektes.
Ergebnis	Simcard-Objekt	Die neue Instanz des angefragten Skript-Objekts.

Funktion	vendor()	
Gibt den Herausgeber der jeweiligen SIM-Karte zurück.		
Ergebnis	Zeichenkette	Der in der NMS-Datenbank gespeicherte Herausgeber der SIM-Karte.

Funktion	active()	
Gibt den Status Aktiv/Inaktiv der jeweiligen SIM-Karte zurück.		
Ergebnis	Wahr/Falsch	Der in der NMS-Datenbank gespeicherte Aktiv-Status der SIM-Karte.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, welcher der SIM-Karte zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugeordnete Gruppe.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON- kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	vardata(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	<code>credentials(<Passwort-Typ>, [Beschreibung])</code>	
Ermittelt für einen gegebenen Passwort-Typ ein Zugangsdaten-Objekt, das mit dem Gerät verknüpft ist.		
Parameter	Passwort-Typ (Zeichenkette, erforderlich)	Eine der folgenden Passwort-Arten: "pin", "puk", "apn"
	Beschreibung (Zeichenkette, optional)	Falls angegeben, werden nur verknüpfte Zugangsdaten zurückgegeben, deren Feld „Beschreibung“ den hier übergebenen Wert enthält. Das Filtern nach einer leeren Zeichenkette hat keinen Effekt.
Ergebnis	Credentials-Objekt	Eine Instanz des angefragten Zugangsdaten-Objekts, falls für die Suchparameter ein passendes Objekt gefunden wurde. Andernfalls wird ein Fehlertoken zurückgegeben. Existieren mehrere Treffer für die angegebenen Kriterien/Parameter, wird das erste passende Zugangsdaten-Objekt zurückgegeben.

Zeichenketten

Objekt	String()	
Liefert ein Objekt zur Manipulation von Zeichenketten.		
Ergebnis	String-Objekt	Die neue Instanz des angefragten Zeichenketten-Objekts.

Funktion	<code>concat(<Teilzeichenkette>[, Teilzeichenkette[, ...]])</code>	
Verbindet mehrere Teilzeichenketten zu einer einzigen Gesamtzeichenkette.		
Parameter	Teilzeichenkette (Zeichenketten, optional)	Die zusammenzufügenden Teilzeichenketten. Hiervon können beliebig viele – getrennt als einzelne Parameter – angegeben werden.
Ergebnis	Zeichenkette	Die zu einer Gesamtzeichenkette zusammengeführten Teilzeichenketten. Die Reihenfolge beim Zusammenfügen entspricht der Reihenfolge in der Parameterliste.

Funktion	format(<Formatdefinition>[, Datenwert[, ...]])	
Füllt einen mit Platzhaltern versehenen Vorlagentext mit konkreten, formatierten Datenwerten aus.		
Parameter	Formatdefinition (Zeichenkette, erforderlich)	Formatangabe, nach der die in den weiteren Parametern übergebenen Datenwerte formatiert werden sollen. Eine Beschreibung der möglichen Formatvarianten kann dem Parameter „format“ der folgenden PHP-Dokumentation entnommen werden: http://de.php.net/manual/de/function.sprintf.php
	Datenwert (beliebiger Typ, optional)	Die Datenwerte, die an den Platzhalter der Formatdefinition eingefügt werden sollen. Hiervon können beliebig viele – getrennt als einzelne Parameter – angegeben werden. Tipp: Statt einer Liste von einzelnen Datenwerten kann auch ein einzelner Datenwert als Zeichenkette im JSON-Format angegeben werden, wie er z.B. intern bei der Übergabe von ganzen Metadaten-Strukturen verwendet wird. Dies erlaubt eine dynamische Verarbeitung komplexerer Metadaten.
Ergebnis	Zeichenkette	Die in der Formatdefinition vorgegebene Zeichenkette, in der die enthaltenen Platzhalter gegen die angegebenen Datenwerte ausgetauscht wurden.

Funktion	<code>implode(<Verbindungszeichenkette>[, Teilzeichenkette[, ...]])</code>	
Fügt mehrere Teilzeichenketten unter Verwendung einer Verbindungszeichenkette zu einer einzigen Gesamtzeichenkette zusammen.		
Parameter	Verbindungszeichenkette (Zeichenkette, erforderlich)	Die Zeichenkette, die beim Zusammenfügen jeweils zwischen den einzelnen Teilzeichenketten eingefügt werden soll (z.B. Pfad- oder Listentrennzeichen).
	Teilzeichenkette (Zeichenketten, optional)	Die zusammenzufügenden Teilzeichenketten. Hiervon können beliebig viele – getrennt als einzelne Parameter – angegeben werden.
Ergebnis	Zeichenkette	Die mittels Verbindungszeichenkette zu einer Gesamtzeichenkette zusammengefügte Teilzeichenketten. Die Reihenfolge beim Zusammenfügen entspricht der Reihenfolge in der Parameterliste. Leere Teilzeichenketten werden vor Beginn des Zusammenfügens automatisch aus der Parameterliste entfernt, um zu verhindern, dass die Verbindungszeichenkette mehrfach direkt aufeinander folgt. Tipp: Statt einer Liste von einzelnen Teilzeichenketten kann auch ein einzelner Datenwert als Zeichenkette im JSON-Format angegeben werden, wie er z.B. intern bei der Übergabe von ganzen Metadaten-Strukturen verwendet wird. Dies erlaubt eine dynamische Verarbeitung komplexerer Metadaten.

Funktion	<code>implodeSlice(<Verbindungszeichenkette>, <Start-Index>, <Elementanzahl>[, Teilzeichenkette[, ...]])</code>	
Fügt mehrere Teilzeichenketten unter Verwendung einer Verbindungszeichenkette zu einer einzigen Gesamtzeichenkette zusammen. Die übergebenen Teilzeichenketten werden dabei als Array („Feld“, „Vektor“) betrachtet, aus dem nur die angegebene Anzahl von Elementen – beginnend ab dem Start-Index – für das Zusammenfügen herangezogen wird.		
Parameter	Verbindungszeichenkette (Zeichenkette, erforderlich)	Die Zeichenkette, die beim Zusammenfügen jeweils zwischen den einzelnen Teilzeichenketten eingefügt werden soll (z.B. Pfad- oder Listentrennzeichen).
	Start-Index (Ganzzahl, erforderlich)	Null-basierter Index der ersten Teilzeichenkette, die für das Zusammenfügen benutzt werden soll.
	Elementanzahl (Ganzzahl, erforderlich)	Anzahl der Teilzeichenketten, die einschließlich der durch den Start-Index angegebenen Teilzeichenkette zusammengefügt werden sollen. Wird die Elementanzahl mit „0“ angegeben, verwendet die Funktion alle restlichen Teilzeichenketten ab dem Startelement.
	Teilzeichenkette (Zeichenketten, optional)	Die zusammenzufügenden Teilzeichenketten. Hiervon können beliebig viele – getrennt als einzelne Parameter – angegeben werden.
Ergebnis	Zeichenkette	Siehe Funktion „ <code>implode()</code> “. Leere Teilzeichenketten werden auch hier vor Beginn des Zusammenfügens entfernt (jedoch erst nach dem „Herausschneiden“ der Elemente, damit die angegebenen Indizes abzählbar bleiben).

Funktion	isset(<Eingabewert>, <Wahrwert>[, Falschwert])	
Gibt einen Wert zurück, der vom Wahrheitsgehalt eines zu prüfenden Eingabewertes abhängt.		
Parameter	Eingabewert (beliebiger Typ, erforderlich)	Der auf Wahrheitsgehalt zu überprüfende Eingabewert. Leere Zeichenketten und die Zeichenkette „0“ gelten als FALSCH, alle anderen Werte als WAHR.
	Wahrwert (beliebiger Typ, erforderlich)	Der zurückzugebende Wert, wenn der Eingabewert zu WAHR ausgewertet wird.
	Falschwert (beliebiger Typ, optional)	Der zurückzugebende Wert, wenn der Eingabewert zu FALSCH ausgewertet wird.
Ergebnis	Beliebiger Typ	Abhängig vom Wahrheitsgehalt des Eingabewertes wird entweder der übergebene Wahrwert oder Falschwert zurückgegeben. Wert und Typ entsprechen dabei dem bzw. den übergebenen Parameter(n). Wird der optionale Falschwert nicht angegeben, generiert die Funktion als Ergebnis ein Fehlertoken, das die Ausgabe der ganzen zur aktuellen Option gehörenden Zeile unterdrückt. Soll dagegen die Option erhalten bleiben, aber mit einem leeren Wert befüllt werden, sollte eine leere Zeichenkette als Falschwert übergeben werden.

Funktion	default(<Eingabewert>[, Standardwert])	
Gibt einen Standardwert zurück, der vom Wahrheitsgehalt eines zu prüfenden Eingabewerts abhängt.		
Parameter	Eingabewert (beliebiger Typ, erforderlich)	Der auf Wahrheitsgehalt zu überprüfende Eingabewert. Leere Zeichenketten und die Zeichenkette „0“ gelten als FALSCH, alle anderen Werte als WAHR.
	Standardwert (beliebiger Typ, optional)	Der zurückzugebende Wert, wenn der Eingabewert zu FALSCH ausgewertet wird.
Ergebnis	Beliebiger Typ	Abhängig vom Wahrheitsgehalt des Eingabewerts wird entweder der unveränderte Eingabewert (bei wahrem Ergebnis) oder der definierte Standardwert (bei falschem Ergebnis) zurückgegeben. Wert und Typ entsprechen dabei dem bzw. den übergebenen Parameter(n). Wird der optionale Standardwert nicht angegeben, generiert die Funktion als Ergebnis ein Fehlertoken, das die Ausgabe der ganzen zur aktuellen Option gehörenden Zeile unterdrückt. Soll dagegen die Option erhalten bleiben, aber mit einem leeren Wert befüllt werden, sollte eine leere Zeichenkette als Falschwert übergeben werden.

Funktion	compare(<Eingabewert>, <Relation>, <Vergleichswert>, <Wahrwert>[, Falschwert])	
Gibt einen Wert zurück, der vom Vergleichsergebnis einer Eingabe mit einem Vergleichswert abhängt.		
Parameter	Eingabewert (beliebiger Typ, erforderlich)	Die mit dem Vergleichswert zu vergleichende Eingabe.
	Relation (Zeichenkette, erforderlich)	Eine Zeichenkette mit dem Relationszeichen bzw. Schlüsselwort aus der folgenden Liste, das die Art des Vergleichs bestimmt: ▪ == (Gleichheit) ▪ != (Ungleichheit) ▪ <= (Kleiner oder gleich) ▪ >= (Größer oder gleich) ▪ < (Kleiner) ▪ > (Größer) ▪ in (Mengeneinschluss) Vergleiche zwischen Eingabe- und Vergleichswert finden typschwach und zeichenkettenbasiert statt. Es gelten die Vergleichsregeln der zugrundeliegenden PHP-Vergleichsoperator-Dokumentation: https://www.php.net/manual/de/language.operators.comparison.php
	Vergleichswert (beliebiger Typ, erforderlich)	Der Referenzwert, mit dem der Eingabewert verglichen werden soll. Als Vergleichswert für eine Prüfung auf Mengeneinschluss muss eine Zeichenkette im JSON-Format angegeben werden, die ein Array (= „Feld“, „Vektor“) oder ein Objekt beschreibt. (Achtung: <i>Korrektes Maskieren der doppelten Anführungszeichen beachten!)</i> Dieses Array oder Objekt wird als durchsuchbare Menge herangezogen. Eine Formatbeschreibung ist unter dem folgenden Link einsehbar: https://www.w3schools.com/js/js_json_arrays.asp Da Arrays und Objekte, die in Metadaten definiert werden, bei der Konfigurationsgenerierung automatisch als JSON ausgegeben werden, kann auf diese Weise z.B. ein Gültigkeitsbereich definiert werden.
	Wahrwert (beliebiger Typ, erforderlich)	Der zurückzugebende Wert, wenn der Vergleich zu WAHR ausgewertet wird.
	Falschwert (beliebiger Typ, optional)	Der zurückzugebende Wert, wenn der Vergleich zu FALSCH ausgewertet wird.

Ergebnis	Beliebiger Typ	Abhängig vom Ergebnis des Vergleichs von Eingabe- und Vergleichswert wird entweder der übergebene Wahrwert oder Falschwert zurückgegeben. Rückgabewert und -typ entsprechen dabei dem/n übergebenen Parameter(n). Wird der optionale Falschwert nicht angegeben, generiert die Funktion als Ergebnis ein Fehlertoken, das die Ausgabe der ganzen zur aktuellen Option gehörenden Zeile unterdrückt. Soll dagegen die Option erhalten bleiben, aber mit einem leeren Wert befüllt werden, sollte eine leere Zeichenkette als Falschwert übergeben werden.
-----------------	----------------	--

Funktion	tolower(<Zeichenkette>)	
Wandelt eine Zeichenkette in Kleinbuchstaben.		
Parameter	Teilzeichenkette (Zeichenkette, erforderlich)	Die umzuschreibende Zeichenkette.
Ergebnis	Zeichenkette	Die in Kleinbuchstaben gewandelte Eingabezeichenkette. Dabei ist zu beachten, dass in Abhängigkeit von der Ländereinstellung des Server-Systems ggf. Sonderzeichen wie z.B. Umlaute nicht gewandelt werden.

Funktion	toupper(<Zeichenkette>)	
Wandelt eine Zeichenkette in Großbuchstaben.		
Parameter	Teilzeichenkette (Zeichenkette, erforderlich)	Die umzuschreibende Zeichenkette.
Ergebnis	Zeichenkette	Die in Großbuchstaben gewandelte Eingabezeichenkette. Dabei ist zu beachten, dass in Abhängigkeit von der Ländereinstellung des Server-Systems ggf. Sonderzeichen wie z.B. Umlaute nicht gewandelt werden.

Funktion	split(<Trennzeichen>, <Zeichenkette>)	
Gibt ein Array aus Zeichenketten zurück, die jeweils Teil von Zeichenkette sind. Die Abtrennung erfolgt dabei an der mit Trennzeichen angegebenen Zeichenkette.		
Parameter	Trennzeichen (Zeichenkette, erforderlich)	Das Trennzeichen an welchem die Teile aufgeteilt werden.
Parameter	Zeichenkette (Zeichenkette, erforderlich)	Die Zeichenkette welche am Trennzeichen aufgeteilt wird.
Ergebnis	Zeichenkette	Gibt ein Array von Zeichenketten zurück, die durch Aufsplitten des Parameters Zeichenkette an Begrenzungen durch Trennzeichen erzeugt werden.

Funktion	<code>substr(<Zeichenkette>, <Start>[, Länge])</code>	
Gibt einen Teilstring der angegebenen Zeichenkette zurück.		
Parameter	Teilzeichenkette (Zeichenkette, erforderlich)	Die Zeichenkette von welcher ein teilstring zurückgegeben werden soll.
	Start (Ganzzahl, erforderlich)	Ist Start nicht-negativ, beginnt der zurückgegebene String an der Position Start in der Zeichenkette, gezählt ab null. Im String 'abcdef' ist das Zeichen an Position 0 ein 'a', an Position 2 ein 'c', und so weiter. Ist start negativ, beginnt die zurückgegebene Zeichenkette beim Start-ten Zeichen vom Ende der Zeichenkette.
	Länge (Ganzzahl, optional)	Maximale Anzahl der zu verwendenden Zeichen aus der Zeichenkette. Wird der Parameter weggelassen oder NULL übergeben, werden alle Zeichen bis zum Ende der Zeichenkette extrahiert.
Ergebnis	Zeichenkette	Der über die Parameter Start und Länge spezifizierte Teilbereich der eingegebenen Zeichenkette.

Funktion	replace(<Suchzeichenkette>, <Ersatzzeichenkette>, <Zeichenkette>)	
Ersetzt alle vorkommen von Suchzeichenkette mit Ersatzzeichenkette in Zeichenkette.		
Parameter	Suchzeichenkette (Zeichenkette, erforderlich)	Die Zeichenkette dessen vorkommen ersetzt werden soll.
	Ersatzzeichenkette (Zeichenkette, erforderlich)	Die Zeichenkette welche stattdessen eingesetzt werden soll. Leere Zeichenkette übergeben damit die vorkommen ersatzlos entfernt werden.
	Zeichenkette (Zeichenkette, erforderlich)	Die Zeichenkette in welcher nach vorkommen gesucht werden soll.
Ergebnis	Zeichenkette	Die neue Zeichenkette in welcher alle Vorkommen ersetzt wurden.

Subnetz

Objekt	Subnet ([Objekt-ID])	
Liefert ein Subnetz-Objekt zur Abfrage von Eigenschaften oder zur Anlage untergeordneter Objekte.		
Parameter	Objekt-ID (Ganzzahl, optional)	Die NMS-ID des abzufragenden Subnetz-Objektes. Wird diese ausgelassen, wird ein leeres Zugangsdaten-Objekt erzeugt, wie es z.B. zur Generierung eines neuen Kennworts benötigt wird.
Ergebnis	Subnet-Objekt	Die neue Instanz des angefragten Subnetz-Objekts.

Funktion	parent()	
Ermittelt das zum Subnetz gehörende übergeordnete Subnetz.		
Ergebnis	Subnet-Objekt	Eine neue Instanz des Subnetzes, in dem das aktuelle Subnetz definiert ist.

Funktion	cidr()	
Gibt die (IPv4-)Adresse des Subnetzes in CIDR-Notation zurück.		
Ergebnis	Zeichenkette	Die Subnetz-Adresse in CIDR-Notation, z.B. 192.168.0.0/24.

Funktion	maskAddr()	
Gibt die (IPv4-)Maske des Subnetzes zurück.		
Ergebnis	Zeichenkette	Die Subnetz-Maske, z.B. 255.255.255.0.

Funktion	firstAddr()	
Gibt die erste zulässige (IPv4-)Hostadresse des Subnetzes zurück.		
Ergebnis	Zeichenkette	Die erste zulässige Hostadresse des Netzes, z.B. 192.168.0.1.

Funktion	lastAddr()	
Gibt die letzte zulässige (IPv4-)Hostadresse des Subnetzes zurück.		
Ergebnis	Zeichenkette	Die letzte zulässige Hostadresse des Netzes, z.B. 192.168.0.254.

Funktion	broadcastAddr()	
Gibt die (IPv4-)Broadcast-Adresse des Subnetzes zurück.		
Ergebnis	Zeichenkette	Die Subnetz-Maske, z.B. 192.168.0.255.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, dem/der das Subnetz -Objekt zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugeordnete Gruppe.

Funktion	createdat()	
Gibt das am Subnetz-Objekt gespeicherte Erstelldatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.

Funktion	updatedat ()	
Gibt das am Subnetz-Objekt gespeicherte Bearbeitungsdatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	<code>vardata(<Attributname>, [Standardwert])</code>	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	calculateSubnet(<IP-Adresse>, <Netzmaske>)	
Errechnet aus einer gegebenen (IPv4-)Hostadresse und einer (IPv4-)Netzmaske bzw. CIDR-Bitgröße die resultierende (IPv4-)Netzadresse.		
Parameter	IP-Adresse (Zeichenkette, erforderlich)	Die (IPv4-)Hostadresse, deren zugehörige Netzadresse berechnet werden soll.
Parameter	Netzmaske (Zeichenkette oder Ganzzahl, erforderlich)	Die für die Berechnung zu verwendende (IPv4-)Netzmaske oder die entsprechend der CIDR-Notation in der Netzwerkmaske zu setzende Anzahl von Bits. Der gültige Wertebereich bei der Angabe von CIDR-Bits liegt zwischen 0 und 32, um die Berechnung jeglicher Netzadressen (unabhängig von der Sinnhaftigkeit bei der Verwendung) zu ermöglichen.
Ergebnis	Zeichenkette	Die errechnete Netzadresse im IPv4-Format. Da diese Hilfsfunktion lediglich als „Netzrechner“ gedacht ist, werden keine Einträge in der NMS-internen Adressverwaltung angelegt oder modifiziert.

Funktion	indexAddr (<Adress-Index>)	
Gibt die (IPv4-)Hostadresse des Subnetzes mit dem angegebenen Adress-Index zurück.		
Parameter	Adress-Index (Ganzzahl, erforderlich)	Der fortlaufende Index der Adresse. Indizes können vorwärts zählend (positiver Index) oder rückwärts zählend (negativer Index) angegeben werden. Die folgenden Indizes korrelieren mit den Ergebnissen der o.g. Funktionen: ▪ 0: Subnet () (Netzadresse) ▪ 1: firstAddr () (erste Hostadresse) ▪ -2: lastAddr () (letzte Hostadresse) ▪ -1: broadcastAddr () (Broadcast-Adresse) Der Maximalwert für den Index ist von der Größe des Subnetzes abhängig: ▪ Höchste positive Indexnummer $\triangleq$ Anzahl der Adressen im Subnetz minus eins (z.B. 255 bei einem /24er-Netz, entspricht der Broadcast-Adresse) ▪ Niedrigste negative Indexnummer $\triangleq$ (minus) Anzahl der Adressen im Subnetz (z.B. -256 bei einem /24er-Netz, entspricht der Netzadresse) Indizes außerhalb des Subnetz-Adressbereiches liefern bei der Abfrage ein leeres Ergebnis bzw. einen Fehlerwert zurück.
Ergebnis	Zeichenkette	Die Hostadresse des Netzes mit dem gewünschten Adress-Index, z.B. 192.168.0.42. bei einem gegebenen Index von 42.

Funktion	allocateAddr([DNS-Hostname], [Beschreibung], [Link-Name])	
Reserviert die nächste freie (IPv4-)Hostadresse des Subnetzes in der NMS-internen Adressverwaltung und gibt die reservierte Adresse zurück.		
Parameter	DNS-Hostname (Zeichenkette, optional)	Der DNS-Name, der der reservierten Adresse in der NMS-internen IP-Adressverwaltung (IPAM) zugewiesen werden soll.
Parameter	Beschreibung (Zeichenkette, optional)	Wird am internen IPAM-Adresseintrag hinterlegt und kann z.B. für Standort- oder sonstige Informationen genutzt werden.
Parameter	Link-Name (Zeichenkette, optional)	Definiert einen eigenen Kontextnamen für die Verknüpfung zwischen der zu reservierenden Adresse und dem Gerät, für das aktuell eine Konfiguration generiert wird. Dies verhindert z.B. das Erzeugen neuer Adressen bei der Änderung von Options- bzw. Variablennamen in der Vorlage. Wird der Parameter ausgelassen, wird stattdessen der Name der Option verwendet, in der diese Funktion aufgerufen wird. Hinweis: Eine neue Adresse wird nur reserviert, wenn für die Kombination aus Gerät und Link-Name noch kein Adressdatensatz existiert. Andernfalls wird die über die genannte Datenkombination früher bereits reservierte Adresse ermittelt und zurückgegeben. Soll für die bestehende Kombination (z.B. durch Infrastrukturänderungen) wieder eine neue Adresse reserviert und vergeben werden, ist die bestehende Verknüpfung manuell zu entfernen.
Ergebnis	Ipaddress-Objekt	Eine neue Instanz der reservierten Adresse.

Funktion	allocateIndexAddr (<Adress-Index>, [DNS-Hostname], [Beschreibung], [Link-Name], [Skip-Indicies])	
Reserviert – falls nicht bereits vergeben – die (IPv4-)Hostadresse des Subnetzes mit dem angegebenen Adress-Index in der NMS-internen Adressverwaltung und gibt die reservierte Adresse zurück. Sollte eine Hostadresse bereits reserviert sein, wird die Hostadresse direkt zurückgegeben und nicht mehr dem Gerät zugeprndet.		
Parameter	Adress-Index (Ganzzahl, erforderlich)	Der fortlaufende Index der Adresse, analog zur Funktion indexAddr(). Diese enthält eine detaillierte Beschreibung des Parameters.
Parameter	DNS-Hostname (Zeichenkette, optional)	Der DNS-Name, der der reservierten Adresse in der NMS-internen IP-Adressverwaltung (IPAM) zugewiesen werden soll.
Parameter	Beschreibung (Zeichenkette, optional)	Wird am internen IPAM-Adresseintrag hinterlegt und kann z.B. für Standort- oder sonstige Informationen genutzt werden.
Parameter	Link-Name (Zeichenkette, optional)	Definiert einen eigenen Kontextnamen für die Verknüpfung zwischen der zu reservierenden Adresse und dem Gerät, für das aktuell eine Konfiguration generiert wird. Wird der Parameter ausgelassen, wird stattdessen der Name der Option verwendet, in der diese Funktion aufgerufen wird. Hinweis: Eine neue Adresse wird nur reserviert, wenn für die Kombination aus Gerät und Link-Name noch kein Adressdatensatz existiert. Andernfalls wird die über die genannte Datenkombination früher bereits reservierte Adresse ermittelt und zurückgegeben. Soll für die bestehende Kombination (z.B. durch Infrastruktur-änderungen) wieder eine neue Adresse reserviert und vergeben werden, ist die bestehende Verknüpfung manuell zu entfernen.
Parameter	Skip-Indicies (Zeichenkette, optional)	Eine Kommagetrennte Liste von Indizes die übersprungen werden sollen. z.B.: „255,256“ Hinweis: Wenn die internalNumber verwendet als Index wird, muss darauf geachtet werden, dass sie erst mit 1 beginnt und somit die erste X.X.X.0 Adresse immer übersprungen wird.
Ergebnis	Ipaddress-Objekt	Eine neue Instanz der reservierten Adresse.

Funktion	assignIndexAddr(<Adress-Index>, [DNS-Hostname], [Beschreibung], [Link-Name])	
Gleiches Verhalten wie bei allocateIndexAddr ordnet allerdings bei bereits reservierter Hostadresse diese trotzdem dem Gerät zu. Somit sind mehrfache Zuordnungen zu Geräten möglich.		
Parameter	Adress-Index (Ganzzahl, erforderlich)	Der fortlaufende Index der Adresse, analog zur Funktion indexAddr(). Diese enthält eine detaillierte Beschreibung des Parameters.
Parameter	DNS-Hostname (Zeichenkette, optional)	Der DNS-Name, der der reservierten Adresse in der NMS-internen IP-Adressverwaltung (IPAM) zugewiesen werden soll.
Parameter	Beschreibung (Zeichenkette, optional)	Wird am internen IPAM-Adresseintrag hinterlegt und kann z.B. für Standort- oder sonstige Informationen genutzt werden.
Parameter	Link-Name (Zeichenkette, optional)	Definiert einen eigenen Kontextnamen für die Verknüpfung zwischen der zu reservierenden Adresse und dem Gerät, für das aktuell eine Konfiguration generiert wird. Wird der Parameter ausgelassen, wird stattdessen der Name der Option verwendet, in der diese Funktion aufgerufen wird. Hinweis: Eine neue Adresse wird nur reserviert, wenn für die Kombination aus Gerät und Link-Name noch kein Adressdatensatz existiert. Andernfalls wird die über die genannte Datenkombination früher bereits reservierte Adresse ermittelt und zurückgegeben. Soll für die bestehende Kombination (z.B. durch Infrastrukturänderungen) wieder eine neue Adresse reserviert und vergeben werden, ist die bestehende Verknüpfung manuell zu entfernen.
Ergebnis	Ipaddress-Objekt	Eine neue Instanz der reservierten Adresse.

Funktion	allocateSubnet(<CIDR-Bitgröße>, [Name], [Gruppen-ID], [Link-Name])	
Reserviert im aktuellen Subnetz ein neues, kleineres Subnetz, speichert dieses in der NMS-internen Adressverwaltung und gibt das reservierte Subnetz zurück.		
Parameter	CIDR-Bitgröße (Ganzzahl, erforderlich)	Die entsprechend der CIDR-Notation in der Netzwerkmaske zu setzende Anzahl von Bits. Der gültige Wertebereich liegt zwischen 1 und 30, um die Erstellung von Masken zu Einzeladressen und leeren Netzen zu verhindern. Die angegebene Bitgröße für das zu reservierende Subnetz muss dabei größer sein als die Bitgröße des aktuellen Netzes.
Parameter	Name (Zeichenkette, optional)	Wird am internen IPAM-Subnetzeintrag hinterlegt und sollte einen treffenden Namen für das Subnetz beinhalten.
Parameter	Gruppen-ID (Ganzzahl, optional)	NMS-ID des Kunden / der Gruppe, dem bzw. der das neue Subnetz zugeordnet werden soll. Wird die ID nicht angegeben oder die ID eines Objektes angegeben, für das kein Schreibzugriff besteht, so wird die Kunden- / Gruppen-zugehörigkeit von dem Template bzw. Snippet geerbt, das diese Funktion aufruft.
Parameter	Link-Name (Zeichenkette, optional)	Definiert einen eigenen Kontextnamen für die Verknüpfung zwischen dem zu reservierenden Subnetz und dem Gerät, für das aktuell eine Konfiguration generiert wird. Dies verhindert z.B. das Erzeugen neuer Subnetze bei der Änderung von Options- bzw. Variablennamen in der Vorlage. Wird der Parameter ausgelassen, wird stattdessen der Name der Option verwendet, in der diese Funktion aufgerufen wird. Hinweis: Ein neues Subnetz wird nur reserviert, wenn für die Kombination aus Gerät und Link-Name noch kein Netzdatensatz existiert. Andernfalls wird das über die genannte Kombination früher bereits reservierte Subnetz ermittelt und zurückgegeben. Soll für die bestehende Kombination (z.B. durch Infrastruktur-änderungen) wieder ein neues Subnetz reserviert und vergeben werden, ist die bestehende Verknüpfung manuell zu entfernen.
Ergebnis	Subnet-Objekt	Eine neue Instanz des reservierten Subnetzes.

Funktion	allocateIndexSubnet(<Subnet-Index>, <CIDR-Bitgröße>, [Name], [Gruppen-ID], [Link-Name], [Skip-Indicies])	
Reserviert im aktuellen Subnetz das n-te Subnetz mit der angegebenen CIDR-Maske, speichert dieses in der NMS-internen Adressverwaltung und gibt das reservierte Subnetz zurück.		
Parameter	Subnet-Index (Ganzzahl, erforderlich)	Das n-te Subnetz wird reserviert.
Parameter	CIDR-Bitgröße (Ganzzahl, erforderlich)	Die entsprechend der CIDR-Notation in der Netzwerkmaske zu setzende Anzahl von Bits. Der gültige Wertebereich liegt zwischen 1 und 30, um die Erstellung von Masken zu Einzeladressen und leeren Netzen zu verhindern. Die angegebene Bitgröße für das zu reservierende Subnetz muss dabei größer sein als die Bitgröße des aktuellen Netzes.
Parameter	Name (Zeichenkette, optional)	Wird am internen IPAM-Subnetzeintrag hinterlegt und sollte einen treffenden Namen für das Subnetz beinhalten.
Parameter	Gruppen-ID (Ganzzahl, optional)	NMS-ID des Kunden / der Gruppe, dem bzw. der das neue Subnetz zugeordnet werden soll. Wird die ID nicht angegeben oder die ID eines Objektes angegeben, für das kein Schreibzugriff besteht, so wird die Kunden- / Gruppen-zugehörigkeit von dem Template bzw. Snippet geerbt, das diese Funktion aufruft.
Parameter	Link-Name (Zeichenkette, optional)	Definiert einen eigenen Kontextnamen für die Verknüpfung zwischen dem zu reservierenden Subnetz und dem Gerät, für das aktuell eine Konfiguration generiert wird. Dies verhindert z.B. das Erzeugen neuer Subnetze bei der Änderung von Options- bzw. Variablennamen in der Vorlage. Wird der Parameter ausgelassen, wird stattdessen der Name der Option verwendet, in der diese Funktion aufgerufen wird. Hinweis: Ein neues Subnetz wird nur reserviert, wenn für die Kombination aus Gerät und Link-Name noch kein Netzdatensatz existiert. Andernfalls wird das über die genannte Kombination früher bereits reservierte Subnetz ermittelt und zurückgegeben. Soll für die bestehende Kombination (z.B. durch Infrastruktur-änderungen) wieder ein neues Subnetz reserviert und vergeben werden, ist die bestehende Verknüpfung manuell zu entfernen.
Parameter	Skip-Indicies (Zeichenkette, optional)	Eine Kommagetrennte Liste von Indizes die übersprungen werden sollen. z.B.: „255,256“ Hinweis: Wenn die internalNumber verwendet als Index wird, muss darauf geachtet werden, dass sie erst mit 1 beginnt und somit das erste 0-ste Netz immer übersprungen wird.
Ergebnis	Subnet-Objekt	Eine neue Instanz des reservierten Subnetzes.

Template

Objekt	Template ([Objekt-ID])	
Liefert ein Template-Objekt zur Abfrage von Eigenschaften und Generierungsparametern.		
Parameter	Objekt-ID (Ganzzahl, optional)	Die NMS-ID des abzufragenden Template-Objektes. Bei Auslassung der Objekt-ID bezieht sich die zurückgegebene Instanz auf das Template-Objekt, das der aktuell zu generierenden bzw. zu aktualisierenden Option übergeordnet ist.
Ergebnis	Template-Objekt	Die neue Instanz des angefragten Template-Objekts.

Funktion	id()	
Gibt die eindeutige NMS-ID des Template-Objekts zurück.		
Ergebnis	Ganzzahl	Die eindeutige ID des Template-Objekts.

Funktion	name ()	
Gibt den Anzeigenamen des Template-Objekts zurück.		
Ergebnis	Zeichenkette	Der Anzeigenamen der Vorlage bzw. des Schnipsels.

Funktion	customer()	
Gibt den Kunden bzw. die Gruppe zurück, dem/der die Vorlage bzw. der Schnipsel zugeordnet ist.		
Ergebnis	Customer-Objekt	Der zugeordnete Kunde bzw. die zugeordnete Gruppe.

Funktion	createdat()	
Gibt das am Template-Objekt gespeicherte Erstelldatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Erstelldatum.

Funktion	credentials(<Passwort-Typ>, [Benutzername], [Beschreibung], [Index])	
Ermittelt für einen gegebenen Passwort-Typ ein Zugangsdaten-Objekt, das mit dem Gerät verknüpft ist.		
Parameter	Passwort-Typ (Zeichenkette, erforderlich)	Eine der folgenden Passwort-Arten: "device", "pin", "puk", "snmpv1", "snmpv2", "snmpv3", "user", "radius", "apn", "keypair", "openvpn", "voice", "vpnpsk", "wlanpsk"
	Benutzername (Zeichenkette, optional)	Falls angegeben, werden nur verknüpfte Zugangsdaten zurückgegeben, deren Feld „Benutzername“ den hier übergebenen Wert enthält. Das Filtern nach einer leeren Zeichenkette hat keinen Effekt.
	Beschreibung (Zeichenkette, optional)	Falls angegeben, werden nur verknüpfte Zugangsdaten zurückgegeben, deren Feld „Beschreibung“ den hier übergebenen Wert enthält. Das Filtern nach einer leeren Zeichenkette hat keinen Effekt. Hinweis: Werden beide optionale Filter angegeben, wirken diese zusammen als UND-Verknüpfung.
	Index (Ganzzahl, optional)	Falls angegeben, wird nur ein Zugangsdatensatz zurückgegeben, der dem Index entspricht sofern mehrere gleiche Zugangsdatensätze vorhanden sind.
Ergebnis	Credentials-Objekt	Eine Instanz des angefragten Zugangsdaten-Objekts, falls für die Suchparameter ein passendes Objekt gefunden wurde.

		Andernfalls wird ein Fehlertoken zurückgegeben. Existieren mehrere Treffer für die angegebenen Kriterien/Parameter, wird das erste passende Zugangsdaten-Objekt zurückgegeben.
--	--	--

Funktion	updatedat ()	
Gibt das am Template-Objekt gespeicherte Bearbeitungsdatum zurück.		
Ergebnis	Zeichenkette	Das gespeicherte Bearbeitungsdatum.

Funktion	meta(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	vardata(<Attributname>, [Standardwert])	
Liefert den Wert eines benannten, am Objekt gespeicherten System-Metadatenattributs zurück.		
Parameter	Attributname (Zeichenkette, erforderlich)	Der Name des auszulesenden Metadatenattributs.
	Standardwert (beliebiger Typ, optional)	Der von der Funktion zurückzugebende Wert, falls kein Metadatum mit dem angegebenen Attributnamen existiert.
Ergebnis	Zeichenkette, ggf. JSON-kodiert	Der im angegebenen Metadatenattribut gespeicherte Wert bzw. der angegebene Standardwert, falls das Attribut nicht existiert. Da Metadaten neben skalaren Werten (Zahlen, Zeichenketten, Wahrheitswerte) auch komplexere Datenstrukturen (Arrays, Objekte) enthalten können, werden letztere JSON-kodiert ausgegeben, um als Zeichenkette dargestellt bzw. unter Wahrung der Struktur in andere Objektmetadaten übernommen werden zu können.

Funktion	<code>countRendered([Formatdefinition])</code>	
Gibt die Anzahl der bisher mit diesem Template/Schnipsel durchgeführten Generierungen bzw. Aktualisierungen aus.		
Parameter	Formatdefinition (Zeichenkette, optional)	Optionale Formatangabe, um die Darstellung der fortlaufenden Zahl zu beeinflussen. Eine Beschreibung der möglichen Formatvarianten kann dem Parameter „ <i>format</i> “ der folgenden PHP-Dokumentation entnommen werden: http://de.php.net/manual/de/function.sprintf.php Wird diese Angabe ausgelassen, wird die Zahl als vorzeichen-behaftete Ganzzahl ausgegeben (entspricht der Formatangabe "%d").
Ergebnis	Zeichenkette	Die formatierte Anzahl der Rendervorgänge für dieses Template. Der aktuelle Rendervorgang ist in diesem Wert bereits enthalten.

Time

Objekt	Time()	
Liefert ein Time-Objekt zur Ausgabe von zur Generierung aktueller Uhrzeit und/oder Datum.		
Ergebnis	Time-Objekt	Die neue Instanz einer aktuellen Uhrzeit.

Funktion	date()	
Gibt das zur Generierung aktuelle Datum zurück.		
Ergebnis	Zeichenkette	Das aktuelle Datum im Format „yyyy-MM-dd“.

Funktion	time()	
Gibt die zur Generierung aktuelle Uhrzeit zurück.		
Ergebnis	Zeichenkette	Die aktuelle Uhrzeit im Format „HH:mm:ss“.

Funktion	dateTime()	
Gibt das zur Generierung aktuelle Datum und die Uhrzeit zurück.		
Ergebnis	Zeichenkette	Das aktuelle Datum und Uhrzeit im Format „yyyy-MM-dd HH:mm:ss“.

Variablen

Zur Zwischenspeicherung von mehrfach benutzten Werten stehen Variablen zur Verfügung. Auf diese Weise lassen sich bestimmte, mit Seiteneffekten behaftete Operationen lediglich einmalig ausführen, obwohl an mehreren Stellen auf das Ergebnis zurückgegriffen werden muss.

Benötigt wird dies beispielsweise bei der Erstellung eines Subnetzes, in welchem dann mehrere Adressen reserviert werden sollen. Ohne die Nutzung von Variablen zur Zwischenspeicherung des erzeugten Subnetzes würde sonst bei jedem Aufruf eines Skriptausdrucks zur Adressreservierung fälschlicherweise auch ein weiteres Subnetz erstellt werden.

Variablen verhalten sich im Wesentlichen wie Konfigurationsoptionen, werden bei der Generierung jedoch nicht in die Konfigurationsdatei mit ausgegeben.

Format

Variablennamen beginnen stets mit einem Dollar-Zeichen. Der Name muss eine Länge von mindestens einem Zeichen aufweisen und darf aus Groß- und Kleinbuchstaben, Ziffern und dem Unterstrich bestehen.

- **Variable** := \$<Bezeichner>
- **Bezeichner** := <Literal>[<Literal>[<Literal>[...]]]
- **Literal** := A...Z | a...z | 0...9 | _

Spezielle Variablen

Wenn ein Template bzw. Schnipsel zur Neuanlage oder zur Aktualisierung von Geräten ausgewertet wird, können bestimmte Konfigurationswerte für die Anzeige in der Benutzeroberfläche bzw. für die Verwendung innerhalb von NMS-Funktionsmerkmalen übernommen werden.

Auf diese Weise lässt sich z.B. im Template ein fortlaufender Hostname generieren, der in der NMS-Oberfläche ohne weitere Zusätze als Hostname für das verwaltete Gerät angezeigt wird, während er in der fertigen Configuration um eine Domänenangabe zu einem vollqualifizierten Namen (FQDN) ergänzt wird.

Zu diesem Zweck gibt es die folgenden WriteOnly-Variablen mit speziellen, fest definierten Namen, deren Werte am Ende des Generierungsprozesses in die an der Benutzeroberfläche angezeigten Gerätedaten übernommen werden:

Variable	Datentyp	Bedeutung / Funktion
\$dms.configtype	Zeichenkette	Legt das Verwaltungsprotokoll fest, über welches das Gerät vom NMS gesteuert wird. Gültige Werte sind "http", "snmp" und "ssh". Wichtig: Kleinschreibung beachten!
\$dms.description	Zeichenkette	Setzt die ausführliche Beschreibung des Geräts.
\$dms.firmware.id	Ganzzahl	NMS-ID (keine Versionsnummer!) der Firmware, die als Sollwert für das Gerät gelten soll. Hinweis: Dies setzt den Referenzwert für Vergleiche, sodass Statusanzeigen im NMS darauf reagieren können und passive FW-Updates ggf. greifen. Ein aktives Update wird durch das Setzen des Wertes nicht angestoßen.

\$dms.password.id	Ganzzahl	NMS-ID des Zugangsdatenobjekts, welche als primäre Zugangsdaten hinterlegt werden sollen.
\$dms.guiurl	Zeichenkette	Kann einen URL aufnehmen, der im Browser beim Anklicken der angezeigten Management-IP-Adresse aufgerufen wird. Ist dieser Eintrag leer, wird der URL aus der angezeigten Management-IP-Adresse und dem Protokollpräfix „http://“ gebildet.
\$dms.hostname	Zeichenkette	Setzt den für das Gerät angezeigten GUI-Hostnamen.
\$dms.ip	Zeichenkette	Legt die Management-IP-Adresse fest, über die das Gerät vom NMS mithilfe von Steuerbefehlen verwaltet wird.
\$dms.location.lat \$dms.location.lng	Fließkommazahl	Weist dem Gerät einen festen Breiten- (lat) bzw. Längengrad (lng) als Geoposition zu. Dadurch kann Geräten ohne GPS (z.B. in Gebäuden) ein fester Standort zugeteilt werden.
\$dms.meta.<Name>	Beliebig	Ordnet dem mit <Name> benannten Metadatenattribut einen Wert zu. Neben skalaren Werten sind auch komplexere Datenstrukturen im JSON-Format zulässig.
\$dms.vardata.<Name>	Beliebig	Ordnet dem mit <Name> benannten System-Metadatenattribut einen Wert zu. Neben skalaren Werten sind auch komplexere Datenstrukturen im JSON-Format zulässig.
\$dms.number	Ganzzahl	Erlaubt die Beeinflussung der internen, fortlaufenden Gerätenummer, die bei der massenhaften Erzeugung von Geräten mittels Templates vergeben/zugewiesen wird.
\$dms.object.id	Ganzzahl	NMS-ID des Objektes (z.B. Fahrzeug, Gebäude), dem das Gerät zugeordnet werden soll. Das Objekt muss bereits existieren.
\$dms.serial	Zeichenkette	Setzt die für das Gerät hinterlegte Seriennummer.
\$dms.timeout	Ganzzahl	Erlaubt für das zu generierende/aktualisierende Gerät das Setzen eines vom systemweiten Standardwert abweichenden Timeouts für Geräteverwaltungsbefehle (Angabe in Sekunden).

Kommentare

Innerhalb eines Templates oder Schnipsels können Kommentare verwendet werden. Diese werden bei der Generierung nicht in die Konfigurationsdatei mit ausgegeben. Als Kommentare werden alle Zeilen betrachtet, die mit einem Doppelkreuz („Hashtag“, „#“) beginnen.

Achtung: Es dürfen keine führenden Leerzeichen o.ä. vor dem Doppelkreuz vorhanden sein.

Beispiele

Fortlaufende Numerierung

```
# Geräte-Hostnamen fortlaufend mit dreistelliger Ziffernfolge numerieren
network.hostname=CUST-TESTAP{%=device().internalnumber("%03d")%}
network.lan.1.address=192.168.5.1
```

```
# IP-Adresse für Alias aus fortlaufender Nummer generieren
network.lan.1.alias.0.address=10.11.12.{%=device().internalnumber()%}
network.lan.1.alias.0.netmask=255.255.255.255
```

Verwendung von Spezialvariablen

```
# Hostname und Management-IP aus Konfiguration in die GUI übernehmen
$dms.hostname={%=option("network.hostname")%}
$dms.ip={%=option("network.lan.1.address")%}

# Abweichender Link für Management-IP-Feld (Protokoll, Port)
$dms.guiurl=https://{%=option("$dms.ip")%}:981

# Zuweisen einer Firmware-Version "X.Y.Z", wenn diese z.B. die NMS-ID "123" hat
$dms.firmware.id=123

# Zuweisen der primären Zugangsdaten anhand eines Benutzernamens "admin"
$dms.password.id={%=device().credentials("device", "admin").id()%}

# Setzen einer festen Geoposition für Geräte ohne GPS
$dms.location.lat=48.5302515
$dms.location.lng=11.5076867
```

Abfrage von GUI-Geräteparametern (z.B. Seriennummer)

```
# Automatisches, zeitgesteuertes Update für Firmware und Konfiguration
auto.swupdate.status=1
auto.swupdate.time=08:00
auto.swupdate.url=https://download.example.org/firmware/{%=device().serial()%}.img
auto.cfupdate.status=1
auto.cfupdate.time=07:00
auto.cfupdate.url=https://download.example.org/configuration/{%=device().serial()%}.zip
```

Automatische Anlage von Subnetzen / IP-Adressen

Aufgabe:

Im NMS-internen IPAM wurde manuell ein Master-Netz (z.B. „10.0.0.0/8“) angelegt, welches vom System unter der Subnetz-ID „1234“ geführt wird. Innerhalb dieses Master-Netzes sollen nun für jedes neue Gerät zwei Subnetze (Clients, Management) mit genügend Kapazität für 10 Host-Adressen erstellt werden.

In jedem der beiden Netze sollen jeweils 2 Adressen allokiert und in der Konfiguration verwendet werden.

Anmerkungen:

- Gewählte Subnetz-CIDR-Bitgröße = 28, dies ermöglicht 14 Host-Adressen.
- Die u.g. Anweisungen dürfen im Template/Schnipsel keine Zeilenumbrüche enthalten. Hier im Dokument sind die Anweisungen nicht anders darstellbar – daher ggf. manuell korrigieren.

```
# Master-Netz anhand der ID ermitteln
# und für Folgeoperationen in Variable ablegen.
$masternet={%=subnet(1234)%}

# Zwei neue Subnetze im Master-Netz reservieren.
# (Die Netze werden in der hier vorliegenden Reihenfolge reserviert.)
$devnet_clients={%=option("$masternet").allocateSubnet(28)%}
$devnet_mgmt={%=option("$masternet").allocateSubnet(28)%}
```

```
# Adressen reservieren und den gewünschten Optionen zuweisen
network.lan.1.alias.1.address={%=option("$devnet_mgmt").allocateAddr("{%=option(\"$dms.hostname\")}%-mgmt1")%}
network.lan.1.alias.1.netmask={%=option("$devnet_mgmt").maskAddr()%}
network.lan.1.alias.2.address={%=option("$devnet_mgmt").allocateAddr("{%=option(\"$dms.hostname\")}%-mgmt2", "Management: Sekundärer Zugriff")%}
network.lan.1.alias.2.netmask={%=option("$devnet_mgmt").maskAddr()%}
network.lan.1.alias.3.address={%=option("$devnet_clients").allocateAddr("", "Infotainment-System")%}
network.lan.1.alias.3.netmask={%=option("$devnet_clients").maskAddr()%}
network.lan.1.alias.4.address={%=option("$devnet_clients").allocateAddr()%}
network.lan.1.alias.4.netmask={%=option("$devnet_clients").maskAddr()%}
```

Einfache Netzrechnerfunktionen ohne Datensatzbezug

```
# Netzadresse aus Hostadresse anhand der Netzmaske oder CIDR-Bits ermitteln
$net1={%=Subnet().calculateSubnet("192.168.13.42", "255.255.255.0")%}
$net2={%=Subnet().calculateSubnet("192.168.13.42", 24)%}
```

Übernahme und Auslesen von Metadaten

```
# Kundenverantwortliche Person(en) als Sachbearbeiter für das Gerät hinterlegen
$dms.meta.Sachbearbeiter={%=template().customer().meta("Verantwortlich")%}

# Hostnamen des Geräts aus mehreren Metadaten zusammensetzen
$customer.id={%=string().format("%05d", "{%=template().customer().id()%}")%}
$dms.hostname=TEST{%=string().concat("{%=template().customer().meta(\"Kürzel\")%}", "-", "{%=string().format(\"%05d\", \"{%=template().customer().id()%}\")%}", "-", "{%=device().internalnumber()%}\" )%}

# Geräte-Metadaten mit komplexerer Datenstruktur befüllen (Variante 1: Index-Notation)
$dms.meta.Hardware.Type=LWac-G
$dms.meta.Hardware.Flags.LTE.Intfs=1
$dms.meta.Hardware.Flags.WLAN.Intfs=1
$dms.meta.Hardware.Flags.WLAN.Mode=ac
$dms.meta.Hardware.Flags.GPS=true

# Geräte-Metadaten mit komplexerer Datenstruktur befüllen (Variante 2: JSON)
$dms.meta.Hardware={"Type":"LWac-G","Flags":{"LTE":{"Intfs":1},"WLAN":{"Intfs":1,"Mode":"ac"},"GPS":true}}

# Geräte-Metadaten nur setzen wenn sie noch nicht existieren
$dms.meta.Config.LAN.DFI={%=string().default("{%=device().meta(\"Config.LAN.DFI\")%}", "lan5") %}
# verwenden kann man die Option dann auch wieder mit einem Defaultwert z.B. in einem Schnipsel:
network.vlan.0.interface={%=string().default("{%=device().meta(\"Config.LAN.DFI\")%}", "lan5") %}
```

Setzen von Gerätekommunikationsprotokolloptionen

```
# vgl. Kap 2. / "Konfiguration von Protokolloptionen"
$dms.meta.DeviceProtocol.Http.Security.Force=true
$dms.meta.DeviceProtocol.Ssh.Port=2222
```

Setzen von Gerätekommunikationstreiberoptionen

```
# vgl. Kap 2. / "Konfiguration von Treiberoptionen"
$dms.meta.DeviceCommand.Netmodule.Update.Firmware.SshScriptPre=123
$dms.meta.DeviceCommand.Netmodule.Update.Firmware.SshScriptPost=124
$dms.meta.DeviceCommand.Netmodule.Update.Firmware.Timeout=600
```

Generierung eines vordefinierten (statt zufällig erzeugten) Gerätepassworts

```
# Angabe von Benutzername und Passwort mittels hartkodierter Zeichenfolge
$admin.pass={%=credentials().generate("device", "UserNameXYZ", "t0pS3crEt!")%}
```

Generierung eines Public-Private-Schlüsselpaares

```
# Funktion liefert kein Ergebnis, benötigt aber einen Dummy-Optionsnamen zur Ausführung
$dummy={%=credentials().generate("keypair")%}
```

Anlage eines Public-Private-Schlüsselpaares per Kopie

```
# Funktion liefert kein Ergebnis, benötigt aber einen Dummy-Optionsnamen zur Ausführung
# Die Vorgabe von SSH-Keypairs kann nur per Kopie aus einem bestehenden Zugangsdatensatz
# erfolgen, der per ID referenziert wird.
$dummy={%=credentials().generate("keypair", "UserNameXYZ", "{%=credentials(1234)%}")%}
```

Datenformatierung und Vergleiche

```
# Kundenbasierte Ausnahme über Metadaten steuern
$customer_id={%=string().isset("{%=device().meta(\"Ausnahmefall\")}%")%, 42,
"{%=template().customer().id()%}")%}

# Option nur in Konfiguration rendern, wenn ein Metadatum am Template gesetzt ist
wlan.0.channel={%=string().default("{%=template().meta(\"WLAN.Kanal\")}%")%}

# Option mit Standardwert setzen, wenn referenzierte Option unbelegt
wlan.1.channel={%=string().default("{%=option(\"wlan.1.channel\")}%")%, "13") %}

# Einfacher Vergleich von zwei vorab berechneten Optionen
$x=23
$y=42
wlan.0.ssid={%= string().compare("{%=option(\"$x\")}%")%, "<", "{%=option(\"$y\")}%")%,
"Netzname", "Netzname_Gastzugang" %}

# Firewall deaktivieren, wenn IP in Metadaten-Whitelist des Templates enthalten
$dms.ip=10.2.3.4
firewall.status={%=string().compare("{%=option(\"$dms.ip\")}%")%, "in",
"{%=template().meta(\"Firewall.Whitelist\")}%")%, 0, 1)%}

# setzen der Config Version abhängig vom Gerätemodell
config.version={%= string().compare("{%=device().model()%}")%, "in",
"[\"NB2900\", \"NB2910\"]", "1.20", "1.18") %}
```

Verwaltung

Firmware

Unter dem Menüpunkt „Geräte“ -> „Firmware-Versionen“ können die verschiedenen Firmwarestände gepflegt werden. Erst wenn ein Firmware Image hochgeladen wurde, kann die Firmware zur Aktualisierung auf Geräten ausgewählt werden. Die Angabe der URL für die Release Notes ist nicht notwendig.

Passwörter

Die Passwörter werden in einer verschlüsselten Passwortdatenbank gespeichert. Der private Key zur Ver- und Entschlüsselung der Kennwörter wird im Filesystem in der Konfigurationsdatei /common/config/params-local.php unter der Variablen „secret“ abgelegt. Dieser Schlüssel sollte sehr gut gesichert werden, da andernfalls die Datenbank und ein Backup der Datenbank wertlos sind.

SIM-Karten

Health Check

IP Address Management (IPAM)

Im NMS-internen IP Address Management können IP-Netzwerke, Subnetze und IP-Adressen verwaltet werden. Wird eine freie IP-Adresse mittels Skriptausdruck im Configuration Template zur Erzeugung eines Gerätes verwendet, so wird diese IP-Adresse im IPAM als verwendet markiert und der entsprechende Hostname des Gerätes eingetragen.

Administration

Aufgabenverwaltung

Über die Aufgabenverwaltung können zeitgesteuerte Jobs/Aufgaben ausgeführt werden.

Hinweis: Bei vielen Aufgaben können Filter verwendet werden. Bei diesen Filtern kann das String Format aus folgender Dokumentation verwendet werden:

<https://www.yiiframework.com/doc/guide/2.0/en/db-query-builder#where>

Beispiele: `status=1` oder `vehicle_id is not null`

Folgende Jobs stehen zur Verfügung:

Aktion	Beschreibung
Gerätestatusabfrage	Fragt den Status aller im System vorhandenen oder auf den Filter zutreffenden Geräte ab.
Zertifikate prüfen & erneuern	Prüft und erneuert Zertifikate in der Zertifizierungsstelle.
Konfigurationen aktualisieren	
Alte verbundene Geräte löschen	Löscht „verbundene Geräte“ von Geräten älter als die konfigurierte Angabe.
Ereignisse für Objekte auslösen	Löst das gewählte Ereignis auf den Filtern zutreffenden Objekten aus. Kann verwendet werden um Pluginereignisse auszulösen welche auf das gewählte Ereignis reagieren.
Verbundene Geräte abfragen	Aktualisiert die „verbundene Geräte“ Liste von Geräten und fügt neue Einträge hinzu.
Firmware aktualisieren	
Firmware Release Download von einer Herstellerseite	Legt neue Versionen auf Basis einer Webseite an und lädt die entsprechenden Firmware Images herunter.
Alte Historieneinträge bereinigen	Löscht alte Historien Einträge.
Alte Einträge von interface_stats bereinigen	Löscht alte Einträge in der interface_stats Tabelle
Mailbenachrichtigung senden	Sendet E-Mail Benachrichtigungen verschiedener Typen.
Warteschlangenprüfung	
Geräte neustarten	Startet die auf den Filter zutreffenden Geräte zum Ausführungszeitpunkt neu.
Verwaiste Auditeinträge von gelöschten Elementen löschen.	Löscht die Einträge aus dem Auditlog welche zu bereits gelöschten Datensätzen im System gehören. Der Eintrag, welcher den eigentlichen Löschvorgang dokumentiert bleibt erhalten.
Reindizierung Suchindex	Löscht den Index in der externen Suchmaschine (opensearch/elasticsearch) und erstellt diesen für die gewählten Elemente neu. Dieser Vorgang kann je nach Anzahl der Elemente ein paar Minuten dauern.

E-Mail Benachrichtigungen

Aktion	Beschreibung
Nicht erreichbare Geräte	Geräte, welche eine definierbare Zeit Offline sind.
Anzahl von Geräten, die seit Monatsanfang online waren	Anzahl von Geräten, die seit Monatsanfang online waren aufgegliedert nach Gruppen.

Anzahl der seit Monatsanfang neu erstellten Geräte	Anzahl der seit Monatsanfang neu erstellten Geräte aufgegliedert nach Gruppen.
Geräte, die den primären Hotlink nicht verwenden	Auflistung der Geräte, welche den primären Hotlink nicht verwenden
Geräte, deren Module einem Muster entsprechen	Geräte, deren Module einem Muster entsprechen. Damit können veraltete Firmwareversionen für Modems gesucht und reportet werden.
Geräte, deren Konfiguration einem Muster entspricht	Mit diesem Report können Geräte gefunden werden, deren Konfiguration einem entsprechendem Suchmuster entspricht oder nicht enthält.
Geräte, die von der NMS-Serverzeit abweichen	Geräte, die von der NMS-Serverzeit um eine bestimmte Differenz abweichen
Geräte mit nicht aktueller Firmware	Geräte mit veralteter Firmware
Geräte mit SIM-Karten, deren verbrauchtes Datenvolumen grenzwertig ist	Auflistung der Geräte mit SIM-Karten, deren verbrauchtes Datenvolumen grenzwertig ist
Zertifikate, deren Ablaufdatum jünger als ein definiertes Datum ist	bald ablaufende oder bereits abgelaufene Zertifikate
Übersicht über verbrauchtes Datenvolumen pro Gruppe und insgesamt	Übersicht über verbrauchtes Datenvolumen pro Gruppe und insgesamt. Dies dient zur Überwachung des Datenverbrauchs über alle SIM-Karten. (insbesondere beim Pooling)

CLI-Interface

Auf der CLI stehen im Verzeichnis /var/www/nms/ mit dem Befehl yii ein paar Hilfsfunktionen zur Verfügung.

Hilfe und Übersicht aller Kommandos:

```
./yii help
```

Die wichtigsten Kommandos werden in den nächsten Kapiteln aufgelistet

Benutzerverwaltung

Benutzer interaktiv über die CLI hinzufügen.

```
./yii user/add
```

Benutzer über Parameter hinzufügen:

```
./yii user/add --username=test --fullname="Max Mustermann" --email="max.muster@musterag.de" --roles=admin
```

Passwort zurücksetzen:

```
./yii user/password
```

Queue Management

Folgende Queues stehen zur Verfügung:

Name	Funktion
status-jobs	Geräte-Statusabfrage
event-jobs	Jobs welche aus Ereignissen z.B. aus Plugins entstehen. Dies können z.B. API-Requests sein.
device-jobs	Aufgaben, welche auf dem Gerät ausgeführt werden. z.B. Reboot, Firmwareupdates, usw.

action-jobs	Aufgaben aus der Aufgabenverwaltung
-------------	-------------------------------------

Informationen über die Warteschlange anzeigen: (dabei <queue> mit dem Namen der jeweiligen Queue ersetzen)

```
./yii <queue>/info
```

Die Queue leeren:

```
./yii <queue>/clear
```

Migrations

Unter Migrations werden Änderungen an dem Datenbankschema verfolgt.

Hinweis: Diese Befehle wirken destruktiv, d.h. bei falscher Verwendung, kann Datenverlust entstehen. Immer ein Datenbank Backup vor Verwendung erstellen.

Historie von Schema Änderungen anzeigen:

```
./yii migrate/history
```

Änderung rückgängig machen:

```
./yii migrate/down
```

Events

Events ermöglichen es, benutzerdefinierten Code zu bestimmten Ausführungszeitpunkten in bestehenden Code einzubauen. Es ist möglich benutzerdefinierten Code an ein Event anzuhängen, so dass der Code automatisch ausgeführt wird, wenn das Ereignis ausgelöst wird.

Folgende Events werden aktuell verwendet:

Event	Funktion
EVENT_AFTER_CHILD_ADD	Wird ausgeführt, nachdem eine Gui-Komponente zu einer anderen hinzugefügt wurde
EVENT_AFTER_CHILD_REMOVE	Wird ausgeführt, nachdem eine Gui-Komponente von einer anderen entfernt wurde
EVENT_AFTER_COMMAND_RUN	Wird ausgeführt, nachdem ein Befehl auf einem Gerät ausgeführt wurde
EVENT_AFTER_CONFIG_QUERYDEVICE	Wird ausgeführt, nachdem die relevanten Geräte vom Downloadcontroller gefunden wurden
EVENT_AFTER_CONFIG_RENDER	Wird ausgeführt, nachdem eine Konfiguration generiert wurde
EVENT_AFTER_CONFIG_TAR	Wird ausgeführt, nachdem eine Netmodule Konfiguration in ein TAR-Archiv gepackt wurde
EVENT_AFTER_CONFIG_TRANSFER	Wird ausgeführt, nachdem eine Konfiguration an ein Gerät übertragen wurde
EVENT_AFTER_CONFIG_UPDATE	Wird ausgeführt, nachdem eine Konfiguration von einer Vorlage aktualisiert wurde
EVENT_AFTER_CONFIG_VALIDATEDDEVICE	Wird ausgeführt, nachdem die relevanten Geräte vom Downloadcontroller validiert wurden
EVENT_AFTER_CONFIG_ZIP	Wird ausgeführt, nachdem eine Netmodule Konfiguration in ein ZIP-Archiv gepackt wurde
EVENT_AFTER_DEVICE_CREATE	Wird ausgeführt, nachdem ein Gerät durch eine Vorlage generiert wurde
EVENT_AFTER_DEVICE_REPLACEMENT	Wird ausgeführt, nachdem ein Gerät in einem Einbauobjekt ausgetauscht wurde
EVENT_AFTER_DEVICES_GENERATE	Wird ausgeführt, nachdem alle angegebenen Geräte durch eine Vorlage generiert wurden
EVENT_AFTER_NEIGHBORHOOD_QUERY	Wird ausgeführt, nachdem die Nachbargeräte abgefragt wurden.
EVENT_AFTER_RENDER	Wird ausgeführt, nachdem eine Gui Komponente angezeigt wurde
EVENT_AFTER_SCHEDULE_PUSH	Wird ausgeführt, nachdem eine Aufgabe gestartet wurde.
EVENT_AFTER_SEND	Wird ausgeführt, nachdem ein HTTP Request gesendet wurde
EVENT_AFTER_STATUS_QUERY	Wird ausgeführt, nachdem der Status eines Gerätes abgefragt wurde

EVENT_AFTER_STATUS_UPDATE	Wird ausgeführt, nachdem der Status eines Gerätes aktualisiert wurde
EVENT_BEFORE_CHILD_ADD	Wird ausgeführt, bevor eine Gui-Komponente zu einer anderen hinzugefügt wird
EVENT_BEFORE_CHILD_REMOVE	Wird ausgeführt, bevor eine Gui-Komponente von einer anderen entfernt wird
EVENT_BEFORE_COMPONENTS_RENDER	Wird ausgeführt, bevor die Gui-Komponenten angezeigt werden
EVENT_BEFORE_CONFIG_QUERYDEVICE	Wird ausgeführt, bevor die relevanten Geräte vom Downloadcontroller gesucht werden
EVENT_BEFORE_CONFIG_RENDER	Wird ausgeführt, bevor eine Konfiguration generiert wird
EVENT_BEFORE_CONFIG_TAR	Wird ausgeführt, bevor eine Netmodule Konfiguration in ein ZIP-Archiv gepackt wird
EVENT_BEFORE_CONFIG_TRANSFER	Wird ausgeführt, bevor eine Netmodule Konfiguration in ein ZIP-Archiv gepackt wird
EVENT_BEFORE_CONFIG_VALIDATEDDEVICE	Wird ausgeführt, bevor die relevanten Geräte vom Downloadcontroller validiert werden
EVENT_BEFORE_CONFIG_ZIP	Wird ausgeführt, bevor die Konfiguration in ein ZIP-Archiv gepackt wird
EVENT_BEFORE_DEVICE_CREATE	Wird ausgeführt, bevor ein Gerät durch eine Vorlage generiert wird
EVENT_BEFORE_DEVICE_REPLACEMENT	Wird ausgeführt, bevor ein Gerät in einem Einbauobjekt ausgetauscht wird
EVENT_BEFORE_DEVICES_GENERATE	Wird ausgeführt, bevor alle angegebenen Geräte durch eine Vorlage generiert werden
EVENT_BEFORE_NEIGHBORHOOD_QUERY	Wird ausgeführt, bevor die Nachbargeräte abgefragt werden
EVENT_BEFORE_RENDER	Wird ausgeführt, bevor eine Gui Komponente angezeigt wird
EVENT_BEFORE_SCHEDULE_PUSH	Wird ausgeführt, bevor Schedule gestartet wird
EVENT_BEFORE_SEND	Wird ausgeführt, bevor ein HTTP Request gesendet wird
EVENT_BEFORE_STATUS_QUERY	Wird ausgeführt, bevor der Status eines Gerätes abgefragt wird
EVENT_BEFORE_STATUS_UPDATE	Wird ausgeführt, bevor der Status eines Gerätes aktualisiert wird
EVENT_CERTIFICATE_RENEW	Wird ausgeführt, wenn ein Zertifikat erneuert wird
EVENT_DEVICE_STATUS_UPDATED	Wird ausgeführt, wenn sich der Status eines Gerätes ändert.
EVENT_ON_UNHANDLED_CONFIGURABLE_PROP	Wird ausgeführt, wenn beim Generieren einer Konfiguration eine Unbekannte Konfigurationsoption entdeckt wurde.

Anlagen

ER-Diagramm / Datenbankschema

