Bedienungsanleitung

Python-Interface für MultiAnalyzer QoS

Deutsch

Inhaltsverzeichnis

Impressum

Informationen gemäß §5 TMG und §2 DL-InfoV

femvenner GmbH
Lise-Meitner-Str. 2
24941 Flensburg

Kontakt

Telefon: +49 461 16839627
E-Mail: webcontact@femvenner.de
Webseite: http://www.femvenner.de

Geschäftsführer

Gunter Hinrichsen
Steffen Zscherneck
Matthias Jahr

Registereintrag

Eintrag im Handelsregister
Registernummer: HRB 10643
Registergericht: Amtsgericht Flensburg

Umsatzsteuer-Identifikationsnummer

Umsatzsteuer-Identifikationsnummer gemäß §27a
Umsatzsteuergesetz: **DE296134379**

Haftungsausschluss

Haftung für Inhalte

Die Inhalte unserer Seiten wurden mit größter Sorgfalt erstellt. Für die Richtigkeit, Vollständigkeit und Aktualität der Inhalte können wir jedoch keine Gewähr übernehmen. Als Dienstleister sind wir gemäß §7 Abs. 1 TMG für eigene Inhalte auf diesen Seiten nach den allgemeinen Gesetzen verantwortlich. Nach §§8 bis 10 TMG sind wir als Dienstleister jedoch nicht verpflichtet, übermittelte oder gespeicherte fremde Informationen zu überwachen oder nach Umständen zu forschen, die auf eine rechtswidrige Tätigkeit hinweisen. Verpflichtungen zur Entfernung oder Sperrung der Nutzung von Informationen nach den allgemeinen Gesetzen bleiben hiervon unberührt. Eine diesbezügliche Haftung ist jedoch erst ab dem Zeitpunkt der Kenntnis einer konkreten Rechtsverletzung möglich. Im Falle eines Rechtsbruchs werden wir diese Inhalte umgehend entfernen.

Copyright

Version 1.2.0 (23/01/2019)

Technische Änderungen vorbehalten.

1 Python Interface für MultiAnalyzer QoS

Das Python Interface erlaubt es dem Nutzer die Daten aus den Aufzeichnungen im MultiAnlyzer QoS (MAQoS) mittels Pythonscripte zu extrahieren, analysieren und gegebenenfalls die Darstellung der Daten im MAQoS anzupassen. Die Scripte können sowohl kontinuierlich im Hintergrund als auch durch den Benutzer selbst ausgelöst werden.

Dies erlaubt es Benutzern auch spezielle Probleme zu isolieren und die Darstellung individuell anzupassen.

1.1 Menüs

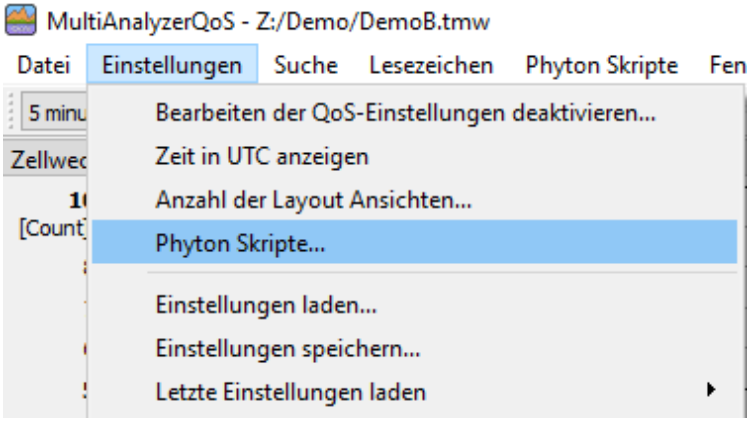

Abbildung 1: Menüoption - Einstellungen Python Skripte

Die Konfiguration der Python Schnittstelle findet sich im MAQoS Menu unter Einstellungen.

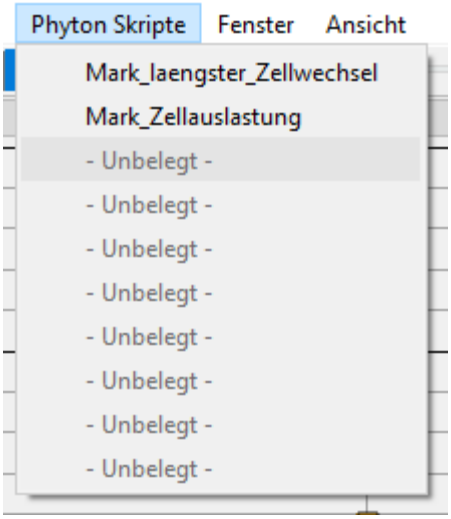

Abbildung 2: Menüoption - Python Skripte

Das Menu „Python Skripte“ enthält durch den Benutzer hinzugefügte Skripte (siehe Einstellungen), welche nicht kontinuierlich ausgeführt werden.

1.2 Einstellungen

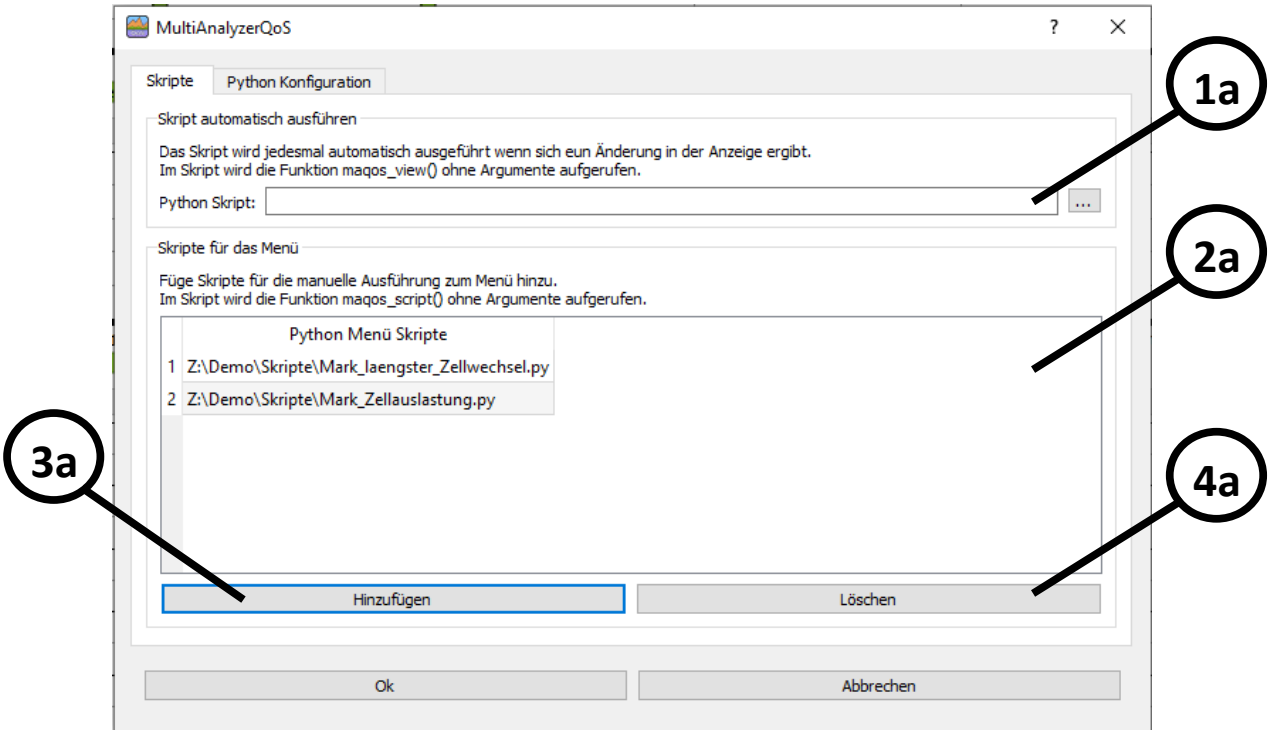

Abbildung 3: Einstellungen Skripte

Nr	Beschreibung
1a	Eingabefeld für automatisch ausgeführtes Skript. Das automatische Skript wird jedes Mal ausgeführt, wenn sich eine Änderung in der Anzeige ergibt. Dieses Skript kann nicht aus dem Menu heraus ausgeführt werden.
2a	Anzeige der im Python Menu hinterlegten Skripte. Diese werden zum Menu „Python Skripte“ hinzugefügt und sind so aus der MAQosS Benutzeroberfläche direkt aufrufbar.
3a	Hinzufügen eines Python Skriptes zum Menu „Python Skripte“
4a	Entfernen des in Anzeige 2 ausgewählten Skriptes aus dem Menu „Python Skripte“

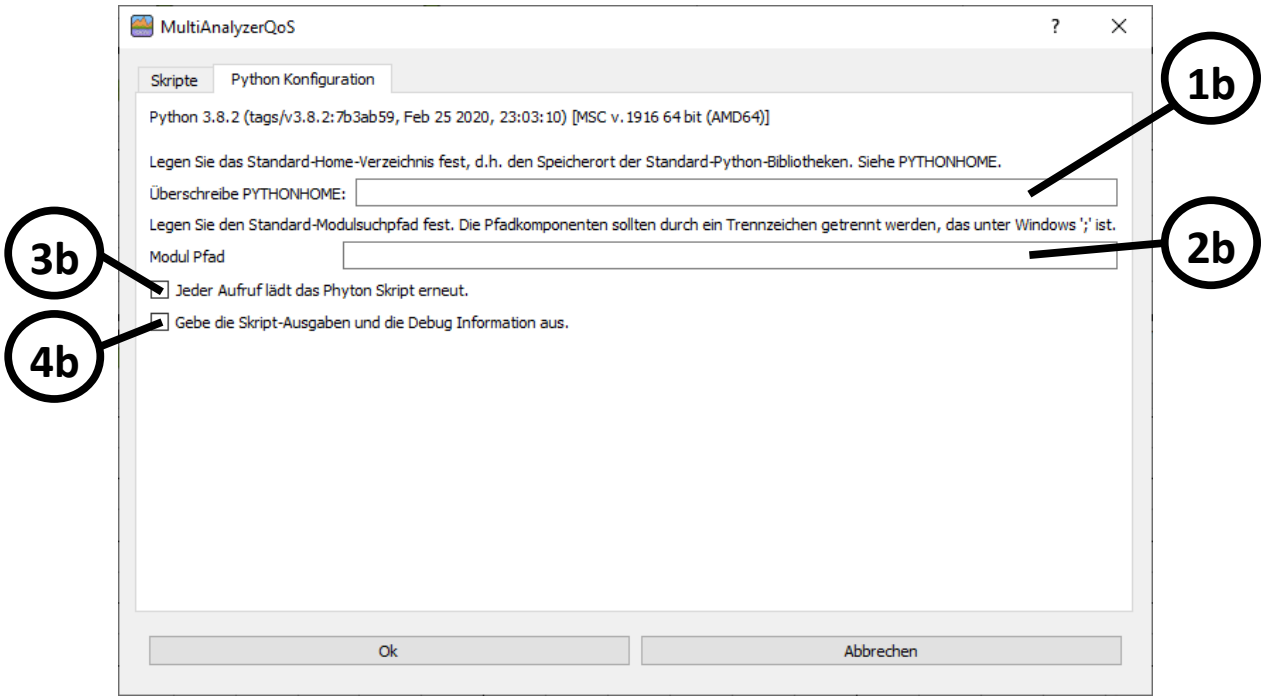

Abbildung 4: Einstellungen - Python Konfiguration

Nr	Beschreibung
1b	Eingabefeld für das Standard Python Verzeichnis. Der hier eingegebene Pfad wird dem systemweit gesetzten Pfad bevorzugt, um Python-Bibliotheken und Programme zu finden
2b	Eingabefeld für das Verzeichnis mit Standard Pythonmodulen
3b	Ist der Haken gesetzt, werden die Python Skripte neu geladen bevor sie ausgeführt werden. Standard Python Verhalten ist, dass ein Skript nur einmal geladen wird und dann gecached wird. Während der Entwicklung eines Skripts kann diese Option aktiviert werden, um neue Änderungen sofort wirksam zu machen
4b	Ist der Haken gesetzt werden die Skript und Debug ausgaben zum MAQoS durchgereicht und angezeigt. Für die Entwicklung von Skripten nützlich.

2 Skripte

Die Python-Schnittstelle unterscheidet zwischen zwei verschiedenen Arten von Skripten. Zum einen Skripte die automatisiert ausgeführt werden, sobald sich die Anzeige verändert. Zum anderen Skripte die direkt vom Benutzer ausgeführt werden.

Beide Arten von Skripten können alle API-Funktionen nutzen, sie unterscheiden sich nur durch die Funktion durch die sie vom MAQoS aufgerufen werden.

In automatischen Skripte wird die Funktion „maqos_view()“ aufgerufen, in benutzergestarteten Skripten die Funktion „maqos_script()“. Beide Funktionen werden ohne Argumente aufgerufen.Es ist auch möglich in einem Skript beide Einsprungfunktionen zu definieren, um das Skript sowohl automatisiert als auch benutzegesteuert nutzen zu können.

Die Skripte müssen das Modul maqos importieren, welches automatisch vom MAQoS übergeben wird sobald das Skript aufgerufen wird. Es ist keine Installation erforderlich.

2.1 Minimalbeispiele

Minimales Beispiel mit Aufruf einer Funktion – automatisiertes Skript:

```
#--------- ShowLayout1.py -----------
# Das Skript stellt sicher, dass Layout 1 aktiviert ist
import maqos   #importiert die API Funktionen

def maqos_view():
    rc, currentLayout = maqos.getLayout()
    if rc == 0 and currentLayout != 0:
        maqos.setLayout(0)

    return r
```

Minimales Beispiel mit Aufruf einer Funktion – benutzergestartetes Skript:

```
#--------- ShowLayout1_.py -----------
# Das Skript stellt sicher, dass Layout 1 aktiviert ist
import maqos   #importiert die API Funktionen

def maqos_script():
    rc, currentLayout = maqos.getLayout()
    if rc == 0 and currentLayout != 0:
        maqos.setLayout(0)

    return rc
```

2.2 Printausgabe in Debugwindow

Sollen Ausgaben der Python print()-Funktion über die Debugschnittstelle (Häkchen bei 4b, Abbildung 4) ausgegeben werden, kann dies durch folgenden Code im Skript ermöglicht werden:

```
#--------------------
# routet print() Ausgabe an MAQoS Debug Window weiter
import sys

class catchOutErr_:
    def __init__(self):
        self.value = ‘‘
    def write(self, text):
        self.value += text

catchOutErr = catchOutErr_()
sys.stdout  = catchOutErr
sys.stderr  = catchOutErr
```

3 Python-API

Dieses Kapitel beschreibt die zur Verfügung stehenden Python Funktionen, mit denen Daten und Anzeigen des MAQoS ausgelesen und angepasst werden können.

3.1 Definitionen

Die folgenden Variablen aus dem MAQoS Fenster werden für den Zugriff auf bestimmte Funktionen benötigt. Sie sind, bis auf den WindowName, auch über die Pythonschnittstelle auslesbar, können aber auch direkt genutzt werden.

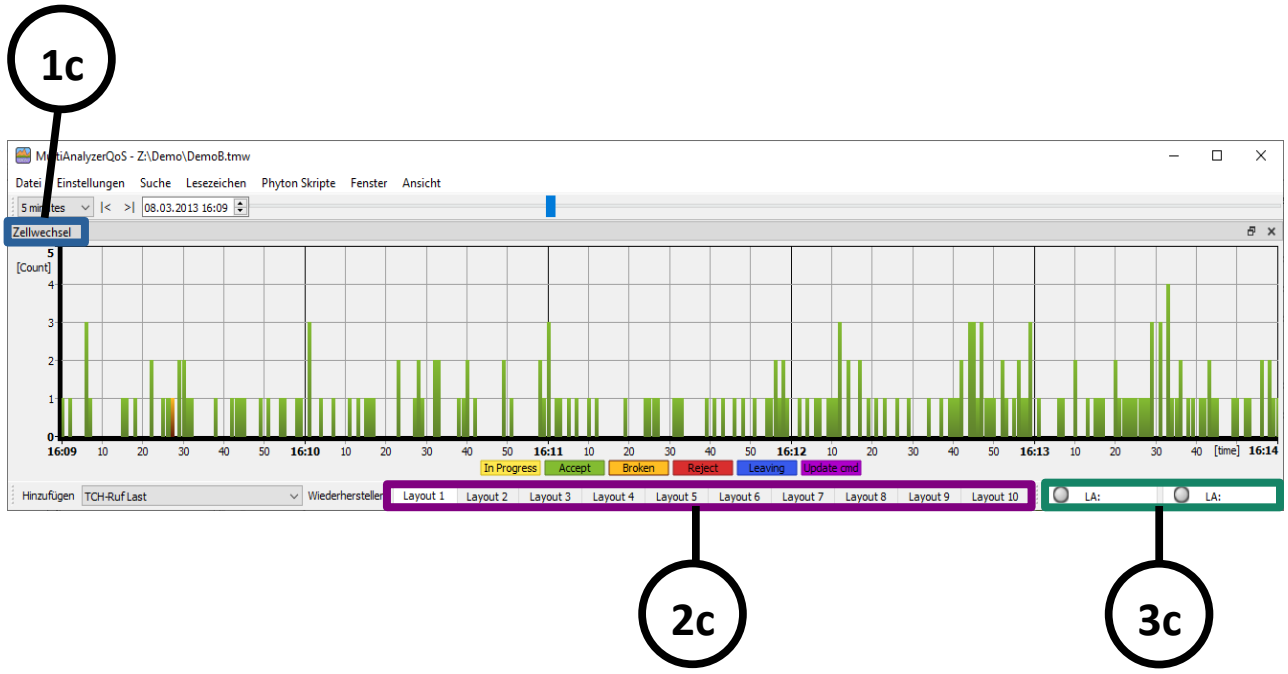

Abbildung 5: WindowName, Layout, CellIndex

Nr	Beschreibung
1c	WindowName
2c	Layout – Die Layouts im MAQoS sind von 1 .. max 10 benannt. Die Anzahl ist im MAQoS konfigurierbar. Siehe auch: getLayout( ) / setLayout(Layout)
3c	CellIndex – wird von links nach rechts ab 0 nummeriert. Sollte bei mehreren Zellen durch Abfrage verifiziert werden. Siehe auch: getCellList( )

3.2 Funktionen - Fenster und Layout

3.2.1 setLayout(Layout)

Setzt das angezeigte Layout.

Argumente	
Layout	Nummer des Layout. Anzahl ist konfigurierbar im MaQoS, max = 10 1 - max
Rückgabe - (*RC*)	
RC	0: *OK* 8: *Fehler*: Anzahl der Variablen ist falsch 10: *Fehler*: Layout unbekannt

→getLayout( )

3.2.2 getLayout( )

Gibt das angezeigte Layout zurück.

Argumente	
keine	
Rückgabe – (*RC, Layout*)	
RC	0: *OK* 8: *Fehler*: Anzahl der Variablen ist falsch
Layout	Nummer des Layout. Anzahl ist konfigurierbar im MaQoS, max = 10 1-max

→ setLayout(Layout)

3.2.3 setWindowStatus(Layout, WindowName, WindowStatus)

Setzt Status eines Fensters

Argumente	
Layout	Nummer des Layout
WindowName	Name des Fensters
WindowStatus	0: Fenster nicht sichtbar (window not visible) 1: Fenster integriert sichtbar (window docked visible) 2: Fenster herausgelöst sichtbar (window floating visible)
Rückgabe - (*RC*)	

RC	0: *OK* 2: *Fehler*: Fenster unbekannt 8: *Fehler*: Anzahl der Variablen ist falsch 10: *Fehler*: Layout unbekannt

→ getWindowStatus(Layout, WindowName)

3.2.4 getWindowStatus(Layout, WindowName)

Gibt den Status eines Fensters zurück

Argumente	
Layout	Nummer des Layout
WindowName	Name des Fensters
Rückgabe – (*RC, WindowStatus*)	
RC	0: OK 2: Fehler: Fenster unbekannt 8: Fehler: Anzahl der Variablen ist falsch 10: Fehler: Layout unbekannt
WindowStatus	-1: Fenster nicht bekannt (unknown) 0: Fenster nicht sichtbar (window not visible) 1: Fenster integriert sichtbar (window docked visible) 2: Fenster herausgelöst sichtbar (window floating visible)

→ setWindowStatus(Layout, WindowName, WindowStatus)

3.2.5 setWindowBk(CellIndex, WindowName, Color1, Color2)

Setzt oder löscht die Hintergrundfarbe eines Fensters

Argumente	
CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
Color1	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : Farbe entfernen
Color2	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : keine zweite Farbe
Rückgabe – (*RC*)	

RC	0: *OK* 2: *Fehler*: Fenster unbekannt 3: *Fehler*: Farbe unbekannt 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt

->getWindowBk(CellIndex, WindowName)

3.2.6 getWindowBk(CellIndex, WindowName)

Gibt die Hintergrundfarbe eines Fensters zurück

Argumente	
CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
Rückgabe – (*RC, Color1, Color2*)	
RC	0: *OK* 2: *Fehler*: Fenster unbekannt 8: *Fehler*: Anzahl der Variablen ist falsch 9: *Fehler*: Funktion wird für das Fenster nicht unterstützt 11: *Fehler:* Zellindex unbekannt
Color1	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : keine Farbe gesetzt
Color2	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : keine Farbe gesetzt

→ setWindowBk(CellIndex, WindowName, Color1, Color2)

3.2.7 setWindowBorder(CellIndex, WindowName, Color1, Color2)

Setzt oder löscht die Rahmenfarbe eines Fensters

Argumente	
CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
Color1	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : Farbe entfernen

Color2	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : keine zweite Farbe
Rückgabe – (*RC*)	
RC	0: *OK* 2: *Fehler*: Fenster unbekannt 3: *Fehler*: Farbe unbekannt 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt

->getWindowBorder(CellIndex, WindowName)

3.2.8 getWindowBorder(CellIndex, WindowName)

Gibt die Rahmenfarbe eines Fensters zurück

Argumente	
CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
Rückgabe – (*RC, Color1, Color2*)	
RC	0: *OK* 2: *Fehler*: Fenster unbekannt 8: *Fehler*: Anzahl der Variablen ist falsch 9: *Fehler*: Funktion wird für das Fenster nicht unterstützt 11: *Fehler:* Zellindex unbekannt
Color1	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : keine Farbe gesetzt
Color2	Farbe 0 … 16777215 (0x00**000000** – 0x00**FFFFFF**) : 24Bit RGB-Wert 4294967295 (0x**FFFFFFFF**) : keine Farbe gesetzt

→ setWindowBorder(CellIndex, WindowName, Color1, Color2)

3.2.9 setWindowMarker(CellIndex, WindowName, Marker, Time, Text)

Setzt oder löscht einen Marker mit Text. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Zum Löschen eines Markers den Text leer übergeben.

Argumente

CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
Marker	1 .. 17 (0 kann nur lesen)
Time	time_t für den Marker
Text	Text: setzt Markertext **leer : löschen des Markers**
Rückgabe – (*RC*)	
RC	0: *OK* *6: Warnung : Text zu lang, wird verkürzt dargestellt* 2: *Fehler*: Fenster unbekannt 7: *Fehler*: Zeit außerhalb des Aufnahmebereichs 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt

->getWindowMarker(CellIndex, WindowName, Marker)

3.2.10 getWindowMarker(CellIndex, WindowName, Marker)

Gibt einen Marker mit Text zurück. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Argumente	
CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
Marker	0 : GUI – Marker 1..17 : Python Marker (benutzedefiniert)
Rückgabe – (*RC, Time, Text*)	
RC	0: *OK* *6: Warnung : Text zu lang, wird verkürzt dargestellt* 2: *Fehler*: Fenster unbekannt 8: *Fehler*: Anzahl der Variablen ist falsch 9: *Fehler*: Funktion wird für das Fenster nicht unterstützt 11: *Fehler*: Zellindex unbekannt
Time	time_t für den Marker 0 : kein Marker vorhanden
Text	Markertext leer : kein Marker vorhanden

→ setWindowMarker(CellIndex, WindowName, Marker, Time, Text)

3.2.11 setWindowMarkedArea(CellIndex, WindowName, ColorBg1, ColorBg2 , Opacity, ColorText, LegendText, StartTimes, EndTimes)

Markiert oder löscht einen Bereich. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Mit *FarbeStift* und *FarbeFüllung* bilden einen Gradienten für die Markierung. *FarbeStift* gibt den oberen Farbwert, *FarbeFüllung* den unteren Farbwert des Gradienten.

Zum löschen eines Bereichs *LegendeText* leer übergeben.

Es können bis zu 32 Teilbereiche gesetzt werden. Dafür müssen die jeweiligen Start- und Endzeitpunkte mit gleichen Index in StartTimeListe und EndZeitListe übergeben werden.

Argumente	
CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
ColorBg1	Farbe des Legendenhintergrundes und des oberen Bereichs der Markierung 0 … 16777215 (0x**000000** – 0x**FFFFFF**) : 24Bit RGB-Wert
ColorBg2	Farbe des unteren Teil der Markierung 0 … 16777215 (0x**000000** – 0x**FFFFFF**) : 24Bit RGB-Wert
Opacity	Transparenz des Hintergrundes 0: aus 1 .. 99: transparent … deckend
ColorText	Farbe des Legendentextes 0 … 16777215 (0x**000000** – 0x**FFFFFF**) : 24Bit RGB-Wert
LegendText	Text der Legenden **leer : Löschen des Bereichs**
StartTimes	Startzeiten (time_t) von bis zu 32 Teilbereichen
EndTimes	Endzeiten (time_t) von bis zu 32 Teilbereichen
Rückgabe – (*RC*)	
RC	0: *OK* 2: *Fehler*: Fenster unbekannt 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 5: Fehler: Endzeit außerhalb des Aufnahmebereichs 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt

→ getWindowMarkedArea(CellIndex, WindowName)

3.2.12 getWindowMarkedArea(CellIndex, WindowName)

Gibt einen markierten Bereich zurück. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Argumente	
CellIndex	Index der angezeigten Zelle
WindowName	Name des Fensters
Rückgabe – (*RC, StartTimes, EndTimes*)	
RC	0: *OK* *1: Warnung : keine MarkedArea gesetzt* 2: *Fehler*: Fenster unbekannt 8: *Fehler*: Anzahl der Variablen ist falsch 9: Fehler: Funktion wird für das Fenster nicht unterstützt 11: *Fehler:* Zellindex unbekannt
StartTimes	Startzeiten (time_t) von bis zu 32 Teilbereichen
EndTimes	Endzeiten (time_t) von bis zu 32 Teilbereichen

→ setWindowMarkedArea(CellIndex, WindowName, ColorBg1, ColorBg2 , Opacity, ColorText, LegendText, StartTimes, EndTimes)

3.2.13 setStartViewTime(CellIndex, StartTime)

Setzt den Start-Zeitpunkt der Anzeige. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t für den Start der Anzeige -1 : automatische abspielen (Online Stream)
Rückgabe – (*RC*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt

→ getStartViewTime(CellIndex)
→ getRunStatus( )

3.2.14 getStartViewTime(CellIndex)

Gibt den Startzeitpunkt der Anzeige zurück. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Argumente	
CellIndex	Index der angezeigten Zelle

Rückgabe – (*RC, StartTime*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt
StartTime	time_t für den Start der Anzeige -1 : automatische abspielen (Online Stream)

→ setStartViewTime(CellIndex, StartTime)

→ getRunStatus( )

3.2.15 setViewRange(Range)

Setzt den Start-Zeitpunkt der Anzeige. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Argumente	
Range	Zeitbereich für die Anzeige *mögliche Werte: 60, 12, 300, 600, 900, 1200, 1800, 3600 [sec]*
Rückgabe – (*RC*)	
RC	0: *OK* 7: *Fehler*: Zeitbereich nicht gültig 8: *Fehler*: Anzahl der Variablen ist falsch

→ getViewRange( )

3.2.16 getViewRange( )

Setzt den Start-Zeitpunkt der Anzeige. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Argumente	
keine	
Rückgabe – (*RC, Range*)	
RC	0: *OK* 7: *Fehler*: Zeitbereich nicht gültig 8: *Fehler*: Anzahl der Variablen ist falsch
Range	Zeitbereich für die Anzeige *mögliche Werte: 60, 12, 300, 600, 900, 1200, 1800, 3600 [sec]*

→ setViewRange(Range)

3.2.17 getTchSlots( )

Setzt den Start-Zeitpunkt der Anzeige. *Nur möglich bei Fenstern mit einer Zeit auf der X-Achse*

Argumente	
keine	
Rückgabe – (*RC, Slots*)	
RC	0: *OK*
Slots	2 .. 64

3.3 Funktionen - Aufnahmedaten

3.3.1 getRecordName(CellIndex)

Gibt den aktuell verwendeten Aufnahmenamen (MAF-File Name oder Streaming Adresse) zurück.

Argumente	
CellIndex	Index der angezeigten Zelle
Rückgabe – (*RC, Name*)	
RC	0: *OK* 11: *Fehler:* Zellindex unbekannt
Name	*Leer* : keine Daten geladen *Dateiname* : MAF-Datei inklusive Pfad *Streamadresse*: Online (bsp.: "MUC:239.0.0.1:4000,BIND:10.0.128.106")

3.3.2 getRecordTime(CellIndex)

Gibt die Start- und Endzeit der aktuellen Aufnahmenamen zurück.

Argumente	
CellIndex	Index der angezeigten Zelle
Rückgabe – (*RC, StartTime, EndTime*)	
RC	0: *OK* 11: *Fehler*: Zellindex unbekannt
StartTime	*-1* : keine Daten geladen *time_t* : Startzeit der Daten (time_t)
EndTime	*-1* : keine Daten geladen *time_t* : Endzeit der Daten (time_t)

3.3.3 getRunStatus( )

Gibt den Online Status der aktuellen Aufnahme zurück.

Argumente	
keine	
Rückgabe – (*RC, Status*)	
RC	0: *OK* 8: *Fehler*: Anzahl der Variablen ist falsch
Status	*0:* keine Datei geladen und nicht im Online Modus 1: Offline Modus: Datei vollständig geladen 2: Offline Modus: Datei wird gerade geladen 8: Online Modus: Anzeige wird NICHT automatisch weitergestellt 9: Online Modus: Anzeige wird automatisch weitergestellt

→ getStartViewTime(CellIndex)

3.3.4 getCellList( )

Gibt eine Liste mit allen Zellen zurück

Argumente	
keine	
Rückgabe – (*RC, Cells*)	
RC	0: *OK* 8: *Fehler*: Anzahl der Variablen ist falsch
Cells	Liste mit allen Zellen der Aufnahme. Jeder Eintrag besteht aus dem Tripel (*CellIndex, Identifizierung, Name*) *CellIndex* : 0-n, Zellindex zum Aufrauf von Funktionen *Identifizierung*: TETRA:LA, TMO:GW, RP, SSI, DMR:Network, CC *Name* : Name der Zelle als Text

3.4 Funktionen - Messdaten

3.4.1 getTchCallLoad(CellIndex, StartTime, Range, TypeFlag)

Gibt eine Liste mit allen Zellen zurück

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)

TypeFlag	Typ des Rückgabewertes, siehe auch Rückgabe 0 : Gesamt 1 : Nach Typen aufgeteilt
Rückgabe TypeFlag 0 - (*RC, List*)	
RC	0: *OK* 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt
List	Liste mit *Range* vielen Werten. Jeweils gesamtZahl aller Calls zum Zeitpunkt
Rückgabe TypeFlag 1 - (*RC, P2PList, P2MPList, P2MPjoinedList, P2MPuserList*)	
RC	0: *OK* 8: *Fehler*: Anzahl der Variablen ist falsch 11: *Fehler:* Zellindex unbekannt
List	Listen mit *Range* vielen Werten. Jeweils Gesamtzahl der Calls zum Zeitpunkt aufgschlüsselt in P2P, P2MP, P2MP joined, P2MP user

3.4.2 getTchCallUsage(CellIndex, StartTime, Range)

Gibt Daten für Ruf-Nutzung zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
Rückgabe – (*RC, CallNames, CallTimes*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
CallNames	SSI/Name - Liste mit max 11 Einträgen. [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge
CallTimes	% of 100 der Rufzeitnutzung – Liste mit max. 11 Einträgen [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge

3.4.3 getTchCallCapacity(CellIndex, StartTime, Range)

Gibt Daten für Ruf-Kapazität zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
Rückgabe – (*RC, Callnames, CallTimes, CallPercentages*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
CallNames	*SSI/Name* - Liste mit max 11 Einträgen. [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge
CallTimes	*Rufzeit in Sekunden* – Liste mit max. 11 Einträgen [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge
CallPercentages	*% of 100 Kapazität* – Liste mit max. 11 Einträgen [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge

3.4.4 getTchSlotCapacity(CellIndex, StartTime, Range)

Gibt Daten für Slot-Kapazität zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
Rückgabe – (*RC, FreeSlots, P2PSlots, P2MPSlots, P2MPjoinedSlots, UserDefCalls*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
FreeSlots	Prozentualer Anteil von freien Slots
P2PSlots	Prozentualer Anteil von P2P Slots
P2MPSlots	Prozentualer Anteil von P2MP Slots
P2MPjoinedSlots	Prozentualer Anteil von P2MP joined Slots
UserDefCalls	Prozentualer Anteil von benutzedefinierten Calls

3.4.5 getTchErlangC(CellIndex, StartTime, Range, TypeFlag)

Gibt Daten für ErlangC zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
TypeFlag	0 : Rufzeit aus Daten 1 : Rufzeit aus Einstellungen
Rückgabe – (*RC, Pw, Calls, CallTime, OverallCallTime*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
Pw	Wahrscheinlichkeit in Prozent
Calls	Anzahl der Rufe
CallTime	Rufzeit TypeFlag = 0 : Durchschnitt TypeFlag = 1 : Benutzedefiniert
OverallCallTime	Gesamtrufzeit

3.4.6 getCcchLoad(CellIndex, StartTime, Range, TypeFlag)

Gibt Daten für CCCH-Last zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
TypeFlag	0-3 : DL MCCH, SCCH 1-3 4-7 : UL MCCH, SCCH 1-3 + 0x00 : Aufgeteilt nach Gruppen (%) + 0x80 : über Gesamtlast (%)
Rückgabe – TypeFlag + 0x00 (*RC, LoadList)*	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
LoadList	Liste mit Lastwerten in Prozent. Liste enthält *Range* viele Werte

Rückgabe – TypeFlag + 0x80 (RC, LoadName, LoadList)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs
LoadName	Liste mit Gruppennamen, Lange wird als Index für *LoadList* benötigt
LoadList	Liste mit Lastwerten in Prozent. Die Liste ist zweidimensional. LoadList[*LoadNameLength*][*Range*]

3.4.7 getCcchUsage(CellIndex, StartTime, Range, TypeFlag)

Gibt Daten für CCCH-Nutzung zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
TypeFlag	0-3 : DL MCCH, SCCH 1-3 4-7 : UL MCCH, SCCH 1-3
Rückgabe (*RC, GroupNameList, GroupPercentageList)*	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
GroupNameList	Gruppennamenliste - Liste mit max 11 Einträgen. [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge
GroupPercentageList	Ccch Nutzung in Prozent - Liste mit max 11 Einträgen. [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge

3.4.8 getCcchCapacity(CellIndex, StartTime, Range, TypeFlag)

Gibt Daten für CCCH-Nutzung zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
TypeFlag	0-3 : DL MCCH, SCCH 1-3 4-7 : UL MCCH, SCCH 1-3

Rückgabe (*RC, GroupNames, GroupPercentages)*	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
GroupNames	Gruppennamenliste - Liste mit max 11 Einträgen. [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge
GroupPercentages	Ccch Nutzung in Prozent - Liste mit max 11 Einträgen. [1 .. 10] : Liste der 10 größten Einträge [11] : zusammengefasster Wert aller anderen Einträge

3.4.9 getTchMessageLoad(CellIndex, StartTime, Range, TypeFlag)

Gibt Daten für Tch-Signalisierungs-Last über die Zeit zurück

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
TypeFlag	0-3 : DL TCH Slot 1-4 4-7 : UL SCH Slot 1-4
Rückgabe (*RC, LoadNames, LoadList)*	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
LoadNames	Liste mit Gruppennamen, Länge wird als Index für *LoadList* benötigt
LoadList	Liste mit TCH-Signalisierungslast. Die Liste ist zweidimensional. LoadList[*LoadNamesLength*][*Range*]

3.4.10 getCellChangeLoad(CellIndex, StartTime, Range)

Gibt Daten für Zellwechsel-Signalisierungs-Last über die Zeit zurück

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)

Rückgabe (*RC, LoadNames, LoadList)*	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
LoadNames	Liste mit Gruppennamen, Länge wird als Index für *LoadList* benötigt
LoadList	Liste mit TCH-Signalisierungslast in Prozent. Die Liste ist zweidimensional. LoadList[*LoadNamesLength*][*Range*]

3.4.11 getCellChangeTimeHistogram(CellIndex, StartTime, Range)

Gibt Daten für ein Zellwechsel-Histogram zurück. Klassen nach Zeit.

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
Rückgabe (*RC, LoadNames, LoadList)*	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
LoadNames	Liste mit Gruppennamen, Länge wird als Index für *LoadList* benötigt
LoadList	Liste ZellwechselDaten als Zahl. Die Liste ist zweidimensional. LoadList[*CellChangeResult*][*BinSize*] *CellChangeResult* : 0=Name, 1=Call restoration ok, 2=Call restoration failed BinSize : 0=unknown, 1='<500', 2='<1000', 3='<1500', 4='<2000', 5='>2000

3.4.12 getCellChangeCallHistogram(CellIndex, StartTime, Range)

Gibt Daten für ein Zellwechsel-Histogram zurück. Klassen nach Rufen.

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
Rückgabe (*RC, LoadNames, LoadList)*	
RC	0: *OK*

	4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
LoadNames	Liste mit Gruppennamen, Länge wird als Index für *LoadList* benötigt
LoadList	Liste ZellwechselDaten als Zahl. Die Liste ist zweidimensional. LoadList[*CellChangeResult*][*BinSize*] *CellChangeResult* : 0=Name, 1=Call restoration ok, 2=Call restoration failed BinSize : 0=unknown, 1='<500', 2='<1000', 3='<1500', 4='<2000', 5='>2000

3.4.13 getCellChangeUsage(CellIndex, StartTime, Range)

Gibt Daten für Zellwechsel-Nutzung zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
Rückgabe – (*RC, ChangeNames, ChangePercentages, ChangeNumbers*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
ChangeNames	*Zellwechseltypen* (Array mit 6 Werten) 0=in progress, 1=accept, 2=broken, 3=reject, 4=leave, 5=update
ChangePercentages	Jeweiliger prozentualer Anteil des Zellwechseltyps (Array mit 6 Werten)
ChangeNumbers	Jeweiliger absoluter Anteil des Zellwechseltyps (Array mit 6 Werten)

3.4.14 getCellChangeType(CellIndex, StartTime, Range)

Gibt Daten für Zellwechsel-Nutzung zurück (Tortendiagram)

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
Rückgabe – (*RC, ChanegNames, ClearChangePercentages, ClearChangeNumbers, , AIEChangePercentages, AIEChangeNumbers*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt

ChangeNames	*Zellwechseltypen* (Array mit 6 Werten) 0=in progress, 1=accept, 2=broken, 3=reject, 4=leave, 5=update
ClearChangePercentages	Jeweiliger prozentualer Anteil des Zellwechseltyps, unverschlüsselte Träger (Array mit 6 Werten)
ClearChangeNumbers	Jeweiliger absoluter Anteil des Zellwechseltyps, unverschlüsselte Träger (Array mit 6 Werten)
AIEChangePercentages	Jeweiliger prozentualer Anteil des Zellwechseltyps, verschlüsselte Träger (Array mit 6 Werten)
AIEChangeNumbers	Jeweiliger absoluter Anteil des Zellwechseltyps, verschlüsselte Träger (Array mit 6 Werten)

3.4.15 getCellChangeType(CellIndex, StartTime, Range, TypeFlag)

Gibt Daten für benutzerdefinierte Signalisierungs Last über die Zeit zurück

Argumente	
CellIndex	Index der angezeigten Zelle
StartTime	time_t Startzeit für die ersten Daten in der Liste
Range	Anzahl der Werte in Sekunden (1 … 3600)
TypeFlag	0-4 – MASQoS Definitionen 1-5
Rückgabe – (*RC, LoadNames, LoadLists*)	
RC	0: *OK* 4: *Fehler*: Startzeit außerhalb des Aufnahmebereichs 11: *Fehler:* Zellindex unbekannt
LoadNames	Liste mit Gruppennamen, Länge wird als Index für *LoadList* benötigt
LoadList	Liste mit TCH-Signalisierungslast in Prozent. Die Liste ist zweidimensional. LoadList[*LoadNamesLength*][*Range*]

Bildverzeichnis