Universelles Videomanagementsystem von Accellence Technologies GmbH

vimacc SDK

TCP Steuerschnittstelle

Spezifikation der VIMACC_CONTROL Schnittstelle zur Steuerung eines **vimacc** Systems durch ein übergeordnetes Managementsystem

Protokoll-Version: **1.10.7**

Impressum

Herausgeber

Gesellschaft:	Accellence Technologies GmbH
Handelsregister:	HRB 110799 Hannover
Geschäftsführer:	Dipl.-Inf. (FH) Frank Christ, Dr.-Ing. Heinz Stephanblome
Redaktion:	Torsten Heinrich, Mike Plötz
Tel:	+49 (0)511 277 2400
Fax:	+49 (0)511 277 2499
E-Mail:	info@accellence.de
Internet:	www.accellence.de / www.vimacc.de
Anschrift:	Accellence Technologies GmbH Garbsener Landstraße 10, 30419 Hannover, Deutschland

 Datum: 24.03.2022

Inhaltsverzeichnis

Abkürzungsverzeichnis

ASCII	American Standard Code for Information Interchange
AAC	Advanced Audio Coding
AES	Advanced Encryption Standard
CA	Certificate Authority oder Certification Authority
CCTV	Closed Circuit Television
DB	Data Base / Datenbank
DCOM	Distributed Component Object Model
DVR	Digital Video Recorder
IP	Internet Protocol
iSCSI	Internet Small Computer System Interface
GUI	Graphical User Interface
GOP	Group Of Pictures
HID	Human Interface Device
LDAP	Lightweight Directory Access Protocol
MMI	Man Machine Interface
NAS	Network Attached Storage (Fibre Channel, iSCSI, …)
NFR	Non-functional Requirement
NTP	Network Time Protocol
NVR	Network Video Recorder
OPC	OLE for Process Control
RFC	Request for Comments
RTSP	RealTime Streaming Protocol
PTZ / SNZ	Pan Tilt Zoom / Schwenken Neigen Zoomen
PKCS	Public Key Cryptography Standards
PKI	Public-Key-Infrastruktur
SAN	Storage Area Network (CIFS, NFS, SMB, …)
SAS	Serial Attached SCSI
SDP	Session Description Protocol (see RFC 4566)
SHA	Secure Hash Algorithm
SQL	Structured Query Language
SRTP	Secure Real-Time Transport Protocol
SSD	Solid-State-Drive
SSL	Secure Sockets Layer
SW	Software
TCP	Transmission Control Protocol
UDP	User Datagram Procol

1 Einleitung

1.1 Zweck des Dokumentes

Das vorliegende Dokument gehört zur Systemdokumentation des Videomanagementsystems **vimacc®** der Accellence Technologies GmbH.

Es beschreibt Steuer- und Streaming-Schnittstellen eines **vimacc®** Systems zu übergeordneten Managementsystemen.

Das Deckblatt des Dokumentes verweist mit 3 Stellen auf die Versionsnummer des **vimacc®** Steuerprotokolls. Die letzte Stelle bezeichnet die Versionsnummer des Dokumentes selbst. (Bsp.: 1.10.5.2 -> Protokollversion 1.10.5, Revision 2)

1.2 Aufbau der Dokumentation

Die **vimacc®** Systemdokumentation besteht aus einer Reihe von Dokumenten, die jeweils einen Teilaspekt behandeln und in sich abgeschlossen sind.

Folgende Dokumente stehen standardmäßig zur Verfügung:

- **vimacc®** Systemdokumentation: Einführung
  Überblick über die allgemeinen Eigenschaften sich daraus ergebenen möglichen Einsatzgebiete.

- **vimacc®** Systemdokumentation: Eigenschaften
  Detaillierte Beschreibung technische Leistungsparameter und Eigenschaften

- **vimacc®** Systemdokumentation: Systemvoraussetzungen
  Informationen zu Minimalanforderungen an Hardware und Betriebssystem

- **vimacc®** Systemdokumentation: Systemplanung
  Randbedingungen, die bei der Planung eines Videosystems zu berücksichtigen sind und Hilfestellung bei der Dimensionierung des Gesamtsystems

- **vimacc®** Systemdokumentation: Bildquellenliste
  Liste verfügbare Bildquellen (Kameras, Encoder), Treiber und anderer anschließbaren Peripheriegeräte

- **vimacc®** Systemdokumentation: Architektur
  Detaillierten Einblick in die Architektur (internes Dokument)

1.3 vimacc Editionen

Die Videomanagementsoftware **vimacc®** wird in verschiedenen Editionen bereitgestellt, die sich im Funktionsumfang und Einsatzzweck unterscheiden.

vimacc® Professional
- Videomanagementsystem zur Installation auf einem einzigen PC oder Server
- Unterstützung für max. 64 Kameras in voller Auflösung und Framerate (entsprechende Hardwareausstattung vorausgesetzt)
- bis zu zwei zusätzliche abgesetzte Workstations
- gleichzeitige Anzeige von Live-, Playback und Alarm-Videos sowie Lagepläne
- unbegrenzte Monitoranzahl je Arbeitsplatz
- Speicherung der Videos im Originalformat
- Export von Videos auf CD/DVD oder USB-Datenträger inkl. aller Metadaten
- Integrierte Videowallfunktionalität
- Zeitsynchrones Playback mit bis zu 1000-facher Geschwindigkeit
- 100facher digitaler Zoom in Live- und Playbackvideos
- Interaktive und stapelbare Lagepläne
- Privatzonenmaskierung in zwei Sicherheitsstufen
- Steuerungsmöglichkeit des Videosystems über http- oder TCP-SDK
- Audio-Aufzeichnung/Wiedergabe

vimacc® Enterprise
- Unterstützt alle Funktionen aus **vimacc®** Professional
- Die Anzahl der Kameras und Workstations ist softwareseitig nicht begrenzt
- Dezentrale Installation der vimacc-Prozesse möglich
- Server-/Dienste-Redundanz und Load Balancing
- Integrierbar in Domänen-Infrastrukturen z.B. Active Directory
- Kopplungsmöglichkeit an externe Systeme (z.B. Leitsysteme, Interkom)

vimacc® OA Video-Subsystem für SCADA-Systeme z.B. SIMATIC WinCC OA
- Unterstützt alle Funktionen aus **vimacc®** Enterprise
- Besitzt keine Oberfläche zur Bedienung, sondern integriert sich vollständig und transparent in ein übergeordnetes SCADA-System

vimacc® Parking Spezial-Edition für Parkhausbetreiber
- Unterstützt alle Funktionen aus **vimacc®** Enterprise
- Tiefe Integration für Commend/Schneider Intercom-Systeme inkl. Sprechkanalsteuerung und Tor-/Schrankensteuerung

vimacc® Safe Office für Büroumgebungen mit erhöhtem Gefahrenpotential
- Unterstützt alle Funktionen aus **vimacc®** Enterprise
- Unterstützung von Alarmsensoren zur Aktivierung stiller Alarme
- automatische Anzeige von Alarmbildern in Nachbarbüros und/oder bei einem Sicherheitsdienst

vimacc® Besuchermanagement für die aktive Lenkung von Personen in Gebäuden
- Unterstützt alle Funktionen aus **vimacc®** Professional
- Zusätzlich direktes Wechselsprechen mit geeigneten Kameras

2 Steuerschnittstellen

2.1 Überblick

vimacc® verfügt über verschiedene Datenschnittstellen, um Verbindungen zwischen Streaming-Quellen und Streaming-Senken herzustellen oder Live-/Playback-Streaming-Daten an Third-Party-Prozesse weiter zu leiten.

Die von **vimacc®** bereitgestellten Funktionen werden mehreren Kommunikations-kanälen (CONTROL, LIVE, PLAYBACK) gemäß Abbildung 2.1 zugeordnet:

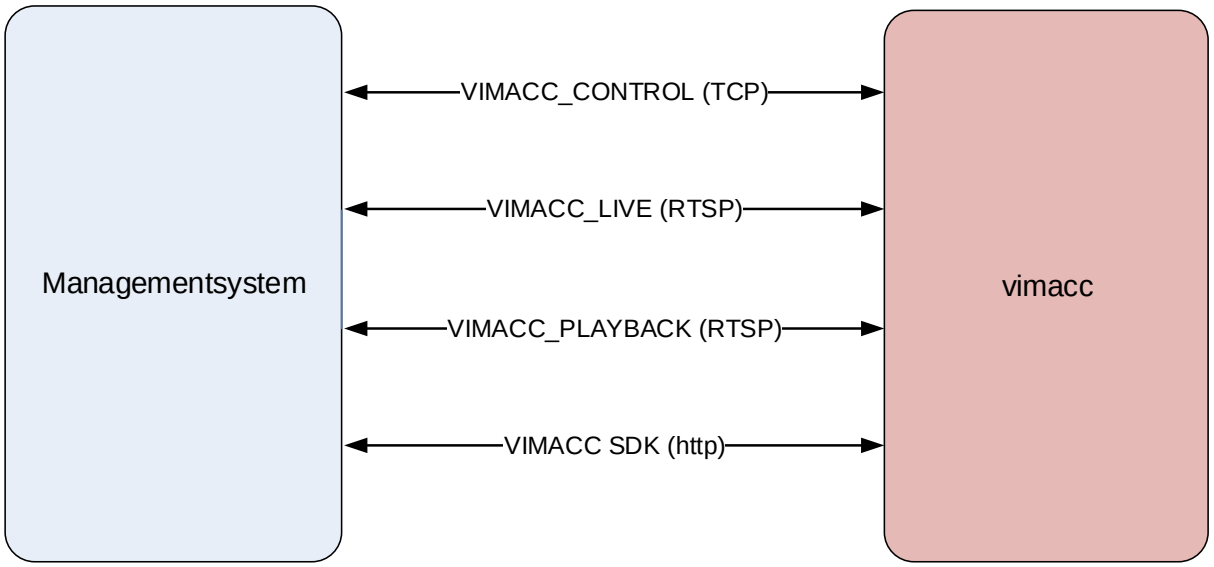

Abbildung 2.1: vimacc SDK - Schnittstellen

VIMACC_CONTROL (TCP): Normierte Steuerung von Verbindungen für die Livedarstellung und für den Archivzugriff, Steuerung von Geräten wie PTZ-Kameras, I/O-Kontakte, Signalisierung von Ereignissen, Laden und Ausführen von Skripten und das Auslesen von Geräte-Kennungslisten und Geräte-Zuständen.

VIMACC_LIVE (RTSP): Zugriff auf die Live-Streams der digitalen Videotechnik wie beispielsweise Netzwerkkameras oder Videoencoder.

VIMACC_PLAYBACK (RTSP): Zugriff auf die aufgezeichneten Streaming-Daten für eine externe Weiterverarbeitung wie z.B. Postanalyse-Prozesse, Playback etc.

VIMACC SDK (http): Die Steuer- und Streaming-Interfaces über das http-Protokoll werden in einem separaten Dokument beschrieben.
Siehe ***vimacc_Systemdokumentation_HTTP_Interface.pdf***

Unter dem Begriff Managementsystem (MMS) sind stellvertretend alle Gewerke zu verstehen, die eine Kopplung mit **vimacc®** implementieren z.B. Einsatz-Leit-Systeme (ELS), etc.

3 VIMACC_CONTROL

3.1 Allgemein

Über die Datenschnittstelle VIMACC_CONTROL kann ein **vimacc®** System mit einem externen System Informationen austauschen. Es können Verbindungen zwischen Videoquellen und Videosenken geschaltet werden und bei bestimmten Quellen (wie z.B. PTZ Kameras) zusätzliche Steuerungsbefehle gesendet werden.

Die Datenschnittstelle VIMACC_CONTROL ist als einfache, zustandslose Schnittstelle implementiert. Dies bedeutet, das **vimacc®** an der Schnittstelle keinerlei Zustände zu den geschalteten Verbindungen speichert und die steuernde Instanz immer auch verantwortlich für das Trennen der aufgebauten Verbindungen ist.

In Anlehnung an die Topologie des Zielsystems, in dem das **vimacc®** System als strukturiertes Untersystem betrieben werden kann, können in jedem Teilsegment **vimacc®** Steuerungs-Interfaces installiert und betrieben werden.

Jedes Steuerungs-Interface ist auf **vimacc®** Ebene als gleichwertig zu sehen, d.h. jedes einzelne Interface ist in der Lage, die Abläufe im gesamten **vimacc®** System zu steuern. Die Koordination dieser Abläufe (Zugriffsberechtigungen, Prioritätenregeln etc.) muss in dem übergeordneten Management-System bzw. der übergeordneten Steuerungs-Instanz erfolgen.

Die VIMACC_CONTROL Schnittstelle zur Steuerung eines **vimacc®** Systems ist in verschiedenen Varianten mit unterschiedlichem Funktionsumfang verfügbar.

Die Verfügbarkeit der entsprechenden Variante wird im Lizenzfile definiert und muss entsprechend im AccVimaccAdministrationCenter konfiguriert werden.
Siehe **vimacc®** Administrator-Handbuch

Die folgenden Protokoll-Varianten sind definiert:

- **VIMACC_CONTROL_BASIC**
  Dieses Protokoll stellt die einfachste Form der Steuerungs-Schnittstelle VIMACC_CONTROL dar.
  Es beinhaltet grundlegende Steuerungsbefehle zum Herstellen von Live-Videoverbindungen und zum Abfragen der Gerätezustände.

- **VIMACC_CONTROL_DEVICES_ALARMS_SCENARIOS**
  Dieses Protokoll erweitert das Protokoll VIMACC_CONTROL_BASIC um Kommandos zum Aufschalten von Szenarien, Übergeben von Alarmen an das **vimacc®** System und zum Annehmen und Abschließen von erzeugten Alarmen.

- **VIMACC_CONTROL_ALL**
  Dieses Protokoll erweitert das Protokoll VIMACC_CONTROL_DEVICES_ALARMS_SCENARIOS um Kommandos zum Schreiben von beliebigen sogenannten "Datenpunkten" des **vimacc®** Systems.

- **VIMACC_CONTROL_FALLBACK**
  Dieses Protokoll bietet nur einen sehr eingeschränkten Satz Kommandos, um einige Informationen über den Zustand eines **vimacc®** Systems abzufragen. Auf dieses Protokoll wird automatisch zurückgeschaltet, wenn ein **vimacc®** System nicht mehr ordnungsgemäß betrieben werden kann, beispielsweise weil die zu Grunde liegende Lizenz abgelaufen ist.

Über die VIMACC_CONTROL Schnittstelle ist kein direkter Zugriff auf die Streaming-Daten möglich.

3.2 Verbindungsaufbau

Ein MMS bzw. eine übergeordnete Steuerungs-Instanz kann je nach Bedarf beliebig viele TCP-Verbindungen zu den Server-Rechnern des **vimacc®** Systems aufbauen, auf denen ein **vimacc®** Steuerungs-Interface betrieben wird.
vimacc® begrenzt nicht die Anzahl der gleichzeitig geführten Verbindungen.

Ein MMS bzw. eine übergeordnete Steuerungs-Instanz sollte diese aufgebauten TCP-Verbindungen mit Prüftelegrammen überwachen. Im Falle eines erkannten Verbindungsfehlers sollte das MMS die Verbindungen schließen und nach einigen Sekunden einen neuen Verbindungsversuch starten.

3.3 Netzwerk- und Transportebene

- Netzwerkebene: IP
- Transportebene: TCP

3.4 Darstellungs- und Sitzungsebene

- Sitzungsebene: TCP Socketverbindungen
- Darstellungsebene: Klartext, UTF-8 kodiert

3.5 Anwendungsebene

Die Datenschnittstelle VIMACC_CONTROL_BASIC ist für die Anwendungsebene ein textorientiertes Protokoll.

vimacc® betreibt auf allen Server-Rechnern, die als Kommunikationsentitäten für das MMS betrieben werden, einen Prozess, der einen TCP-Server auf einem konfigurierbaren Port im Listen-Modus hält. Das MMS kann jederzeit eine TCP-Verbindung zu den Prozessadressen, bestehend aus der IP-Adresse und der Portnummer, aufbauen.

Nach erfolgreichem Verbindungsaufbau meldet sich der **vimacc®** TCP-Server mit Namen des freigegebenen Protokolls (siehe oben), der Versionsnummer des Protokolls und der Versionsnummer.

vimacc→MMS:
`<Protokoll-Name>:Version <Versionsnummer>;vimacc:Version <Versionsnummer>`

Anschließend initiiert das MMS eine Protokollsitzung, indem es sich beim **vimacc®** System authentifiziert (siehe Kapitel 3.7.2).

Nach initiierter Protokollsitzung kann das MMS Steuerdaten in Form von textbasierten Protokollkommandos über die TCP-Verbindung senden. Jedes Protokollkommando wird vom entsprechenden **vimacc®** Prozess stets über textbasierte Meldungen bestätigt. Der Kommandobestätigung können asynchron Ergebnisse der Kommandoausführung folgen.

Die Protokoll-Sequenzen werden UTF-8 kodiert auf der TCP-Verbindung übertragen. Alle Protokollkommando und Meldungen werden durch die Escape-Sequenz `\r\n` terminiert.

Alle Parameter der Protokollkommandos und der Meldungen werden als Text-Strings interpretiert.

3.6 Netzwerkressourcen

Der TCP-Server für die Datenschnittstelle VIMACC_CONTROL_BASIC ist unter einem TCP-Port erreichbar, der in der projektspezifischen **vimacc®** Installation konfiguriert werden kann.
Der Default Port ist 4227.

3.7 Telegrammbeschreibung

3.7.1 Aufbau der Steuerungsbefehle

Die Steuerdaten werden als ASCII-Strings übertragen und bestehen aus Schlüssel-Wert-Paaren, die durch ein Gleichheitszeichen ('=') verbunden sind.

Ein Steuerungsbefehl besteht immer aus einem Schlüssel-Wert-Paar, gefolgt von optionalen Parametern, die jeweils durch das Trennzeichen Semikolon (';') voneinander getrennt werden.
Ein Steuerungsbefehl besteht immer aus dem Schlüsselwort `cmd` mit dem Namen des Befehls als Wert, gefolgt von einer beliebig langen Parameterliste.
Die Parameter sind dabei ebenfalls als Schlüssel-Wert-Paar in der Form `<Parameter-Name>=Wert` zu übergeben.

Die Reihenfolge der Schlüssel-Wert-Paare ist nicht festgelegt.

Sollen in dem Wert eines Parameters Leerzeichen, Sonderzeichen (wie etwa (''), (\), (\r), (\n) und (\t)), oder die Zeichen (=) und (;) übergeben werden, so muss diesen Zeichen das Escape-Zeichen (\) vorangestellt werden (Vorgang genannt: "escapen").

Sollen in dem Wert eines Parameters ebenfalls Schlüssel-Wert-Paare übergeben werden, so sind diese mehrfach zu "escapen" (siehe Befehl `writeDp` Kapitel 3.7.5.2).

Ein komplettes Protokollkommando wird immer mit der Escape-Sequenz `\r\n` abgeschlossen.

Jeder Steuerungsbefehl wird von **vimacc®** mit einem Antwort-Text quittiert. Die Antworten werden ebenfalls als ASCII-Strings übertragen.
Jede Antwort besteht aus mehreren Schlüssel-Wert-Paaren, die jeweils durch das Trennzeichen Semikolon (';') voneinander getrennt werden.
Vorangestellt ist immer der Parameter `msgsize` gefolgt von der Anzahl der Zeichen, die in der Nachricht enthalten sind.
Daran anschließend folgt das Schlüsselwort `resp`, gefolgt von dem Namen des Steuerungsbefehls als Wert, gefolgt von der Liste der Parameter des Steuerungsbefehls, die durch das Trennzeichen Semikolon (';') voneinander getrennt werden. Die Antwort wird dann angefügt.
Der Wert des Parameters `msgsize` enthält immer die Anzahl der Zeichen des Antworttextes beginnend bei dem Schlüsselwort `resp`.

Beispiel:
MMS→vimacc:
`cmd=keepalive;userdata=1234`

vimacc→MMS:
`msgsize=38;resp=keepalive;userdata=1234;answer=ok`

Hinweis:
Die Steuerungsschnittstelle wird kontinuierlich erweitert. Dabei wird auf Kompatibilität mit älteren Protokollversionen geachtet. Allerdings ist es möglich, dass bei existierenden Kommandos oder Antworten zusätzliche Parameter eingefügt werden. Ein Protokollparser sollte also damit umgehen können, dass eine Antwort auf einen Steuerungsbefehl zusätzliche Parameter enthält.

Nachfolgend sind die möglichen Steuerbefehle und ihre Argumente benannt. Außerdem werden die Antworten genannt, die in jedem Fall von der Gegenstelle zurückgesandt werden.
Aus Gründen der Übersichtlichkeit wird bei der Angabe der Antworttexte der Eintrag `msgsize=<len>;` nicht mit angegeben.

3.7.2 Authentifizierung

Nach erfolgreichem TCP-Verbindungsaufbau wird die Authentifizierung durchgeführt. Für die Authentifizierung verwenden das MMS und **vimacc®** den gleichen Benutzernamen und das gleiche Passwort, wobei der Benutzername und das Passwort nicht im Klartext über das Netzwerk versendet werden.
Die Authentifizierung läuft dazu in Anlehnung an das Digest-Access-Authentication-Verfahren folgendermaßen ab:

Das MMS muss zunächst dem **vimacc®** TCP Server seinen Anmeldewunsch signalisieren.
Anschließend wird der **vimacc®** TCP Server einen zufälligen Text generieren (die sogenannte 'server challenge') und an den Client zurücksenden. Der Client muss daraufhin eine Textfolge bilden, die sich aus diesem zufälligen Text, dem Benutzernamen und dem Passwort zusammensetzt und daraus einen MD5-Hashwert generieren.
Die ASCII Repräsentation der hexadezimalen Darstellung dieses Hashwertes muss dem TCP Server zur Überprüfung gesendet werden, woraufhin dieser die Anmeldung bestätigen oder ablehnen kann.

Ablauf der Kommandos und die zugehörige Antworten:

1. MMS→vimacc:
   `cmd=login;userdata=<text>`

   vimacc→MMS:

```
resp=login;userdata=<text>;answer=failed,access denied;
serverchallenge=<random string>
```

2. MMS→vimacc:

```
cmd=login;userdata=<text>;clientresponse=<Ascii(Md5(<usernam
e>:<password>:<random string from serverchallenge>))>
```

vimacc→MMS:

a) Antwort des Clients korrekt:

```
resp=login;userdata=<text>;clientresponse=<Ascii(Md5(<userna
me>:<password>:<random string from serverchallenge>))>;
answer=ok,access granted
```

b) Antwort des Clients nicht korrekt:

```
resp=login;userdata=<text>;clientresponse=Ascii(Md5(<usernam
e>:<password>:<random string from serverchallenge>));
answer=failed,access denied;serverchallenge=<random string>
```

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:
Angenommen seien die folgenden Zugangsdaten:

Benutzername: "UserName"

Passwort: "Password"

- MMS→vimacc:

```
cmd=login;userdata=1234
```

vimacc→MMS:

```
resp=login;userdata=1234;answer=failed,access denied;
serverchallenge=75798a683873f75071b7da939173f09a
```

- MMS→vimacc:

Auf Seiten des Clients muss der folgende Hashwert berechnet werden:
Ascii(Md5("UserName:Password:75798a683873f75071b7da939173f09a")) =
792604ca7fb36e0177f24899e004590b

```
cmd=login;userdata=1234;clientresponse=792604ca7fb36e0177
f24899e004590b
```

vimacc→MMS:

```
resp=login;userdata=1234;clientresponse=792604ca7fb36e017
7f24899e004590b;answer=ok,access granted
```

3.7.3 VIMACC_CONTROL_BASIC

3.7.3.1 Allgemein

Dieses Protokoll stellt die einfachste Form der Steuerungs-Schnittstelle VIMACC_CONTROL dar.
Es beinhaltet grundlegende Steuerungsbefehle zum Herstellen von Live-Videoverbindungen, zum Abfragen der Gerätelisten und zum Abfragen von Gerätezuständen.

3.7.3.2 Abfragen der verfügbaren Befehle

Kommando:
`cmd=help;userdata=<text>`

Antwort:
`resp=help;userdata=<text>;answer=ok,parameterlist{\r\nbefehl#1 \r\n...befehl#n\r\n]}`

Dieses Kommando dient zur Abfrage der verfügbaren Steuerungsbefehle, die in der installierten Protokoll-Variante zur Verfügung stehen.
Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:

- MMS→vimacc:
  `cmd=help;userdata=1234`

  vimacc→MMS:
  `resp=help;userdata=1234;answer=ok,parameterlist{\r\nlogin \r\nhelp\r\nkeepalive\r\nshow\r\n… subscribeevents\r\n}`

3.7.3.3 Überwachung der Steuerverbindung

Kommando:
`cmd=keepalive;userdata=<text>`

Antwort:
`resp=keepalive;userdata=<text>;answer=ok`

Das MMS sollte alle 5 Sekunden einen `keepalive` Befehl senden. Bleibt dieser Befehl aus, so wird **vimacc®** die Steuerverbindung schließen. Das MMS muss daraufhin eine neue Steuerverbindung aufbauen und muss sich erneut authentifizieren.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:

- MMS→vimacc:
  `cmd=keepalive;userdata=1234`
  vimacc→MMS:
  `resp=keepalive;userdata=1234;answer=ok`

3.7.3.4 Aufschalten von Live- oder Wiedergabe-Streams

Kommando:
`cmd=show;contextid=<text>;deviceid=<`*`KameraID`*`>;dest=<`*`SenkenID`*`>;`
`videodlg=<`*`number`*`>;userdata=<text>`

Antwort:
`resp=show;contextid=<text>;deviceid=<`*`KameraID`*`>;dest=<`*`SenkenID`*`>`
`;videodlg=<`*`number`*`>;userdata=<text>;answer=ok|failed`

Verschaltet werden immer Streaming-Quellen mit Streaming-Senken. Als Senke kommen sowohl **vimacc®** Display-Instanzen als auch **vimacc®** Workstation-Instanzen in Frage. Diese werden durch den Parameter `dest` festgelegt.
Als Streaming-Quellen können sowohl Live- als auch Wiedergabe-Streams verwendet werden.
Wiedergabe-Streams müssen über das Kommando `streamcontrol` nach der Aufschaltung gesteuert werden (siehe Kapitel 3.7.3.11).

Über den Parameter `videodlg` wird die Nummer eines Videoquadranten der Workstation adressiert. Es sind hier nur ganzzahlige Werte größer oder gleich 1 zulässig.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Der Parameter `contextid` wird nicht ausgewertet, sondern dient zur Protokollierung und wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Die Anzahl der maximal verfügbaren Videodialoge und die Anordnung im Anzeigebereich sind von der verwendeten **vimacc®** Display- bzw. Workstation-Applikationen abhängig.
Die Zählung der Videodialoge erfolgt dabei immer von links oben nach rechts unten.

Ein Beispiel für ein Nummerierungsschema zeigt die folgende Abbildung:

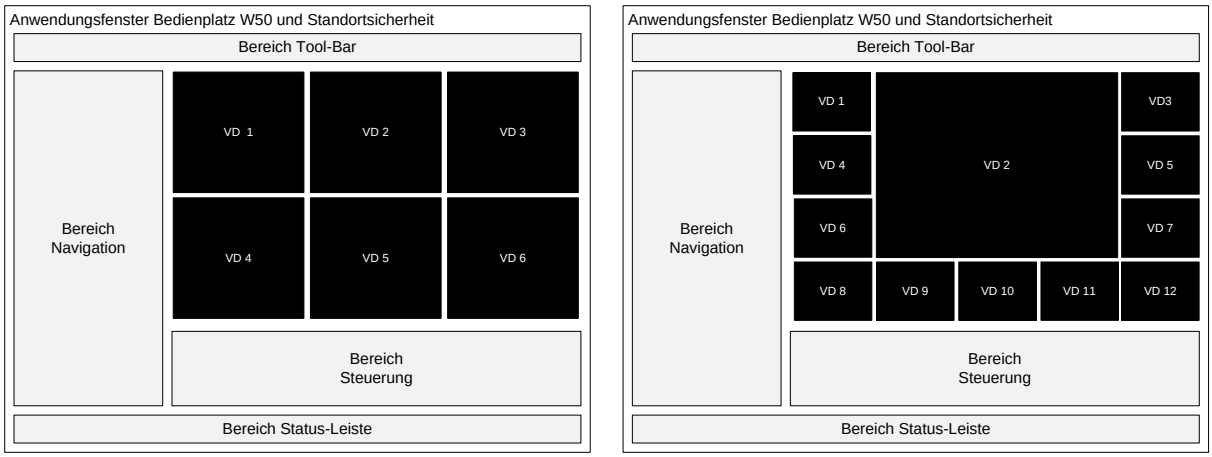

Abbildung 3.1: Nummerierungsschema der Videodialoge einer vimacc Workstation

Beispiele:

- MMS→vimacc:
  `cmd=show;contextid=778;deviceid=Camera_0001;dest=AP_1;videodlg=5`

  vimacc→MMS:
  `resp=show;contextid=778;deviceid=Camera_0001;dest=AP_1;videodlg=5;answer=ok`

3.7.3.5 Trennen von Verbindungen zu Live- oder Wiedergabe-Streams

Kommando:
`cmd=clear;contextid=<text>;dest=<`*`SenkenID`*`>;videodlg=<`*`number`*`>;userdata=<text>`

Antwort:
`resp=clear;contextid=<text>;dest=<`*`SenkenID`*`>;videodlg=<`*`number`*`>;userdata=<text>;answer=ok|failed`

Vorab aufgebaute Verbindungen von Live- oder Wiedergabe-Streams werden immer über die Streaming-Senken wieder abgebaut. Als Senke kommen sowohl **vimacc®** Display-Instanzen als auch **vimacc®** Workstation-Instanzen in Frage.

Der Parameter `dest` gibt die ID der Streaming-Senken an.

Über den Parameter `videodlg` wird die Nummer eines Videoquadranten der Workstation adressiert (→ Befehl `cmd=show`). Wird hier der Wert `0` übergeben, so werden alle Videoquadranten gelöscht.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiele:

- MMS→vimacc:
  `cmd=clear;contextid=778;dest=AP_1;videodlg=5`

  vimacc→MMS:
  `resp=clear;contextid=778;dest=AP_1;videodlg=5;answer=ok`

3.7.3.6 Aufschalten von Sequenzen

(verfügbar ab vimacc 2.2.8.7)

Kommando:
`cmd=startsequence;contextid=<text>;sequence=<`*`SequenzID|SequenzName`*`>;dest=<`*`SenkenID`*`>;videodlg=<`*`number`*`>;userdata=<text>`

Antwort:
`resp=startsequence;contextid=<text>;sequence=<`*`SequenzID|SequenzName`*`>;dest=<`*`SenkenID`*`>;videodlg=<`*`number`*`>;userdata=<text>;answer=ok|failed`

Die Aufschaltung von Sequenzen erfolgt analog zu der Aufschaltung von Kameras. Es wird immer eine Sequenz auf eine Streaming-Senke aufgeschaltet. Als Senke kommen sowohl **vimacc®** Display-Instanzen als auch **vimacc®** Workstation-Instanzen in Frage. Diese werden durch den Parameter `dest` festgelegt.

Über den Parameter `videodlg` wird die Nummer eines Videoquadranten der Workstation adressiert. Es sind hier nur ganzzahlige Werte größer oder gleich 1 zulässig.
Die Anzahl der maximal verfügbaren Videodialoge und die Anordnung im Anzeigebereich sind von der verwendeten **vimacc®** Display- bzw. Workstation-Applikationen abhängig.
Die Zählung der Videodialoge erfolgt dabei immer von links oben nach rechts unten. (siehe Kapitel 3.7.3.4).

Die Auswahl der Sequenz erfolgt über den Parameter `sequence`. Hierbei kann sowohl die ID als auch der Name der Sequenz verwendet werden.
Die Liste der verfügbaren Sequenzen kann vom **vimacc** System abgefragt werden (siehe Kapitel 3.7.3.22).
Groß-/Kleinschreibung des Sequenznamens muß beachtet werden!

Nachdem die Sequenz erfolgreich aufgeschaltet wurde, startet automatisch deren Abarbeitung auf der zugehörigen **vimacc®** Display- bzw. Workstation-Instanz. Der Ablauf kann allerdings durch ein zusätzliches Kommando noch gesteuert werden (siehe Kapitel 3.7.3.7).

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Der Parameter `contextid` wird nicht ausgewertet, sondern dient zur Protokollierung und wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiele:

- MMS→vimacc:
  ```
  cmd=startsequence;contextid=778;sequence=sequenz_0001;dest=AP_1;videodlg=5;userdata=userdata#1
  ```

  vimacc→MMS:
  ```
  resp=startsequence;contextid=778;sequence=sequenz_0001;dest=AP_1;videodlg=5;userdata=userdata#1;answer=ok
  ```

3.7.3.7 Steuern von aufgeschalteten Sequenzen

(verfügbar ab vimacc 2.2.8.7)

Kommando:
```
cmd=controlsequence;contextid=<text>;dest=<destinationID>;videodlg=<number>;action=<play|pause|stop|goto>;position=<position>;userdata=<text>
```

Antwort:
```
resp=controlsequence;contextid=<text>;dest=<destinationID>;videodlg=<number>;action=<play|pause|goto>;(optional:
position=<position>;)userdata=<text>;answer=ok|failed
```

Nachdem eine Sequenz auf einen Videodialog aufgeschaltet wurde (siehe Kapitel 3.7.3.6), kann deren Ablauf über dieses Kommando gesteuert werden.

Der Parameter `dest` bestimmt die Instanz, auf der die Sequenz aufgeschaltet worden ist.

Über den Parameter `videodlg` wird die Nummer eines Videoquadranten der Workstation adressiert (siehe Kapitel 3.7.3.4).

Das entsprechende Steuerkommando wird über den Parameter `action` festgelegt. Gültige Schlüsselwörter für den Parameter `action` sind:

- `pause` Sequenz pausieren

- `play` Sequenz starten
- `goto` bestimmte Position innerhalb der Sequenz auswählen.

Beim dem Steuerkommando `action=goto` wird zusätzlich noch der Parameter `position` erwartet, in dem durch eine ganzzahlige Nummer (größer 0) die gewünschte Position der Sequenz festgelegt werden muss.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Der Parameter `contextid` wird nicht ausgewertet, sondern dient zur Protokollierung und wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiele:

- MMS→vimacc:
  `cmd=controlsequence;contextid=778;dest=AP_1;videodlg=5;action=pause;userdata=userdata#1`

  vimacc→MMS:
  `resp=controlsequence;contextid=778;dest=AP_1;videodlg=5;action=pause;userdata=userdata#1;answer=ok`

- MMS→vimacc:
  `cmd=controlsequence;contextid=779;dest=AP_1;videodlg=5;action=goto;position=3;userdata=userdata#1`

  vimacc→MMS:
  `resp=controlsequence;contextid=779;dest=AP_1;videodlg=5;action=goto;position=3;userdata=userdata#1;answer=ok`

3.7.3.8 Setzen der Geometrie einer vimacc Workstation-Instanz

Kommando:

`cmd=setworkstationgeometry;dest=<`*`SenkenID`*`>;xpos=<Wert>;ypos=<Wert>;width=<Wert>;height=<Wert>;showframe=<0|1>;topmost=<0|1>;hidden=<0|1>;shownavigation=<0|1>;showcontrols=<0|1>;title=<text>;contextid=<text>;userdata=<text>`

Antwort:

`resp=setworkstationgeometry;dest=<`*`SenkenID`*`>;xpos=<Wert>;ypos=<Wert>;width=<Wert>;height=<Wert>;showframe=<0|1>;topmost=<0|1>;hidden=<0|1>;shownavigation=<0|1>;showcontrols=<0|1>;title=<text>;contextid=<text>;userdata=<text>;answer=ok|failed`

Mit diesem Kommando können die Position, die Größe und das Aussehen einer **vimacc** Workstation-Instanz bestimmt werden.

Die Parameter `xpos, ypos, width` und `height` bestimmen die geometrischen Eigenschaften des Anwendungsfensters der **vimacc®** Workstation-Instanz.

Der Parameter `showframe` bestimmt, ob die Anwendung mit einem Rahmen oder rahmenlos dargestellt werden soll.

Der Parameter `shownavigation` bestimmt, ob der linke Navigationsbereich der Anwendung dargestellt werden soll.

Der Parameter `showcontrols` bestimmt, ob der untere Bereich der Steuerelemente der Anwendung dargestellt werden soll.

Der Parameter `topmost` bestimmt, ob das Anwendungsfenster generell immer über allen anderen Fenstern dargestellt werden soll.

Der Parameter `hidden` bestimmt, ob die Anwendung sichtbar oder nicht sichtbar sein soll.

Der Parameter `title` bestimmt den Text, der in der Titelzeile der adressierten **vimacc®** Workstation Instanz eingeblendet werden soll.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Hinweis:
Bei diesem Kommando sind die Parameter `xpos, ypos, width, height, showframe, showcontrols, shownavigation, topmost, hidden` , und `title` optional, d.h. es müssen nicht immer alle Parameter innerhalb des Kommandos übertragen werden. Das Kommando
`cmd=setworkstationgeometry;dest=4001;contextid=1234;title=Test`
ist ebenso zulässig wie das Kommando
`cmd=setworkstationgeometry;dest=4001;contextid=1234;hidden=1`

Beispiel:
MMS→vimacc:
`cmd=setworkstationgeometry;dest=4001;xpos=0;ypos=0;width=1024;height=768;showframe=1;topmost=1;hidden=0;shownavigation=1;showcontrols=<0|1>;titel=Videomodul;contextid=1234`

vimacc→MMS:
`resp=setworkstationgeometry;dest=4001;xpos=0;ypos=0;width=1024;height=768;showframe=1;topmost=1;shownavigation=1;showcontrols=1;hidden=0;titel=Videomodul;contextid=1234;answer=ok`

Die folgende Abbildung zeigt die Standard-Darstellungsform einer **vimacc** Workstation. Diese Darstellungsform wird ebenfalls eingenommen nach Empfang der Parameter `showframe=1;shownavigation=1;showcontrols=1;`

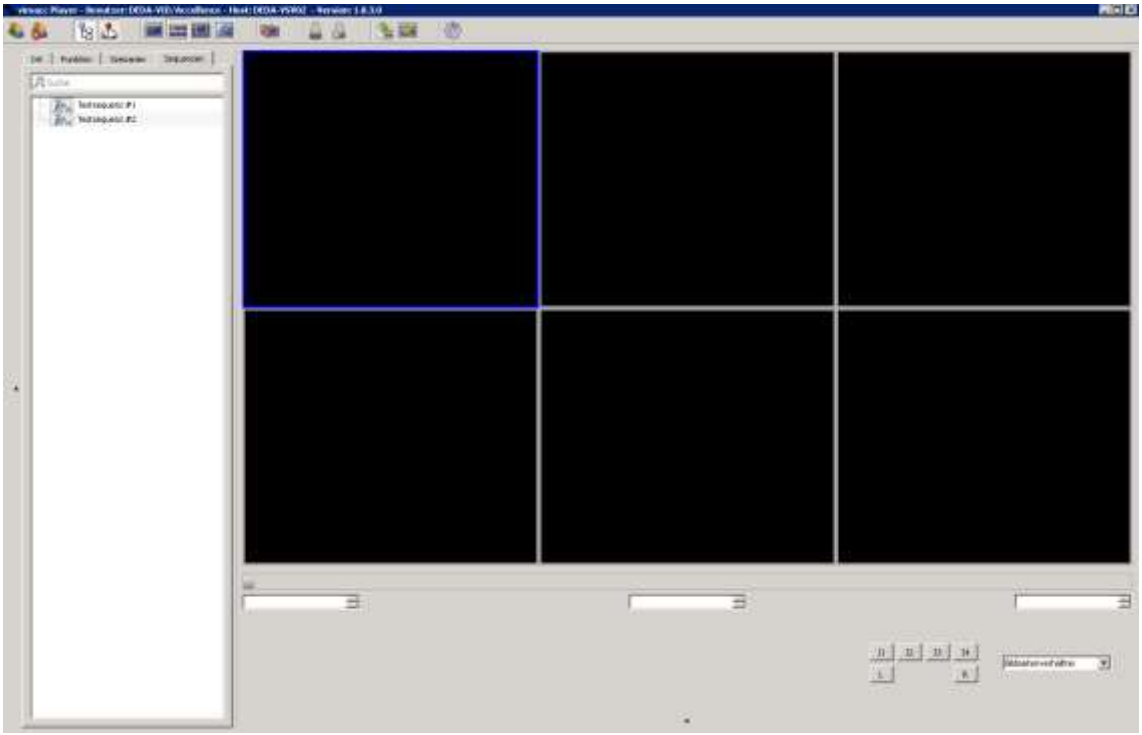

Abbildung 3.2: vimacc Workstation mit Rahmen, mit Navigationsbereich, mit Steuerelementen

Die folgende Abbildung zeigt eine **vimacc®** Workstation-Instanz nach Senden der Parameter `showframe=1;shownavigation=0;showcontrols=1;`

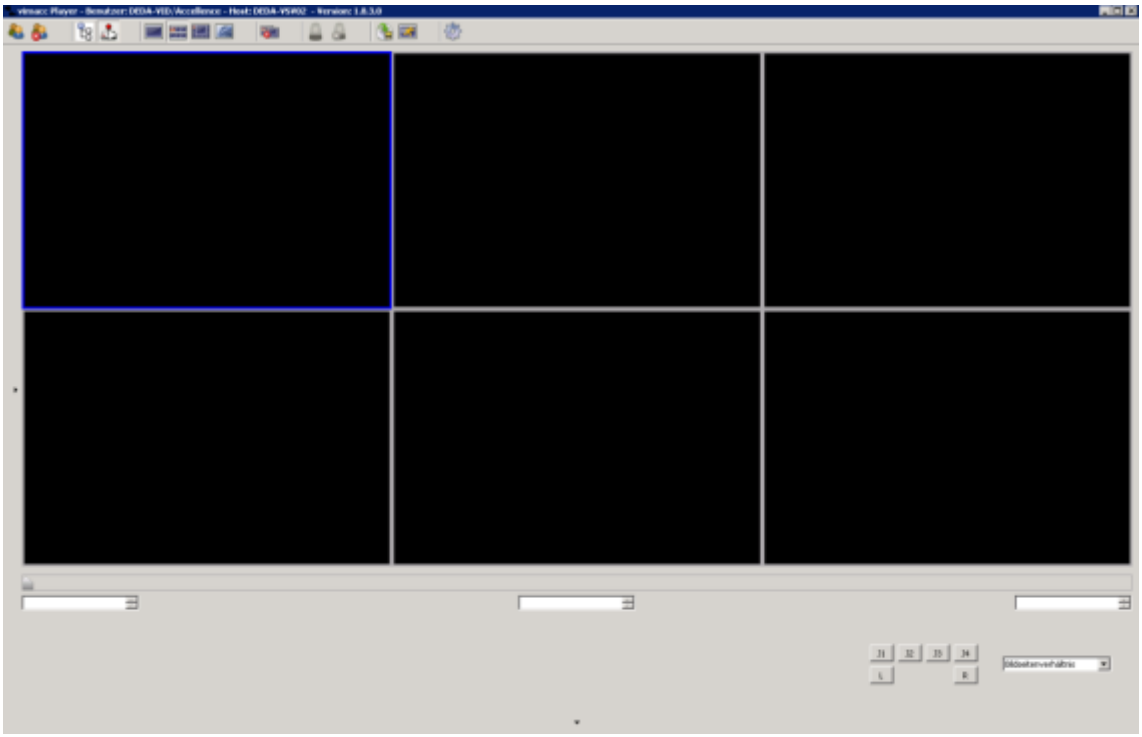

Abbildung 3.3: vimacc Workstation mit Rahmen und Steuerelementen, ohne Navigationsleiste

Die folgende Abbildung zeigt eine **vimacc®** Workstation-Instanz nach Senden der Parameter `showframe=1;shownavigation=0;showcontrols=0;`

Abbildung 3.4: vimacc Workstation mit Rahmen, ohne Navigationsbereich und ohne Steuerelemente

Die folgende Abbildung zeigt eine **vimacc®** Workstation-Instanz nach Senden der Parameter `showframe=0;shownavigation=0;showcontrols=0;`

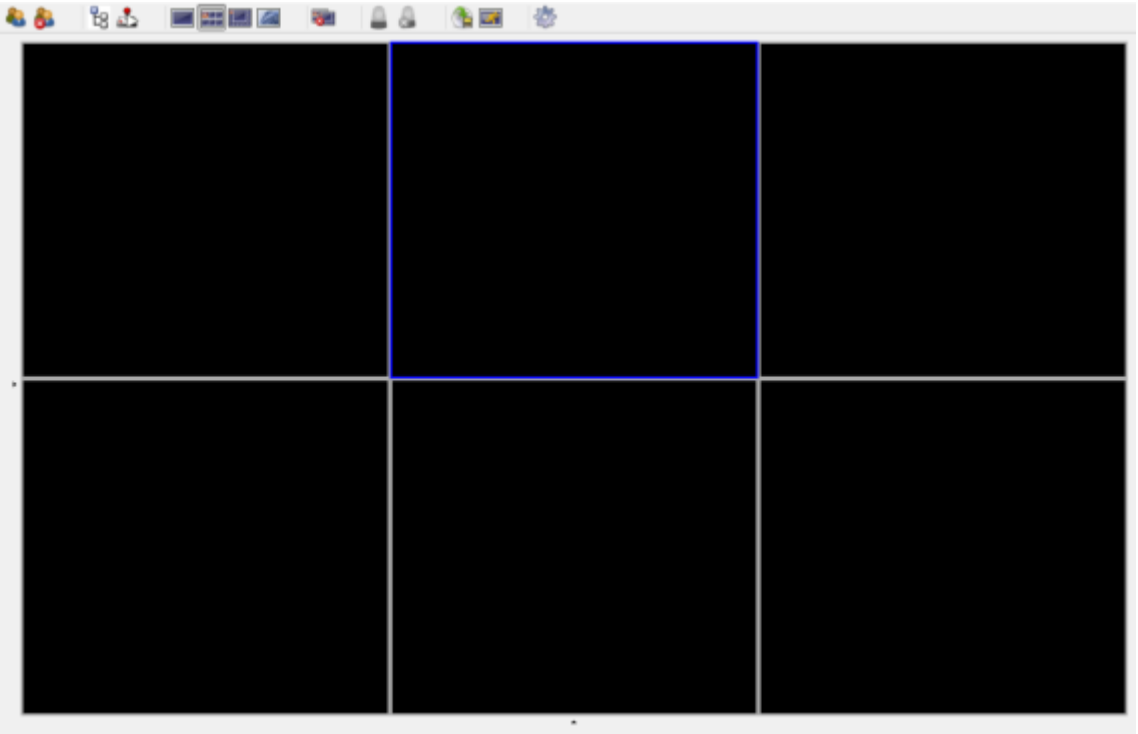

Abbildung 3.5: vimacc Workstation ohne Rahmen, Navigationsbereich und Steuerelemente

3.7.3.9 Setzen der Anordnung der Videodialoge einer vimacc Workstation-Instanz

Kommando:

`cmd=setworkstationgrid;dest=<SenkenID>;(dialogcount<Wert>|gridlayout=<Layoutname>);clearunused=<0|1:D=0>;contextid=<text>;userdata=<text>`

Antwort:

`resp=setworkstationgrid;dest=<SenkenID>;>;(dialogcount<Wert>|gridlayout=<Layoutname>);clearunused=<0|1:D=0>;contextid=<text>;userdata=<text>; answer=ok|failed`

Mit diesem Kommando kann die Anordnung der darzustellenden Videodialoge einer **vimacc** Workstation-Instanz festgelegt werden.

Die Parameter `dialogcount` legt die Anzahl der gleichzeitig darzustellenden Videodialoge fest. Abhängig von der installierten Ausprägung der **vimacc®** Workstation-Instanz wird dabei automatisch das zugehörige Layout ausgewählt. Abbildung 3.1 zeigt beispielhaft das eingestellte Layout für die Werte `dialogcount=6` und `dialogcount=12`.

Anstelle des Parameters `dialogcount` kann der Parameter `gridlayout` verwendet werden, um das gewünschte Layout der Videodialoge namentlich zu benennen.

Abhängig von der installierten Ausprägung der **vimacc®** Workstation-Instanz können dabei verschiedene Layouts ausgewählt werden. Durch die Verwendung des Parameters `gridlayout` können Anordnungen ausgewählt werden, die durch die Angabe der Dialoganzahl nicht eindeutig festgelegt werden könnten. Die Angabe `gridlayout=2x3` wählt beispielsweise ein zweizeiliges und dreispaltiges Layout aus, was durch die Angabe `dialogcount=6` nicht eindeutig bestimmt werden könnte. Abbildung 3.6 zeigt beispielhaft weitere Möglichkeiten für die Anordnung von Videodialogen.

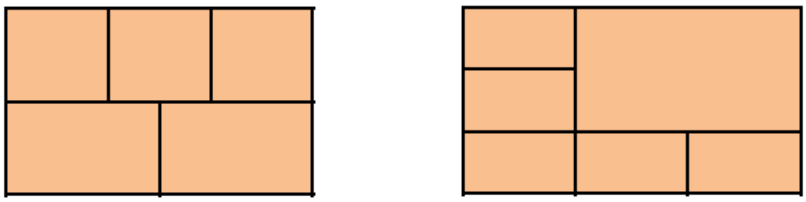

Abbildung 3.6: vimacc Workstation Layouts: "3+2" und "5+1"

Die maximale Anzahl der darstellbaren Videodialoge ist ebenfalls abhängig von der installierten Ausprägung der **vimacc®** Workstation-Instanz. Üblicherweise können die **vimacc®** Workstation-Instanzen bis zu 24 Videodialoge gleichzeitig darstellen.

Der Parameter `clearunused` bestimmt, ob bestehende Aufschaltungen, die nach Umschalten auf das gewählte Layout nicht mehr sichtbar sind, automatisch getrennt werden sollen.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc®** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:
MMS→vimacc:
```
cmd=setworkstationgrid;dest=1071;dialogcount=6;clearunused=1;c
ontextid=1234;userdata=test
```

vimacc→MMS:
```
resp=setworkstationgrid;dest=1071;dialogcount=6;clearunused=1;
contextid=1234;userdata=test;answer=ok
```

MMS→vimacc:
```
cmd=setworkstationgrid;dest=1071;gridlayout=2x3;clearunused=1;
contextid=1234;userdata=test
```

vimacc→MMS:
`resp=setworkstationgrid;dest=1071;gridlayout=2x3;clearunused=1;contextid=1234;userdata=test;answer=ok`

3.7.3.10 Festlegen der Darstellung innerhalb der Videodialoge einer Workstation-Instanz

Kommando:
`cmd=setworkstationscalemode;dest=<SenkenID>;cmdparam=<ZOOM|FIT|KEEP>;contextid=<text>;userdata=<text>`

Antwort:
`resp=setworkstationscalemode;dest=<SenkenID>;cmdparam=<ZOOM|FIT|KEEP>;contextid=<text>;userdata=<text>; answer=ok|failed`

Mit diesem Kommando kann das Skalierverhalten der Videodialoge einer **vimacc®** Workstation-Instanz festgelegt werden.

Die Größe der einzelnen Videodialoge einer **vimacc®** Workstation-Instanz ist abhängig von der Fenstergröße der Anwendung, die einerseits vom Benutzer selbst verändert werden kann, andererseits durch die maximale Auflösung des Ausgabebildschirms begrenzt ist. Somit kann sich das resultierende Seitenverhältnis der Videodialoge derart ändern, dass dieses nicht dem Seitenverhältnis des übertragenen Videostreams entspricht.
Durch das Kommando `setworkstationscalemode` kann festgelegt werden, auf welche Weise das Videobild des aufgeschalteten Streams in dem Videodialog dargestellt werden soll.

Folgende Werte können für den Parameter `cmdparam` übergeben werden:

- `KEEP` Beibehalten des Bildseitenverhältnisses des Videostreams

  Der Videostreams wird so in dem Videodialog dargestellt, dass dessen Bildseitenverhältnisses unverändert bleibt. Durch diese Art der Darstellung entstehen je nach Größe und Seitenverhältnis der Videodialoge schwarze Ränder.

- `ZOOM` Zoomen auf die Größe des Videodialoges

  Das Videobild wird derart skaliert, dass es ohne schwarze Ränder in den Videodialog eingepasst wird. Dadurch werden allerdings Bildbereiche abgeschnitten.

- `ZOOM` Aufziehen auf die Größe des Videodialoges

  Das Videobild wird derart skaliert, dass es horizontal oder vertikal skaliert wird, so dass der gesamte Videodialog ausgefüllt wird. Je nach Seitenverhältnis des

Videodialoges wird das Videobild dadurch mehr oder weniger verzerrt.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:
MMS→vimacc:
```
cmd=setworkstationscalemode;dest=1071;cmdparam=KEEP;contextid=
1234;userdata=test
```

vimacc→MMS:
```
resp=setworkstationscalemode;dest=1071;cmdparam=KEEP;contextid
=1234;userdata=test;answer=ok
```

3.7.3.11 Steuerung eines Wiedergabe-Streams

Kommando:
```
cmd=streamcontrol;mode=playback;dest=<SenkenID>;streamcmd=<sta
rt | pause | speed | posu | posa | posr | stfw | strw | pint>;
cmdparam=<parameter>;contextid=<text>;userdata=<text>
```

Antwort:
```
resp=streamcontrol;mode=playback;dest=<WorkstationId>;streamcm
d=<start | pause | speed | posu | posa | posr | stfw | strw |
pint>;cmdparam=<parameter>;contextid=<text>;userdata=<text>;an
swer=ok|failed
```

Die Steuerung eines Wiedergabe-Streams erfolgt immer über Steuerung der **vimacc®** Display- bzw. Workstation-Instanz, auf der der entsprechende Stream mit dem Kommando `show` (siehe Kapitel) aufgeschaltet worden ist.

Da die **vimacc®** Display- bzw. Workstation-Instanzen immer alle momentan aufgeschalteten Wiedergabe-Streams zeitlich synchronisieren, wirkt sich das Steuerungskommando `streamcontrol` immer auf alle Wiedergabe-Streams der steuernden Instanz aus. (Aus diesem Grund entfällt bei diesem Kommando die Angabe einer Videodialognummer.)

Der Parameter `dest` gibt die **vimacc®** ID der zu steuernden Display- bzw. Workstation-Instanz an.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Das entsprechende Steuerkommando wird mit dem Parameter `streamcmd` festgelegt. Gültige Schlüsselwörter für diesen Parameter sind:

- `start` Starten der Wiedergabe

  Optional kann über den Parameter `cmdparam` die Geschwindigkeit im Bereich -500 bis +500 angegeben werden. Der Default-Wert für die Geschwindigkeit ist +100.
  Ein positiver Wert bewirkt das Abspielen in normaler Richtung, ein negativer Wert bewirkt das Abspielen in umgekehrter Richtung.
  z.B. `streamcmd=start;cmdparam=200` vorwärts abspielen mit doppelter Geschwindigkeit

  `streamcmd=start;cmdparam=-200` rückwärts abspielen mit doppelter Geschwindigkeit

- `pause` Pausieren der Wiedergabe

  Es wird ein Standbild angezeigt, sofern auf der aktuellen Position eine Bild zu finden ist. Pause sollte eingenommen werden, wenn z.B. per Positions-Slider gesucht wird.

- `speed` Setzen der Wiedergabegeschwindigkeit

  Im Parameter `cmdparam` wird Angabe der Geschwindigkeit im Bereich -500 bis +500 erwartet. Der Default-Wert für die Geschwindigkeit ist +100.
  Ein positiver Wert bewirkt das Abspielen in normaler Richtung, ein negativer Wert bewirkt das Abspielen in umgekehrter Richtung.

- `posu` Positionsupdate abonnieren alle x Millisekunden

  Im Parameter `cmdparam` wird Angabe eines Zeitintervalls `<intervall ms>` erwartet.
  Im `events` Datenpunkt der **vimacc®** Display- bzw. Workstation-Instanz (siehe Kapitel 3.7.3.24) wird daraufhin alle x Millisekunden die aktuelle Wiedergabeposition ausgegeben.

  Hinweise:
  Ein zu kleiner Wert führt zu einer sehr hohen Anzahl von Meldungen!

- `posa` Anfahren einer absoluten Zeitposition

Im Parameter `cmdparam` wird Angabe eines UTC Zeitstempels in der Notation `yyyy-MM-dd'T'hh:mm:ss.zzz` erwartet.

- `posr` Relativ Positionieren

  Im Parameter `cmdparam` wird der Offset in der Form `Sekunden.Millisekunden` erwartet (`ss.zzz`).

- `stfw` Einzelbild vorwärts

  Dieses Kommando ist nur im Zustand `Pause` sinnvoll.

- `strw` Einzelbild rückwärts

  Dieses Kommando ist nur im Zustand `Pause` sinnvoll.

- `pint <0|1>` Pauseninterpretation ein- und ausschalten

  Im Parameter `cmdparam` wird der Wert 0 oder 1 erwartet.
  1: Pausen zeitrichtig wiedergeben,
  0: Pausen überspringen

Beispiele:

- MMS→vimacc:
  ```
  cmd=streamcontrol;mode=playback;dest=1071;contextid=1234;
  userdata=test;streamcmd=start;cmdparam=500
  ```

  vimacc→MMS:
  ```
  resp=streamcontrol;mode=playback;dest=1071;contextid=1234
  ;userdata=test;streamcmd=start;cmdparam=500;answer=ok
  ```

- MMS→vimacc:
  ```
  cmd=streamcontrol;mode=playback;dest=1071;contextid=1234;
  userdata=test;streamcmd=posa;cmdparam=2014-04-
  02T11:17:37.000
  ```

  vimacc→MMS:
  ```
  resp=streamcontrol;mode=playback;dest=1071;contextid=1234
  ;userdata=test;streamcmd=posa;cmdparam=2014-04-
  02T11:17:37.000;answer=ok
  ```

3.7.3.12 Steuerung einer PTZ-Kamera

Kommando:

`cmd=<move|iris|focus>;<keyword>=<% speed>[;>;<keyword>=<% speed>…];contextid=<text>;userdata=<text>;source=<`*`KameraID`*`>|iris|focus`

Antwort:

`resp=<move|iris|focus>;<keyword>=<% speed>[;>;<keyword>=<% speed>…];contextid=<text>;userdata=<text>;source=<`*`KameraID`*`>;answer=ok|failed`

Bei Steuerungsbefehlen zum Bewegen des Schwenk-/Neige-/Zoom-Mechanik einer PTZ-Kamera muss immer im Parameter `source` die ID der entsprechenden Kamera übergeben werden.

Es können gleichzeitig Bewegungen für verschiedene Achsen angegeben werden, die durch ein Semikolon getrennt sind.

Es können auch Kommandos abgesetzt werden, die nicht in der entsprechenden Kamera implementiert sind. So kann es z.B. vorkommen, dass eine Kamera keine automatische Blendensteuerung kennt (s.u.). Das von **vimacc®** abgesetzte Kommando würde in diesem Fall keine Reaktion bewirken. Über die hier beschriebene Schnittstelle würde der abgesetzte Befehl trotzdem positiv quittiert werden.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc®** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Gültige Schlüsselwörter für den Parameter <`keyword`> des Kommandos `move` sind:

- `up` aufwärts schwenken
- `down` abwärts schwenken
- `left` nach links schwenken
- `right` nach rechts schwenken
- `zoomin` hineinzoomen
- `zoomout` herauszoomen

  dem Kommando folgt jeweils die Geschwindigkeit im Bereich 0% bis 100%, z.B. `up=80`

- `preset` Festposition anfahren (verfügbar ab vimacc 2.2.8.7)

  dem Kommando folgt die Nummer der Festposition.

- `stop` Kamerabewegung anhalten

  mit dem Kommando `cmd=move;stop=1` werden alle Schlüsselwörter des Kommandos außer `preset` mit dem Wert O verschickt. Dieser Kommandostring entspricht daher dem folgenden Kommandostring:

  `cmd=move;up=0;left=0;down=0;right=0;zoomIn=0;zoomout=0`

Gültige Schlüsselwörter für den Parameter <`keyword`> des Kommandos `iris` sind:

- `start` manuelle Einstellung der Blende

  dem Kommando folgt jeweils die Geschwindigkeit im Bereich -100% bis100%, (`start` > 0 Blende öffnen, `start` < 0 Blende schließen), z.B. `start=-80`

- `auto=1` Wechsel in den Automatik-Modus

- `stop=1` Blendenbewegung anhalten

Gültige Schlüsselwörter für den Parameter <`keyword`> des Kommandos `focus` sind:

- `step` schrittweise fokussieren

  dem Kommando folgt die Anzahl der Schritte, die ausgeführt werden sollen.

- `start` Wechsel in den Automatik-Modus

  dem Kommando folgt die Geschwindigkeit im Bereich -100% bis 100%, z.B. `start=-70`

- `stop=1` Fokussierung stoppen

Beispiele:

- MMS→vimacc:
  `cmd=move;up=75;contextid=1234;source=Camera_0001`

  vimacc→MMS:
  `resp=move;up=75;contextid=1234;source=Camera_0001;answer=ok`

- MMS→vimacc:
  `cmd=move;up=20;left=45;down=20;contextid=1234;source=Camera_0001`

  vimacc→MMS:
  `resp=move;up=20;left=45;down=20;contextid=1234;source=Camera_0001;answer=ok`

- MMS→vimacc:
  `cmd=move;stop=1;contextid=1234;source=Camera_0001`

vimacc→MMS:
`resp=move;stop=1;contextid=1234;source=Camera_0001;answer=ok`

- MMS→vimacc:
  `cmd=iris;start=50;contextid=1234;source=Camera_0001`

  vimacc→MMS:
  `resp=iris;start=50;contextid=1234;source=Camera_0001;answer=ok`
- MMS→vimacc:
  `cmd=iris;stop=1;contextid=1234;source=Camera_0001`

  vimacc→MMS:
  `resp=iris;stop=1;contextid=1234;source=Camera_0001;answer=ok`

- MMS→vimacc:
  `cmd=focus;start=50;contextid=1234;source=Camera_0001`

  vimacc→MMS:
  `resp=focus;start=50;contextid=1234;source=Camera_0001;answer=ok`

- MMS→vimacc:
  `cmd=focus;stop=1;contextid=1234;source=Camera_0001`

  vimacc→MMS:
  `resp=focus;stop=1;contextid=1234;source=Camera_0001;answer=ok`

3.7.3.13 Presets einer Kamera konfigurieren

(verfügbar ab vimacc 2.2.8.7)

Kommando:
`cmd=<setpreset>;source=<KameraID>;preset=<PresetNo>;userdata=<text>;contextid=<text>`

Antwort:
`cmd=<setpreset>;source=<KameraID>;preset=<PresetNo>;userdata=<text>;contextid=<text>;answer=ok|failed`

Bei Steuerungsbefehlen zum Speichern eines Presets einer PTZ-Kamera muss immer im Parameter `source` die ID der entsprechenden Kamera übergeben werden.

Mit dem Parameter `preset` wird die Preset-Position in der Kamera angegeben, auf der gespeichert werden soll. In der Dokumentation der Kamera ist zu prüfen, welche Preset-Positionen verfügbar sind.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc®** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Hinweis:
Es erfolgt keine Prüfung, ob der Speicherplatz auf der Kamera verfügbar ist. Eine Speicherung auf einer Preset-Position löscht eine vorherige Konfiguration.

Beispiel:
MMS→vimacc:
`cmd=setpreset;source=Camera_0001;preset=1;userdate=1234;contextid=1234;`

vimacc→MMS:
`cmd=setpreset;source=Camera_0001;preset=1;userdate=1234;contextid=1234; answer=ok`

3.7.3.14 Abfragen der konfigurierten Kameras

Kommando:
```
cmd=getcameralist;metainfo=<0|1>;userdata=<text>
```

Antwort:
```
resp=getcameralist;metainfo=<0|1>;userdata=<text>;answer=ok,pa
rameterlist{\r\nname=<name#1>\;id=<id#1>\;metainfo=<metainfo#1
>>\r\n...name=<name#n>\;id=<id#n>\;metainfo=<metainfo#n>>\r\n}
```

Mit diesem Kommando kann die Liste der innerhalb des **vimacc®** Systems konfigurierten Kameras abgefragt werden.

Der Parameter `metainfo` bestimmt, ob in der Antwort die zu dem Gerät gespeicherten Metainformationen mit zurückgegeben werden sollen oder nicht.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Für jede konfigurierte Kamera wird innerhalb des Antworttextes eine Zeile in der Form
`name=Kamera-Name;id=Kamera-Id`
oder
`name=Kamera-Name;id=Kamera-Id;metainfo=Kamera-Metainfos`
zurückgegeben.
Beispiel:
MMS→vimacc:
```
cmd=getcameralist;metainfo=0;userdata=1234
```

vimacc→MMS:
```
resp=getcameralist;userdata=1234;answer=ok,parameterlist{\r\nn
ame=cameraName#1\;id=cameraId#1\r\n
name=cameraName#2\;id=cameraId#2\r\n}
```

3.7.3.15 Abfragen der verfügbaren Wiedergabe-Streams

Kommando:
```
cmd=getplaybacklist;mediatype=<video|audio>;userdata=<text>
```

Antwort:
```
resp=getplaybackList;mediatype=<video|audio>;userdata=<text>;a
nswer=ok|failed,parameterlist{\r\nid=<id#1>;name=<name#1>;medi
atype=<type>;definition_parameter=<text#1>;function=<function#
1>;merge=<merge#1>;privacy=<privacy#1>;associations=<associati
ons#1>\r\n … id=<id#n>;name=<name#n>;...\r\n}
```

Mit diesem Kommando kann die Liste der innerhalb des **vimacc®** Systems verfügbaren Wiedergabe-Streams abgefragt werden.

Der Parameter `mediatype` bestimmt, welcher Art von Streams abgefragt werden sollen. Der Wert `video` fragt nur Video-Streams ab, der Wert `audio` fragt die Audio-Streams ab. Sollen mehrere Mediatypen abgefragt werden, so sind die entsprechenden `mediatype` Werte mit einem Komma (',') voneinander zu trennen.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Für jeden verfügbaren Wiedergabe-Stream wird innerhalb des Antworttextes eine Zeile in der Form

```
id=<id>;name=<name>;mediatype=<type>;definition_parameter=<text>;function=<function>;merge=<mergeID>;privacy=<privacy>;associations=<associations>
```

zurückgegeben, wobei nicht zwingend immer alle Parameter vorhanden sein müssen.

Folgende Informationen werden zurückgegeben:

- `id`:
  ID zur eindeutigen Referenzierung des Wiedergabe-Streams.

- `name`:
  Der Name des Wiedergabe-Streams. Dies ist üblicherweise der Name der Kamera (bzw. des Devices), die den Stream aufgezeichnet hat.

- `mediatype`:
  Der Medientyp des Wiedergabe-Streams, also entweder `video` oder `audio`.

- `definition_parameter`:
  Dieser Parameter liefert systemspezifische Konfigurationsparameter zurück, die im Zuge der Konfiguration der Kamera (bzw. des Devices) angegeben wurden, die den Stream aufgezeichnet hat. Diese Werte werden von **vimacc®** automatisch in den Wiedergabe-Stream übertragen.

- `function`:
  Dieser Parameter liefert den systemspezifischen Konfigurationsparameter des Datenfeldes `function` zurück, der im Zuge der Konfiguration der Kamera (bzw. des Devices) angegeben wurden, die den Stream aufgezeichnet hat. Diese Werte werden von vimacc automatisch in den Wiedergabe-Stream übertragen.

- `merge`:
  Dieser Parameter liefert den systemspezifischen Konfigurationsparameter des Datenfeldes `merge` zurück, der im Zuge der Konfiguration der Kamera (bzw. des Devices) angegeben wurden, die den Stream aufgezeichnet hat. Diese Werte werden von **vimacc®** automatisch in den Wiedergabe-Stream übertragen.

  Das Datenfeld `merge` dient dazu, zusammengehörige Wiedergabespuren zu

kennzeichnen, so dass der *AccVimaccServer* beim Abruf eines Streams die zugehörigen Beiträge zusammenführen (`'mergen'`) kann.
Dies ist zum Beispiel dann sinnvoll, wenn für eine Kamera sowohl eine normale Archivaufzeichnung mit geringer Bildwiederholrate als auch eine Alarmaufzeichnung mit hoher Bildwiederholrate vorhanden ist. Der *AccVimaccServer* wird dann beim Abruf des Streams einer Kamera beide Spuren laden und zusammenführen und den Teil mit der hohen Bildwiederholrate bevorzugt wiedergeben.

- `privacy`:
  Dieser Parameter liefert den systemspezifischen Konfigurationsparameter des Datenfeldes `privacy` zurück, der im Zuge der Konfiguration der Kamera angegeben wurde, die den Stream aufgezeichnet hat. Diese Werte werden von **vimacc®** automatisch in den Wiedergabe-Stream übertragen und geben die für diese Kamera konfigurierten privaten Zonen zurück.

- `associations`:
  Dieser Parameter liefert den systemspezifischen Konfigurationsparameter des Datenfeldes `associations` zurück, der im Zuge der Konfiguration der Kamera (bzw. des Devices) angegeben wurde, die den Stream aufgezeichnet hat. Diese Werte werden von **vimacc®** automatisch in den Wiedergabe-Stream übertragen und geben die mit diesem Kanal assoziierten Kanäle zurück. Eine derartige Assoziation kann z.B. ein zu einem Videokanal zugeordneter Audiokanal sein.

Beispiel:
MMS→vimacc:

```
cmd=getplaybacklist;mediatype=video;userdata=1234
```

vimacc→MMS:

```
resp=getplaybacklist;mediatype=video;userdata=1234;answer=ok,p
arameterlist {
id=Showroom;mediatype=video;name=Showroom;function=PTZ,EMA
id=autobahn;mediatype=video;definition_parameter=playbackonly\
=true
id=cam0002;mediatype=video;definition_parameter=hidefromtopolo
gy\=1
id=cam0116_archiv;mediatype=video;name=Aussenhaut/Tor/Nord;fun
ction=FIX,Tor,Aussenhaut;merge=cam0116_prealarm
}
```

3.7.3.16 Abfragen Wiedergabe-Session eines Playback-Streams

Kommando:

```
cmd=getplaybacksessionsforplaybackid;playbackid=<id>;userdata=
<text>
```

Antwort:

```
resp=getplaybacksessionsforplaybackid;playbackid=<id>;userdata
=<text>;answer=ok|failed,parameterlist{\r\n
sessionid=<id#1>\;sessionname=<sessionname#1>\;controller=<con
troller#1>\;port=<port#1>\r\n …
sessionid=<id#1>\;sessionname=<sessionname#1>\;controller=<con
troller#1>\;port=<port#1>\r\n}
```

Mit diesem Kommando kann die Liste der Wiedergabe-Sessions eines **vimacc®** Wiedergabe-Streams abgefragt werden.

vimacc® ist in der Lage, Audio- und Videodaten auf mehreren Servern aufzuzeichnen. Beispielsweise wird in einem System mit redundanten Servern bei Ausfall eines Servers die Aufzeichnung automatisch auf dem redundanten Server weitergeführt. Dies hat zur Folge, dass für eine Wiedergabe eines aufgezeichneten Streams der Inhalt von zwei verschiedenen Servern geladen werden muss. Die Zuordnung der Teilstücke zu einem bestimmten Wiedergabe-Stream erfolgt dabei durch sogenannte Sessions, die durch eine ID, den Namen und den IP-Port des Aufzeichnungsservers eindeutig bestimmt werden.

Die Sessions eines bestimmten Wiedergabe-Streams können mit dem Kommando `getplaybacksessionsforplaybackid` abgerufen werden, wobei der Parameter `playbackid` die ID des Wiedergabe-Streams angibt und damit den wiederzugebenen Beitrag referenziert.

Die Liste der verfügbaren Wiedergabe-Streams selbst kann mit dem Kommando `getplaybacklist` abgerufen werden (siehe Kapitel 3.7.3.15).

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Für jede verfügbare Wiedergabe-Session wird innerhalb des Antworttextes eine Zeile in der Form

```
sessionid=<id>\;sessionname=<sessionname>\;controller=<control
ler>\;port=<port>
```

zurückgegeben, wobei nicht zwingend immer alle Parameter vorhanden sein müssen.

Folgende Informationen werden zurückgegeben:

- `sessionid`:
  ID zur eindeutigen Referenzierung der Wiedergabe-Session.

- `sessionname`:
  Der Name des Wiedergabe-Session.

- `controller`:
  Der Name des Aufzeichnungsservers, der diese Wiedergabe-Session bereitstellt.

- `port`:
  Der IP-Port, über den der zugehörige Stream vom Aufzeichnungsserver abgerufen werden kann.

Hinweis:
Dieses Kommando ist nur relevant, wenn ein Wiedergabe-Stream von einem Wiedergabe-Objekt abgerufen wird, das direkt per TCP/IP mit den **vimacc®** Aufzeichnungsservern kommuniziert und die Streams selbständig abruft. Dies ist beispielsweise bei einem autonomen **vimacc®** Video-Widget der Fall (Implementierungsbeispiel ist das Video-EWO bei WinCC OA).
Ein autonomes Video-Widget unterliegt nicht dem **vimacc®** Rechtemanagement, da es keine Verbindung zur zentralen **vimacc®** Konfiguration hat.

Innerhalb eines **vimacc®** Systems werden die Wiedergabe-Streams dagegen nur mit der ID des Streams referenziert und die **vimacc®** Objekte rufen selbständig die verschiedenen Teilstücke von den Aufzeichnungsservern ab.

Beispiel:
MMS→vimacc:
```
cmd=getplaybacksessionsforplaybackid;playbackid=cam0230;userda
ta=1234
```

vimacc→MMS:
```
resp=getplaybacksessionsforplaybackid;playbackid=cam0230;userd
ata=1234;answer=ok,parameterlist {
sessionid=114;sessionname=REC1_PREALARM;controller=laptop-
heinrich;port=9371
}
```

3.7.3.17 Abfragen der zeitlichen Grenzen eines Wiedergabe-Streams

Kommando:
```
cmd=getstreaminfo;playbackid=<playbackid>;userdata=<text>
```

Antwort:
```
resp=getstreaminfo;playbackid=<playbackid>;userdata=<text>;ans
wer=ok|failed;begintime=<utc timestamp iso 8601: yyyy-MM-
dd'T'hh:mm:ss.zzz>;endtime=<utc timestamp iso 8601: yyyy-MM-
dd'T'hh:mm:ss.zzz>
```

Mit diesem Kommando können die Grenzen eines Wiedergabe-Streams abgefragt werden. Hierbei werden die absoluten Zeitpunkte für den Anfang und das Ende des

Streams ermittelt. Sollte sich der aufgezeichnete Inhalt auf mehreren Aufzeichnungsservern befinden, so werden die Grenzen für den gesamten Inhalt ermittelt.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Der Parameter `playbackid` gibt die ID des Wiedergabe-Streams an, dessen Grenzen ermittelt werden sollen.

Die ermittelten Grenzen werden als UTC Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` zurückgeliefert.

Beispiel:
MMS→vimacc:
```
cmd=getstreaminfo;playbackid=cam0231_archiv;userdata=test
```

vimacc→MMS:
```
resp=getstreaminfo;playbackid=cam0231_archiv;userdata=test;answer=ok;begintime=2014-04-01T11:42:19.246;endtime=2014-04-01T11:52:26.196
```

3.7.3.18 Abfragen der Timeline eines Wiedergabe-Streams

Kommando:

```
cmd=getstreamtimeline;playbackid=<playbackid>;userdata=<text>
```

Antwort:

```
resp=getstreamtimeline;playbackid=<playbackid>;userdata=<text>
;answer=ok|failed,parameterlist{\r\n
begintime=<timestamp>;endtime=<timestamp\r\n
…
begintime=<timestamp>;endtime=<timestamp\r\n
begintime_merged=<timestamp>;endtime_merged=<timestamp\r\n
…
begintime_merged=<timestamp>;endtime_merged=<timestamp
\r\n}
```

Mit diesem Kommando kann die gesamte Timeline eines Wiedergabe-Streams abgefragt werden. Die sogenannte Timeline kann aus mehreren zusammenhängenden Zeitbereichen bestehen und zwischen den Zeitbereichen kann es unterschiedlich große Pausen geben.

Dieses Kommando unterscheidet sich von dem Kommando `getstreaminfo` (siehe Kapitel 3.7.3.17) insofern, dass hier alle zusammenhängenden Bereiche des Wiedergabe-Streams ermittelt werden und nicht nur die Grenzen des gesamten Zeitbereiches zurückgegeben werden.

Sollte sich der aufgezeichnete Inhalt auf mehreren Aufzeichnungsservern befinden, so werden die Zeitbereiche auf allen Servern ermittelt und alle gefundenen Zeitbereiche zurückgegeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Der Parameter `playbackid` gibt die ID des Wiedergabe-Streams an, dessen Grenzen ermittelt werden sollen.

Die ermittelten Grenzen werden als UTC (Coordinated Universal Time) Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` zurückgeliefert.

Für jeden zusammenhängenden Zeitabschnitt wird innerhalb des Antworttextes eine Zeile in der Form

```
begintime=<timestamp>\;endtime=<timestamp>\r\n
```

und/oder in der Form

```
begintime_merged=<timestamp>\;endtime_merged=<timestamp>\r\n
```

zurückgegeben.

Die Bereiche, die durch `begintime` und `endtime` festgelegt sind, beschreiben Bereiche, in denen aufgezeichnetes Material auf den Aufzeichnungsservern gefunden wurde.

Die Bereiche, die durch `begintime_merged` und `endtime_merged` festgelegt sind, beschreiben dagegen Bereiche, in denen aufgezeichnetes Material in weiteren Aufzeichnungsspuren gefunden wurde. Dies betrifft z.B. Videomaterial, das auf Grund eines Alarmereignisses in der zugehörigen Alarmaufzeichnungsspur gespeichert wurde.

Da üblicherweise bei der Wiedergabe eines Streams in einer Anwendung mir Benutzerschnittstelle nur eine Timeline angezeigt wird, können durch die so ermittelten Informationen Bereiche einer Archivaufzeichnung von denen einer Alarmaufzeichnung z.B. durch eine andere Farbe unterschieden werden.

Hinweis:
Dieses Kommando belastet die Aufzeichnungsserver stärker als das Kommando `getstreaminfo` (siehe Kapitel 3.7.3.17). Sollen nur die äußeren Grenzen eines Streams ermittelt werden, sollte daher das Kommando `getstreaminfo` verwendet werden.

Beispiel:
MMS→vimacc:
```
cmd=getstreamtimeline;playbackid=cam0231_archiv;userdata=test
```

vimacc→MMS:
```
resp=getstreamtimeline;playbackid=cam0231_archiv;userdata=test
;answer=ok;parameterlist
{\r\nbegintime=2014-04-01T11:42:19.246\;endtime=2014-04-
01T12:49:38.727\r\n
begintime_merged=2014-04-01T11:50:51.197\;endtime=2014-04-
01T11:55:26.196\r\n}
```

3.7.3.19 Abfragen der konfigurierten Monitore

Kommando:
```
cmd=getmonitorlist;metainfo=<0|1>;userdata=<text>
```

Antwort:
```
resp=getmonitorlist;metainfo=<0|1>;userdata=<text>;answer=ok,p
arameterlist{\r\nname=<name#1>;id=<id#1><;metainfo=<metainfo#1
>>\r\n...name=<name#n>;id=<id#n><;metainfo=<metainfo#n>>\r\n}
```

Mit diesem Kommando kann die Liste der innerhalb des **vimacc®** Systems konfigurierten Display-Instanzen abgefragt werden.

Der Parameter `metainfo` bestimmt, ob in der Antwort die zu dem Gerät gespeicherten Metainformationen mit zurückgegeben werden sollen oder nicht.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Für jede konfigurierte Display-Instanz wird innerhalb des Antworttextes eine Zeile in der Form
`name=Display-Name;id=Display-Id`
oder
`name=Display-Name;id=Display-Id;metainfo=Display-Metainfos`
zurückgegeben.

Beispiel:
MMS→vimacc:
```
cmd=getmonitorlist;metainfo=0;userdata=1234
```

vimacc→MMS:
```
resp=getmonitorlist;metainfo=0;userdata=1234;answer=ok,paramet
erlist{\r\nname=displayName#1\;id=displayId#1\r\nname=displayN
ame#2\;id=displayId#2\r\n}
```

3.7.3.20 Abfragen der konfigurierten Workstation-Instanzen

Kommando:
```
cmd=getworkstationlist;metainfo=0|1;userdata=<text>
```

Antwort:
```
resp=getworkstationlist;metainfo=<0|1>;userdata=<text>;answer=
ok,parameterlist{\r\nname=<name#1>;id=<id#1>\;metainfo=<metain
fo#1>>\r\n...name=<name#n>\;id=<id#n>\;metainfo=<metainfo#n>>\
r\n}
```

Mit diesem Kommando kann die Liste der innerhalb des **vimacc®** Systems konfigurierten Workstation-Instanzen abgefragt werden.

Der Parameter `metainfo` bestimmt, ob in der Antwort die zu dem Gerät gespeicherten Metainformationen mit zurückgegeben werden sollen oder nicht.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Für jede konfigurierten Workstation-Instanz wird innerhalb des Antworttextes eine Zeile in der Form
`name=Workstation-Name\;id=Workstation-Id`
oder
`name=Workstation-Name\;id=Workstation-Id\;metainfo=Workstation-Metainfo`
zurückgegeben.

Beispiel:
MMS→vimacc:
```
cmd=getworkstationlist;metainfo=0;userdata=1234
```

vimacc→MMS:
```
resp=getworkstationlist;metainfo=0;userdata=1234;answer=okpara
meterlist{\r\nname=AP#1\;id=workstationId#1\r\nname=
AP#2\;id=workstationId#2\r\n}
```

3.7.3.21 Abfragen der konfigurierten Szenarien

Kommando:
```
cmd=getscenariolist;userdata=<text>
```

Antwort:
```
resp=getscenariolist;userdata=<text>;answer=ok,parameterlist{\
r\nname=<name#1>\;id=<id#1>\r\n...name=<name#n>\;<id>=<id#n>\r
\n}
```

Mit diesem Kommando kann die Liste der innerhalb des **vimacc®** Systems konfigurierten Szenarien abgefragt werden.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Für jedes konfigurierte Szenario wird innerhalb des Antworttextes eine Zeile in der Form `name=Szenario-Name;id=Szenario-Id` zurückgegeben.

Beispiel:
MMS→vimacc:
```
cmd=getscenariolist;userdata=1234
```

vimacc→MMS:
```
resp=getscenariolist;userdata=1234;answer=ok,parameterlist{\r\
nname=Szenario#1\;id=SzenarioId#1\r\nname=Szenario#2\;id=Szena
rioId#2\r\n}
```

3.7.3.22 Abfragen der konfigurierten Sequenzen

(verfügbar ab vimacc 2.2.8.7)

Kommando:
```
cmd=getsequencelist;userdata=<text>
```

Antwort:
```
resp=getsequencelist;userdata=<text>;answer=ok,parameterlist{\
r\nname=<name#1>\;id=<id#1>\r\n...name=<name#n>\;<id>=<id#n>\r
\n}
```

Mit diesem Kommando kann die Liste der innerhalb des **vimacc®** Systems konfigurierten Sequenzen abgefragt werden.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Für jede konfigurierte Sequenz wird innerhalb des Antworttextes eine Zeile in der Form `name=Sequenz-Name;id=Sequenz-Id` zurückgegeben.

Beispiel:
MMS→vimacc:
```
cmd=getsequencelist;userdata=1234
```

vimacc→MMS:
```
resp=getsequencelist;userdata=1234;answer=ok,parameterlist{\r\
nname=Sequenz#1\;id=SequenzId#1\r\nname=Sequenz#2\;id=SequenzI
d#2\r\n}
```

3.7.3.23 Anfordern von Status-Informationen von vimacc Devices

Kommando:
```
cmd=subscribedevicestatus;function=<text>;userdata=<text>
```

Antwort:
```
resp=subscribedevicestatus;function=<text>;userdata=<text>;ans
wer=ok|failed
```

Statusmeldung:
```
resp=devicestatus;userdata=<text>;deviceid=<deviceid>;property
=<property>;content=<status text>
```
Mit diesem Kommando können Änderungen der Statusinformationen von **vimacc®** Gerätetypen abonniert werden.

Innerhalb eines **vimacc®** Systems werden Geräte (=Devices) immer mit einem oder mehreren Funktionsbezeichnern parametriert.
Bezeichner können z.B. sein:

- `PTZ` oder `FIX` für PTZ-, bzw. Fix-Kameras,
- `HID` für Human-Interface-Devices,
- `SERVER` für **vimacc®** Streaming-Server
- `WORKSTATION` für **vimacc®** Workstation Instanzen
- usw.

Der Parameter `function` in diesem Kommando spezifiziert, von welchem Gerätetyp die Statusinformationen angefordert werden.
Sollen mehrere Gerätetypen abonniert werden, so sind die entsprechenden `function` Werte mit einem Komma (',') voneinander zu trennen.
Jeder erneute Aufruf von `subscribedevicestatus` überschreibt frühere Aufrufe dieses Kommandos.

Der Eintrag `function=none` beendet die Übertragung von Statusinformationen.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Nach dem erfolgreichen Absetzen dieses Kommandos werden einmalig die Zustandsinformationen aller entsprechenden Gerätetypen mit der Statusmeldung `resp=devicestatus` an das MMS gesendet.
Weitere Statusmeldungen `resp=devicestatus` werden dann nur noch an das MMS gesendet, wenn sich der Status eines der abonnierten **vimacc®** Devices geändert hat.

Hinweis:
Je nach Anzahl der konfigurierten **vimacc®** Devices kann es innerhalb eines **vimacc®** Systems zu einer sehr großen Anzahl von Statusänderungen kommen. Alle Statusinformation zu der steuernden Instanz zu übertragen, könnte zu einer sehr hohen Netzwerkauslastung führen.
Damit diese Last nicht ungewollt erzeugt wird, ist bewusst auf den Eintrag `function=all` verzichtet worden, und es müssen alle zu abonnierenden Gerätetypen explizit genannt werden.

Hinweis:
Dieses Kommando muss mit Bedacht eingesetzt werden. **vimacc®** ist ein sehr vitales System, bei dem jedes Device unmittelbar die Zustandsänderungen veröffentlicht. Bei Systemen mit einer großen Anzahl von Geräten (z.B. einigen hundert Videokameras) kann ein unbedachtes Abonnieren von Statusmeldungen zu einer sehr hohen Anzahl von Meldungen vom Control-Interface führen. In diesem Fall sollte durch die richtige Wahl des Parameters `function` die Anzahl der abonnierten Meldungen eingeschränkt werden.
Beispiel:

- MMS→vimacc:
  `cmd=subscribedevicestatus;function=PTZ,FIX;userdata=789`

  vimacc→MMS:
  `resp=subscribedevicestatus;function=PTZ,FIX;userdata=789;answer=ok`

  vimacc→MMS (spontane Zustandsänderung):
  `resp=devicestatus;userdata=789;deviceid=1001;property=streaming;content=video\=ok`

3.7.3.23.1 Status-Informationen von vimacc Devices

Informationen über den Status von **vimacc®** Devices werden üblicherweise in mehrere Bereiche unterteilt, die jeweils in dem Parameter `property=<property>` zurückgegeben werden.
Der zurückgegebene Parameter `content=<status text>` enthält dann die eigentliche Status-Information in Textform.

Status-Informationen von **vimacc®** Devices werden u.a. über die Bereiche `control`, `streaming`, `recording`, `function` und `availability` abgebildet, wobei nicht

bei jeder Geräteklasse alle Bereiche belegt werden.
Bei einer **vimacc®** Workstation-Instanz (`function=WORKSTATION`) wird beispielsweise der Eintrag `recording` immer den Inhalt `state=off` enthalten, denn es werden nie Aufzeichnungsverbindungen von einer Workstation-Instanz zu den Recording-Servern aufgebaut.
Mögliche Parameter sind:

- `control`:
  Zeigt den Zustand der Gerätesteuerung an.
- `streaming`:
  Zeigt den Zustand der Streaming-Verbindungen an.
- `recording`:
  Zeigt den Zustand von Aufzeichnungs-Verbindungen an.
- `function`:
  Zeigt an, welche Funktionen das Gerät unterstützt.
- `monitoring`:
  Enthält die überwachten Geräteinformationen, wie z.B. verwendete Bandbreite ('`bandwidth`') oder Bildwiederholrate ('`framerate`').
- `availability`:
  Zeigt an, ob ein Gerät verfügbar ist oder nicht. Verfügbarkeit bedeutet in diesem Zusammenhang, ob das Gerät generell in der Systemdatenbank gefunden wurde und von der zugehörigen **vimacc®** Software-Komponente initialisiert und im **vimacc®** System registriert wurde. Ob auf das Gerät tatsächlich ordnungsgemäß zugegriffen werden kann, muss aus den zusätzlichen Status-Informationen wie etwa `control`, `streaming` oder `recording` abgelesen werden (siehe oben).

3.7.3.23.2 Gemeldete Status-Informationen von Kameras

An dieser Stelle sollen beispielhaft die Status-Informationen bzgl. einer Kamera angeben werden.

- `control`:
  Konnte die Verbindung zu der steuernden Instanz aufgebaut werden, so lautet die zugehörige Statusmeldung für eine bestimmte Kamera:
  `resp=devicestatus;userdata=5678;deviceid=cam0230;property=control;content=state\=ok`

- `streaming`:
  Konnte die Streaming Verbindung aufgebaut werden, so lautet die zugehörige Statusmeldung für eine bestimmte Kamera:
  `resp=devicestatus;userdata=5678;deviceid=cam0230;property=streaming;content=video\=ok`

  Wird die Streaming Verbindung z.B. auf Grund eines Netzwerkfehlers getrennt, so lautet die zugehörige Statusmeldung:
  `resp=devicestatus;userdata=5678;deviceid=cam0230;property`

```
=streaming;content=video\=disconnected
```

Anschließend wird **vimacc®** periodisch versuchen, die Verbindung zu der Kamera erneut aufzubauen, so dass die folgenden Meldungen im Wechsel zu erwarten sind:

```
resp=devicestatus;userdata=5678;deviceid=cam0230_prealarm
;property=streaming;content=video\=no rtp

resp=devicestatus;userdata=5678;deviceid=cam0230_prealarm
;property=streaming;content=video\=pending
```

Sobald die Verbindung wieder aufgebaut werden konnte, ist die folgende Statusmeldung zu erwarten:

```
resp=devicestatus;userdata=5678;deviceid=cam0230;property
=streaming;content=video\=ok
```

- `recording`:
  Die zugehörige Statusmeldung lautet für eine bestimmte Kamera:
  ```
  resp=devicestatus;userdata=5678;deviceid=cam0230;property
  =recording;content=REC1_PREALARM@localhost\=start prealar
  m 0 3600
  ```

- `function`:
  Die zugehörige Statusmeldung lautet für eine bestimmte Kamera:
  ```
  resp=devicestatus;userdata=5678;deviceid=cam0230;property
  =function;content=PTZ,EMA
  ```

- `availability`:
  Die zugehörige Statusmeldung lautet für eine verfügbare Kamera:
  ```
  resp=devicestatus;userdata=5678;deviceid=cam0230;property
  =availability;content=ok
  ```
  Für eine Kamera, die nicht mehr verfügbar ist, lautet die Meldung:
  ```
  resp=devicestatus;userdata=5678;deviceid=cam0230;property
  =availability;content=not ok
  ```

3.7.3.24 Anfordern von Ereignis-Informationen von vimacc Devices

Kommando:

```
cmd=subscribeevents;function=<text>;userdata=<text>
```

Antwort:

```
resp=subscribeevents;function=<text>;userdata=<text>;answer=ok
|failed
```

Ereignismeldung:

```
resp=event;deviceid=<deviceid>;property=<property>;content=<ev
ent text>
```

Mit diesem Kommando können Alarme bzw. Ereignisse von **vimacc®** Devices abonniert werden.
Sobald ein Event eines der abonnierten **vimacc®** Devices erkannt wurde, wird es als Meldung `resp=event` an das MMS weitergeleitet.

Der Parameter `function` in diesem Kommando spezifiziert, von welchem Gerätetyp die Eventinformationen angefordert werden (→ siehe Befehl `subscribedevicestatus`).
Sollen mehrere Gerätetypen abonniert werden, so sind die entsprechenden `function` Werte mit einem Komma (',') voneinander zu trennen.
Jeder erneute Aufruf von `subscribeevents` überschreibt frühere Aufrufe dieses Kommandos.

Der Eintrag `function=all` abonniert die Eventmeldungen aller Geräte innerhalb des **vimacc** Systems.

Der Eintrag `function=none` beendet die Übertagung von Eventinformationen.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:

- MMS→vimacc:
  `cmd=subscribeevents;function=PTZ,FIX;userdata=7`

  vimacc→MMS:
  `cmd=subscribeevents;function=PTZ,FIX;userdata=7;answer=ok`

  vimacc→MMS (spontanes Ereignis):
  `resp=event;userdata=7;deviceid=4001;property=deviceup;content=true`

3.7.3.24.1 Gemeldete Ereignis-Informationen von vimacc Devices

Ereignis-Informationen von **vimacc®** Devices werden nicht normiert, da diese für die verschiedenen Geräteklassen zu unterschiedlich ausfallen können.

Wie bei den Statusinformationen werden auch hier die verschiedenen Bereiche in dem Parameter `property=<property>` zurückgegeben werden.
Der zurückgegebene Parameter `content=<event text>` enthält dann die eigentliche Ereignis-Information in Textform.

3.7.3.24.2 Ereignis-Informationen von Workstation-Instanzen

An dieser Stelle sollen beispielhaft einige mögliche Ereignisse von **vimacc®** Workstation-Instanzen aufgelistet werden.

`deviceup`:
Die Workstation-Instanz wurde gestartet. Die zugehörige Meldung lautet:
```
resp=event;userdata=5678;deviceid=401;property=deviceup;conten
t=true
```

`playback`:
Die Workstation-Instanz meldet über dieses Event den Status bzgl. der Wiedergabe eines oder mehrerer Streams. Hieran ist zu erkennen, ob eine Wiedergabe gestartet oder gestoppt wurde, welchen zeitlichen Bereich alle aufgeschalteten Wiedergabe-Streams überdecken und welches die aktuelle Wiedergabeposition ist. Da immer alle Wiedergabe-Streams zeitlich synchronisiert werden, gelten diese Angaben immer für alle aufgeschalteten Wiedergabe-Streams.
Die zugehörige Meldung lautet:
```
resp=event;userdata=5678;deviceid=401;property=playback;conten
t=streaming 100 3605364223097 3605356915780 3605412483511
```

`dialogs/VD<dialognummer>`
Die Workstation-Instanz meldet über dieses Event, welcher Stream in dem entsprechenden Videodialog aufgeschaltet ist, ob es sich um einen Live- oder einen Wiedergabe-Stream handelt und in welchem Status sich dieser Stream befindet.
Die zugehörige Meldung lautet:
```
resp=event;userdata=5678;deviceid=401;property=dialogs/VD1;con
tent=live streaming cam0231_prealarm
```

`alarmhandling`:
Zeigt Informationen über die Behandlung von Alarmen an einer Workstation-Instanz an. Der Parameter `status` innerhalb von `<event text>` gibt Auskunft über die Art der Alarmbehanldung.

Wurde z.B. ein Alarm an einer **vimacc®** Workstation erzeugt (von dem Benutzer oder durch die Aufschaltung eines Alarm-Szenarios (→Befehl `showscenario`), so lautet die zugehörige Meldung:
```
resp=event;userdata=5678;deviceid=401;property=alarmhandling;c
ontent=status\\\=created\\\;alarmid\\\=APLaptop-Heinrich 2013-
07-09 10:59:07 Uhr\\\;timestamputc\\\=2013-07-09T08:59:07.665
```

Ein Alarm, der von einem Benutzer an einer **vimacc®** Workstation angenommen wurde, führt zu folgender Meldung:
```
resp=event;userdata=5678;deviceid=401;property=alarmhandling;c
ontent=status\\\=accepted\\\;alarmid\\\=1234\\\;timestamputc\\
\=2013-07-09T09:48:58.866
```

Ein Alarm, der von einem Benutzer an einer **vimacc** Workstation beendet wurde, führt zu folgender Meldung:
```
resp=event;userdata=5678;deviceid=401;property=alarmhandling;c
ontent=status\\\=finished\\\;alarmid\\\=APLaptop-Heinrich
2013-07-09 10:59:07 Uhr\\\;timestamputc\\\=2013-07-
09T09:36:54.406
```

3.7.3.25 Anfordern von Status-Informationen von Wiedergabe-Streams

Kommando:
`cmd=subscribeplaybackstatus;mediatype=<video|audio|all>;userdata=<text>`

Antwort:
`resp=subscribeplaybackstatus;mediatype=<video|audio|all>;userdata=<text>;answer=ok|failed`

Statusmeldung:
`resp=playbackstatus;playbackid=<playbackid>;property=<property>;content=<status text>`

Mit diesem Kommando können Änderungen der Statusinformationen von **vimacc®** Wiedergabe-Streams abonniert werden.

Der Parameter `mediatype` in diesem Kommando spezifiziert, von welchem Media Typ die Statusinformationen angefordert werden. Innerhalb eines **vimacc®** Systems stehen die Mediatypen `video` und `audio` zur Verfügung.

Sollen mehrere Mediatypen abonniert werden, so sind die entsprechenden Werte mit einem Komma (',') voneinander zu trennen.
Jeder erneute Aufruf von `subscribeplaybackstatus` überschreibt frühere Aufrufe dieses Kommandos.

Der Eintrag `mediatype=none` beendet die Übertragung von Statusinformationen.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Nach dem erfolgreichen Absetzen dieses Kommandos werden einmalig die Zustandsinformationen aller entsprechenden Mediatypen mit der Statusmeldung `resp=playbackstatus` an das MMS gesendet.
Weitere Statusmeldungen `resp=playbackstatus` werden dann nur noch an das MMS gesendet, wenn sich der Status eines der abonnierten **vimacc®** Wiedergabe-Streams geändert hat.

Hinweis:
Im Zuge einer Redundanzumschaltung ändert sich beispielsweise der Eintrag für die zugehörigen Wiedergabe-Sessions, so dass diese Änderung durch das Status-Property `playbacksessions` mit dem Inhalt `changed` signalisiert wird. In diesem Fall muss die Liste der zugehörigen Wiedergabe-Sessions erneut durch das Kommando `getplaybacksessionsforplaybackid` (siehe Kapitel 3.7.3.16) abgerufen werden.

Beispiel:

- MMS→vimacc:
  `cmd=subscribeplaybackstatus;mediatype=video;userdata=789`

  vimacc→MMS:
  `resp=subscribeplaybackstatus;mediatype=video;userdata=789;answer=ok`

  vimacc→MMS (spontane Zustandsänderung):
  `resp=playbackstatus;userdata=789;playbackid=tankstelle;property=streaming;content=video\=ok`

3.7.3.25.1 Status-Informationen von Wiedergabe-Streams

Informationen über den Status von **vimacc®** Wiedergabe-Streams werden üblicherweise in mehrere Bereiche unterteilt, die jeweils in dem Parameter `property=<property>` zurückgegeben werden.
Der zurückgegebene Parameter `content=<status text>` enthält dann die eigentliche Status-Information in Textform.

Status-Informationen von **vimacc®** Wiedergabe-Streams werden u.a. über die Bereiche `control`, `streaming`, `recording`, `function` und `availability` abgebildet, wobei nicht bei jeder Geräteklasse alle Bereiche belegt werden.
Mögliche Parameter sind:

- `control`:
  Zeigt den Zustand der Gerätesteuerung an.
- `streaming`:
  Zeigt den Zustand der Streaming-Verbindungen an.
- `recording`:
  Zeigt den Zustand von Aufzeichnungs-Verbindungen an.
- `function`:
  Zeigt an, welche Funktionen das Gerät unterstützt.
- `availability`:
  Zeigt an, ob ein Wiedergabe-Stream verfügbar ist oder nicht.
- `playbacksessions`:
  Enthält dieser Parameter den Statustext `changed`, so wurde von den Aufzeichnungsservern die Liste der Wiedergabe-Sessions eines Streams modifiziert.

3.7.3.26 Anfordern von Status-Informationen des vimacc Systems

Kommando:
`cmd=subscribesystemstatus;userdata=<text>;activate=<0|1>`

Antwort:
`resp=subscribesystemstatus;userdata=<text>;activate=<0|1>;answer=ok|failed`

Statusmeldung:
`resp=systemstatus;userdata=<text>;property=<property>;content=<status text>`

Mit diesem Kommando können Änderungen der Statusinformationen des **vimacc** Systems abonniert werden.
Systeminformationen sind beispielsweise:

- Informationen über nicht zu erreichende **vimacc®** Rechner,
- Informationen über nicht zu erreichende **vimacc®** Dienste,
- Informationen über ausgefallene Kameras,
- Informationen über fehlerhafte oder ungenügende Lizenzen,
- usw.

Der Parameter `activate` dient dazu, die Statusinformationen zu abonnieren oder bereits abonnierte Statusinformationen wieder zu deaktivieren.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Nach dem erfolgreichen Absetzen dieses Kommandos werden einmalig die Zustandsinformationen des **vimacc®**Systems mit der Statusmeldung `resp=systemstatus` an das MMS gesendet.
Weitere Statusmeldungen `resp=systemstatus` werden dann nur noch an das MMS gesendet, wenn sich der Systemzustand geändert hat.

Beispiel:

- MMS→vimacc:
  `cmd=subscribesystemstatus;userdata=789;activate=1`

  vimacc→MMS:
  `resp=subscribesystemstatus;userdata=789;activate=1;answer=ok`

  vimacc→MMS (spontanes Ereignis):
  `resp=systemstatus;userdata=789;property=devices;content=state\=error\;source\=cam2001,cam2002"`

3.7.3.26.1 Gemeldete Status-Informationen des vimacc Systems

Informationen über den Zustand des **vimacc®** Systems werden ebenfalls in mehrere Bereiche unterteilt, die jeweils in dem Parameter `property=<property>` zurückgegeben werden.
Der zurückgegebene Parameter `content=<status text>` enthält dann die eigentliche Status-Information in Textform.

Folgende Informationen werden zurückgegeben:

- `availability:`
  Zeigt an, ob der Gesamtzustand des **vimacc®** Systems ermittelt werden konnte.

Stehen die relevanten Informationen nicht zur Verfügung, wird die folgende Meldung gesendet:

```
resp=systemstatus;userdata=1234;property=availability;content=not ok
```

Sobald die relevanten Informationen verfügbar sind, wird die folgende Meldung gesendet:

```
resp=systemstatus;userdata=1234;property=availability;content=ok
```

- `system`:
  Zeigt den Gesamtzustand des **vimacc** Systems an. Liegt kein Fehler vor, so lautet die zugehörige Meldung:

  ```
  resp=systemstatus;userdata=1234;property=system;content=state\=ok
  ```

  Wurde ein Fehler erkannt, so werden im Parameter `source` die Bereiche aufgelistet, in denen eine Fehler erkannt wurde. Die Liste der Geräte, die einen Fehler gemeldet haben, wird daraufhin ebenfalls über eine Statusmeldung mit dem entsprechenden Parameter `property` übermittelt.

  Das folgenden Beispiel zeigt etwa die Meldung, wenn in den Bereichen `services` und `devices` Fehler erkannt worden sind:

  ```
  resp=systemstatus;userdata=1234;property=system;content=state\=error\;source\=services,devices
  ```

- `devices`:
  Zeigt den Gesamtzustand von **vimacc®** Devices wie etwa Kamera-Devices, HID-Controller (z.B. für Joysticks) an. Workstation- und Display-Instanzen werden hier nicht erfasst, sondern werden jeweils in getrennten Meldungen übermittelt.
  Fehlerhafte Devices werden in einer komma-separierten Liste innerhalb des Parameters `source` mit ausgegeben.

  Eine entsprechende Meldung lautet z.B.:

  ```
  resp=systemstatus;userdata=1234;property=devices;content=state\=error\;source\=4000,cam0231b_archiv,cam0231b_prealarm,cam0232_archiv,cam0232_prealarm
  ```

- `displays`:
  Zeigt den Gesamtzustand aller konfigurierten **vimacc®** Display-Instanzen an. Fehlerhafte Display-Instanzen werden in einer komma-separierten Liste innerhalb des Parameters `source` mit ausgegeben.

  Eine entsprechende Meldung lautet z.B.:

  ```
  resp=systemstatus;userdata=1234;property=displays;content=state\=ok
  ```

- `workstations`:
  Zeigt den Gesamtzustand aller konfigurierten **vimacc®** Workstation-Instanzen an.
  Fehlerhafte Workstation-Instanzen werden in einer komma-separierten Liste innerhalb des Parameters `source` mit ausgegeben.

  Eine entsprechende Meldung lautet z.B.:
  ```
  resp=systemstatus;userdata=1234;property=workstations;con
  tent=state\=ok
  ```

  Sollte keine der konfigurierten **vimacc®** Workstation-Instanzen gestartet sein, bzw. wird die letzte der konfigurierten **vimacc®** Workstation-Instanzen beendet, so wird die folgende Fehlermeldung erzeugt:

  ```
  resp=systemstatus;userdata=1234;property=workstations;con
  tent=state\=error\;errcause\=no workstation available\;so
  urce\=401
  ```

- `hosts`:
  Zeigt den Gesamtzustand aller konfigurierten **vimacc®** Controller-Instanzen an. Fehlerhafte Controller-Instanzen werden in einer komma-separierten Liste innerhalb des Parameters `source` mit ausgegeben.

  Eine entsprechende Meldung lautet z.B.:
  ```
  resp=systemstatus;userdata=s;property=hosts;content=state
  \=ok
  ```

- `services`:
  Zeigt den Gesamtzustand aller konfigurierten **vimacc®** Services an. Unter **vimacc®** Services werden die bereitgestellten Funktionalitäten bestimmter **vimacc®** Module, wie etwa dem *AccVimaccEventManager* oder dem *AccVimaccControlInterface* verstanden .
  Fehlerhafte Services werden in einer komma-separierten Liste innerhalb des Parameters `source` mit ausgegeben.

  Eine entsprechende Meldung lautet z.B.:
  ```
  resp=systemstatus;userdata=s;property=services;content=st
  ate\=ok
  ```

- `licence`:
  Zeigt an, ob die Lizenzprüfung Fehler bzgl. der eingesetzten **vimacc®** Module oder dem verwendeten Mengengerüst (Eingabe- und Ausgabekanäle) festgestellt hat.
  Eine entsprechende Meldung lautet z.B.:
  ```
  resp=systemstatus;userdata=s;property=licence;content=sta
  te\=ok
  ```

3.7.3.27 Anfordern von Zustands-Informationen über die vimacc Konfigurationsserver

Kommando:
`cmd=subscribeconfigserverstatus;userdata=<text>;activate=<0|1>`

Antwort:
`resp=subscribeconfigserverstatus;userdata=<text>;activate=<0|1>;answer=ok|failed`

Statusmeldung:
`resp=configserverstatus;property=<property>;content=<status text>`

vimacc® kann als verteiltes System betrieben werden, bei dem die zentralen Komponenten aus Gründen höhere Ausfallsicherheit auf verschiedenen Rechnern betrieben werden können. Der zentrale Datenbankdienst für die gesamte Konfiguration eines **vimacc** Systems, dessen Steuerung und all seine Systemzustände (der sogenannte **vimacc** Konfigurationsserver) wird in einem solchen System auf zwei verschiedenen Rechnern betrieben, wobei eine der Komponenten die führende Rolle übernimmt und der redundante Partner im Hot-Standby Betrieb betrieben wird.

Mit dem Kommando `subscribeconfigserverstatus` können Status-informationen über den aktiven Konfigurationsserver abonniert werden.

Der Parameter `activate` dient dazu, die Statusinformationen zu abonnieren oder bereits abonnierte Statusinformationen wieder zu deaktivieren.

Nach dem erfolgreichen Absetzen dieses Kommandos werden einmalig die Zustandsinformationen des aktiven Konfigurationsservers mit der Statusmeldung `resp=configserverstatus` an das MMS gesendet.
Weitere Statusmeldungen `resp=devicestatus` werden dann nur noch im Falle einer Änderung der entsprechenden Werte an das MMS gesendet.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.
Beispiel:
MMS→vimacc:
`cmd=subscribeconfigserverstatus;userdata=786;activate=1`

vimacc→MMS:
`resp=subscribeconfigserverstatus;userdata=786;activate=1;answer=ok`

vimacc→MMS (spontanes Ereignis):
`resp=configserverstatus;userdata=786;property=hostname;content=Server#1`

3.7.3.27.1 Gemeldete Zustands-Informationen der vimacc Konfigurationsserver

Informationen über den Zustand der **vimacc®** Konfigurationsserver werden ebenfalls in mehrere Bereiche unterteilt, die jeweils in dem Parameter `property=<property>` zurückgegeben werden.
Der zurückgegebene Parameter `content=<status text>` enthält dann die eigentliche Status-Information in Textform.

Folgende Informationen werden zurückgegeben:

- `hostname`:
  Liefert den Namen des aktiven Konfigurastionsservers.
  Die zugehörige Meldung lautet:
  `resp=configserverstatus;userdata=1234;property=hostname;content=laptop-heinrich`

- `redundancyState`:
  Liefert den aktuellen Redundanzstatus.

  Die zugehörige Meldung lautet:
  `resp=configserverstatus;userdata=1234;property=redundancyState;content=RC0`

  `RC0` bedeutet, dass beide Konfigurationsserver aktiv sind und dass die **vimacc®** Konfigurationsdatenbank `RC0` geladen ist. Der aktuelle Konfigurations-Master wird in dem Parameter `hostname` zurückgegeben wird (s.o.).

  `RC1` bedeutet, dass ein Konfigurationsserver ausgefallen ist und dass die **vimacc®** Konfigurationsdatenbank `RC1` geladen ist. Der aktuelle Konfigurations-Master wird in dem Parameter `hostname` zurückgegeben wird (s.o.).

  `RC2` bedeutet, dass ein Konfigurationsserver ausgefallen ist und dass die **vimacc®** Konfigurationsdatenbank `RC2` geladen ist. Der aktuelle Konfigurations-Master wird in dem Parameter `hostname` zurückgegeben wird (s.o.).

3.7.3.28 Anfordern von Status-Informationen von vimacc Hostrechnern

Kommando:
`cmd=subscribehoststatus;hostnames=<text>;userdata=<text>;activate=<0|1>`

Antwort:
`resp=subscribehoststatus;hostnames=<text>;userdata=<text>;activate=<0|1>;answer=ok|failed`

Statusmeldung:
`resp=hoststatus;userdata=<text>;hostname=<text>;property=<property>;content=<status text>`

Mit diesem Kommando können Änderungen der Statusinformationen von **vimacc®** Hostrechnern abonniert werden.

Der Parameter `hostnames` in diesem Kommando spezifiziert, von welchen Hostrechnern die Statusinformationen angefordert werden.
Sollen mehrere Hostrechner abonniert werden, so sind die entsprechenden Rechnernamen mit einem Komma (',') voneinander zu trennen.
Der Parameter `hostnames` ist optional. Wird kein Rechnername übergeben, so werden die Statusinformationen von allen **vimacc®** Hostrechnern übermittelt.

Der Parameter `activate=1` aktiviert, `activate=0` deaktiviert die Anmeldung für Statusinformationen.

Jeder erneute Aufruf von `subscribehoststatus` überschreibt frühere Aufrufe dieses Kommandos.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Nach dem erfolgreichen Absetzen dieses Kommandos werden einmalig die Zustandsinformationen aller verfügbaren Hostrechner mit der Statusmeldung `resp=hoststatus` (siehe unten) an den Aufrufer gesendet.

Weitere Statusmeldungen `resp=hoststatus` werden dann nur noch an das MMS gesendet, wenn sich der Status eines der abonnierten **vimacc®** Hostrechner geändert hat.

Beispiel:

- MMS→vimacc:
  `cmd=subscribehoststatus;hostnames=host#1,host#2;userdata=789;activate=1`

  vimacc→MMS:
  `resp=subscribehoststatus;names=host#1,host#2;userdata=789;activate=1;answer=ok`

  vimacc→MMS (spontane Zustandsänderung):
  `resp=hoststatus;userdata=789;hostname=host#1;property=monitoring;content=memoryusage\=0.59\;partitionusage\=0.67`

3.7.3.28.1 Status-Informationen von vimacc Hostrechnern

Informationen über den Status von **vimacc®** Hostrechnern können in mehrere Bereiche unterteilt, die jeweils in dem Parameter `property=<property>` zurückgegeben werden.

Der zurückgegebene Parameter `content=<status text>` enthält dann die eigentliche Status-Information in Textform.

Mögliche Parameter sind:

- `monitoring`:
  Enthält die überwachten Geräteinformationen, wie z.B. Speicherauslastung ('`memoryusage=…`') und Auslastung der Speicherkapazität ('`partitionusage=…`') in Prozent.
- `availability`:
  Zeigt an, ob ein Gerät verfügbar ist oder nicht. Verfügbarkeit bedeutet in diesem Zusammenhang, ob das Gerät generell in der Systemdatenbank gefunden wurde und von der zugehörigen **vimacc®** Software-Komponente initialisiert und im **vimacc®** System registriert wurde.

3.7.4 VIMACC_CONTROL_DEVICES_ALARMS_ SCENARIOS

3.7.4.1 Allgemein

Dieses Protokoll erweitert das Protokoll VIMACC_CONTROL_BASIC um Kommandos zum Aufschalten von Szenarien, Übergeben von Alarmen an das **vimacc®** System und zum Annehmen und Abschließen von erzeugten Alarmen.

3.7.4.2 Aufschalten eines Szenarios

Kommando:

`cmd=showscenario;scenario=<name>;dest=<SenkenID>;createAlarm=<1|0>;contextid=<text>;userdata=<text>`

Antwort:

`resp=showscenario;scenario=<name>;dest=<SenkenID>;createAlarm=<1|0>;contextid=<text>;userdata=<text>;answer=ok|failed`

Mit diesem Kommando kann ein Szenario auf einer **vimacc®** Workstation-Instanz oder einer **vimacc®** Display-Instanz aufgeschaltet werden.

Szenarien werden dabei über ihren Namen referenziert. Die Szenarien müssen vorab im **vimacc®** System konfiguriert worden sein. Ist das Szenario zum Zeitpunkt der Aufschaltung nicht bekannt, so enthält die Antwort auf das Kommando den Text `answer=failed,unknown scenario.`

Der Parameter `createAlarm=<1|0>` bestimmt, ob bei der Aufschaltung des Szenarios ebenfalls ein Alarm von der entsprechenden **vimacc®** Workstation-Instanz ausgelöst werden soll. Die Auslösung eines Alarms zusammen mit einem Szenario bewirkt, dass alle in dem Szenario enthaltenen Videokameras in Alarmaufzeichnung gesetzt werden (vorausgesetzt, für die entsprechenden Kameras ist eine Aufzeichnungsverbindung konfiguriert). Üblicherweise bewirkt die Aktivierung einer Alarmaufzeichnung einer Kamera die Sicherung einer evtl. konfigurierten Voralarm-Aufzeichnung und die Erhöhung der Bildwiederholrate der Aufzeichnungsverbindung. Die Identifizierung des erzeugten Alarms erfolgt fortan über den im Parameter `contextid` übergebenen Wert. Der Aufrufer muss dabei die Eindeutigkeit der `contextid` sicherstellen.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Zum Beenden der durch das Szenario aufgeschalteten Verbindungen kann der Befehl `cmd=clear` (siehe oben) verwendet werden.

Hinweis:
Ein durch die Aufschaltung eines Szenarios aktivierter Alarm und die dadurch aktivierte Alarmaufzeichnung kann durch die Bestätigung des Alarms an einer der **vimacc®** Workstation-Instanzen beendet werden oder durch das entsprechende Kommando auf der Steuerschnittstelle (→ Befehl `finishAlarm`).

Beispiel:

- MMS→vimacc:
  `cmd=showscenario;scenario=Scenario#1;contextid=1234;dest=AP_1`

  vimacc→MMS: (Szenario bekannt)
  `resp=showscenario;scenario=Scenario#1;contextid=1234;dest=AP_1;answer=ok`

  vimacc→MMS: (Szenario unbekannt)
  `resp=showscenario;scenario=Scenario#1;contextid=1234;dest=AP_1;answer=unknown scenario`

3.7.4.3 Melden eines Alarmereignisses an das vimacc System

vimacc® ist in der Lage, auf spontane Ereignisse (sogenannte *Events*) zu reagieren und gegebenenfalls Aktionen automatisiert auszuführen, die einem Ereignis per Konfiguration zugeordnet worden sind.
Ein derartiges spontanes Ereignis kann zum Beispiel eine erkannte Bewegung in einem Kamerabild, ein geschlossener oder geöffneter Kontakt an einem I/O-Modul oder auch ein von einem MMS übergebener Alarm sein.

Eine **vimacc®** Workstation ist darüber hinaus in der Lage, die im **vimacc®** System signalisierten Ereignisse bzw. Alarme in einer sogenannten Alarmqueue darzustellen, um dem angemeldeten Benutzer einen Überblick über die signalisierten Ereignisse zu geben und eine Abarbeitung in einer bestimmten Reihenfolge zu ermöglichen.

Das Kommando `createalarmforalarmqueue` bietet die Möglichkeit, Alarme innerhalb des **vimacc®** Systems anzulegen, die in der Alarmqueue angezeigt werden sollen und die von einem Benutzer 'bearbeitet' werden sollen.
Weiterhin existieren Kommandos um Alarme zu beenden oder einen bestimmten Alarm aus der Alarmqueue an einer **vimacc®** Workstation anzunehmen und zu bestätigen (siehe Kapitel 3.7.4.4 und 3.7.4.5).

Kommando:
`cmd=createalarmforalarmqueue;contextid=<text>;timetolive=<wert>;scenario=<name>;alarmtype=<text>;alarmprio=<wert>;destinationids=<Komma-separierte ID Liste>;userdata=<text>`

Antwort:
`resp=createalarmforalarmqueue;contextid=<text>;scenario=<name;timetolive=<wert>;alarmtype=<text>;alarmprio=<wert>;destinationids=<ids>;userdata=<text>;answer=ok|failed`

Mit diesem Kommando kann ein Alarm innerhalb des **vimacc®** Systems angelegt werden, der automatisch in die Alarmqueue des Systems eingefügt wird.

Der Parameter `contextid` dient fortan zur Identifizierung und Referenzierung des erzeugten Alarms. Der Aufrufer muss dabei die Eindeutigkeit der `contextid` sicherstellen.
Ist bereits ein Alarm mit dieser `contextid` innerhalb des **vimacc®** Systems aktiv, so wird die Antwort `answer=failed,duplicate contextid` zurückgegeben.

Das Annehmen oder Beenden eines Alarms ist später nur mit Hilfe der hier übergebenen `contextid` möglich (→ siehe Befehle `acceptalarm` und `finishalarm`).

Ein auf diese Weise erzeugter Alarm wird automatisch in eine Alarm-Warteschlange, die sogenannte Alarm-Queue, eingetragen, und es wird erwartet, dass der Alarm von einem Benutzer (oder vom MMS) angenommen und bestätigt wird.

Der Parameter `timetolive` legt die Lebensdauer des Alarms in Sekunden fest. Der Wert `0` bedeutet `unendlich`, so dass der Alarm solange bestehen bleibt, bis er angenommen und bestätigt worden ist, d.h. er wird nicht automatisch vom **vimacc®** System beendet.
Ein Wert `x mit x>0` bedeutet, dass der Alarm nach x Sekunden automatisch vom **vimacc®** System beendet werden soll.
Es ist zu beachten, dass wenn bei dem Parameter `timetolive` ein Wert größer 0 übergeben wird, der Alarm nach Ablauf der entsprechenden Zeit automatisch vom **vimacc®** System beendet wird, unabhängig davon, ob ein Benutzer den Alarm angenommen und bestätigt hat.

Der Parameter `scenario` bestimmt das Szenario, das aufgeschaltet werden soll, wenn der Alarm an einem **vimacc®** Arbeitsplatz angenommen worden ist. Die Alarmannahme kann dabei entweder vom Benutzer selbst, oder gesteuert durch das MMS erfolgen (→ siehe Befehl `acceptalarm`).
Szenarien werden dabei über ihren Namen referenziert. Die Szenarien müssen vorab im **vimacc®** System konfiguriert worden sein.

Der Parameter `alarmtype` ist optional und erlaubt die Festlegung einer Typbezeichnung für den erzeugten Alarm. Dieser Typ kann zur besseren Unterscheidung von verschiedenen Alarmen verwendet werden.

Der Parameter `alarmprio` ist optional und erlaubt die Festlegung einer Alarmpriorität. Innerhalb eines **vimacc®** Systems ist die höchste Alarmpriorität durch den Wert `0` definiert. Wird dieser Parameter nicht angegeben, so wird ein Alarm mit der Priorität `1`erzeugt.

Der Parameter `destinationids` ist optional und erlaubt die Festlegung, an welchen **vimacc®** Arbeitsplätzen der erzeugte Alarm in der Alarmqueue angezeigt werden soll. Hier wird eine Komma-separierte Liste von **vimacc®** IDs der entsprechenden **vimacc®**

Arbeitsplatz-Instanzen erwartet. Wird dieser Parameter nicht angegeben, so wird der Alarm an allen **vimacc®** Arbeitsplätzen angezeigt.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:

- MMS→vimacc:
  `cmd=createalarmforalarmqueue;contextid=1234;timetolive=0;scenario=scenario\ #1`

  vimacc→MMS: (contextid noch nicht vergeben)
  `resp=createalarmforalarmqueue;contextid=1234;timetolive=0;scenario=scenario\ #1;answer=ok`

  vimacc→MMS: (contextid bereits vergeben)
  `resp=createalarmforalarmqueue;contextid=1234;timetolive=0;scenario=scenario\ #1;answer=failed,duplicate contextid`

3.7.4.4 Annehmen eines Alarms für eine Workstation-Instanz

Kommando:
`cmd=acceptalarm;contextid=<text>;dest=<SenkenID>;userdata=<text>`

Antwort:
`resp=acceptalarm;contextid=<text>;dest=<SenkenID>;userdata=<text>;answer=ok|failed`

Mit diesem Kommando kann ein anstehender Alarm innerhalb des **vimacc®** Systems einer **vimacc** Workstation-Instanz direkt zur Bearbeitung zugewiesen werden, d.h. der entsprechende Alarm wird stellvertretend für einen Benutzer von der steuernden Instanz angenommen.

Die Annahme eines Alarms bewirkt, dass dieser aus der Alarm-Warteschlange des Systems ausgetragen wird und ein eventuell vorab übergebenes Szenario (→ siehe Befehl `createalarm`) an der entsprechenden **vimacc®** Workstation-Instanz aufgeschaltet wird.

Der Parameter `contextid` dient zur Identifizierung des Alarms. Ist kein Alarm mit dieser `contextid` innerhalb des **vimacc®** Systems bekannt, so kann dieser Befehl nicht ausgeführt werden und es wird die Antwort `answer=failed,unknown contextid` zurückgegeben.

Der Parameter `dest` definiert die **vimacc®** Workstation-Instanz, an der der Alarm angenommen werden soll. Ein vorab übergebenes Szenario (→ siehe Befehl `createalarm`) wird genau an dieser **vimacc®** Workstation-Instanz aufgeschaltet. Ist

die referenzierte Workstation nicht verfügbar, so kann dieser Befehl nicht ausgeführt werden und es wird die Antwort `answer=failed,device not available` zurückgegeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet. Zusätzlich wird dieser Wert in der **vimacc®** Ereignisdatenbank dokumentiert und kann z.B. im Zuge einer Nachbearbeitung als Suchkriterium verwendet werden.

Beispiel:

- MMS→vimacc:
  `cmd=acceptalarm;contextid=1234;dest=AP_1;userdata=data`

  vimacc→MMS: (contextid bekannt)
  `resp=acceptalarm;contextid=1234;dest=AP_1;userdata=data;answer=ok`

  vimacc→MMS: (contextid nicht bekannt)
  `resp=acceptalarm;contextid=1234;dest=AP_1;userdata=data;answer=failed,unknown contextid`
  vimacc→MMS: (Workstation-Instanz nicht verfügbar)
  `resp=acceptalarm;contextid=1234;dest=AP_1;userdata=data;answer=failed,device not available`

3.7.4.5 Beenden eines Alarms

Kommando:
`cmd=finishalarm;contextid=<text>;userdata=<text>;tags=<text>`

Antwort:
`resp=finishalarm;contextid=<text>;userdata=<text>;tags=<text>;answer=ok|failed`

Mit diesem Kommando kann ein Alarm innerhalb des **vimacc®** Systems beendet werden.

Die Beendigung eines Alarms bewirkt, dass entsprechend konfigurierte Aktionen automatisch durchgeführt werden und der Alarm abgeschlossen wird, so dass keinerlei Aktionen bzgl. dieses Alarms mehr durchgeführt werden können.

Falls der Alarm sich noch in der Alarm-Warteschlange des Systems enthalten ist, so wird er dort automatisch ausgetragen.

Der Parameter `contextid` dient zur Identifizierung des Alarms. Ist kein Alarm mit dieser `contextid` innerhalb des **vimacc®** Systems bekannt, so kann dieser Befehl nicht ausgeführt werden und es wird die Antwort `answer=failed,unknown contextid` zurückgegeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet. Zusätzlich wird dieser Wert in der **vimacc®** Ereignisdatenbank dokumentiert und kann z.B. im Zuge einer Nachbearbeitung als Suchkriterium verwendet werden.

Der Parameter `tags` wird nicht ausgewertet; allerdings wird er in der **vimacc®** Ereignisdatenbank dokumentiert und kann z.B. im Zuge einer Nachbearbeitung als Suchkriterium verwendet werden.

Beispiel:

- MMS→vimacc:
  ```
  cmd=finishalarm;contextid=1234;userdata=data;tags=EMA
  ```

  vimacc→MMS: (contextid bekannt)
  ```
  resp=finishalarm;contextid=1234;userdata=data;tags=EMA;answer=ok
  ```

  vimacc→MMS: (contextid nicht bekannt)
  ```
  resp=finishalarm;contextid=1234;userdata=data;tags=EMA;answer=unknown contextid
  ```

3.7.4.6 Auslösen des Alarmzustandes eines vimacc Devices

Kommando:

```
cmd=triggerdevicealarm;source=<sourceID>;alarmid=<text>;alarmtime=<utc timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz,
Default: current daytime>;userdata=<text>;contextid=<text>
```

Antwort:

```
resp=triggerdevicealarm;source=<sourceID>;alarmid=<text>;alarmtime=<utc timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz,
Default: current daytime>;userdata=<text>;contextid=<text>;
answer=ok|failed
```

Mit diesem Kommando kann ein bestimmtes **vimacc®** Device in den Zustand 'Alarm' versetzt werden.

Der Zustand 'Alarm' kann je nach Device unterschiedliche Reaktionen auslösen.
In **vimacc®** können z.B. für Videokameras verschiedene Aufzeichnungsverbindungen konfiguriert werden, von denen üblicherweise eine Verbindung für eine 'normale' Archivierung und eine weitere für eine Alarmaufzeichnung verwendet wird. Eine Alarmaufzeichnung kann dann wiederum so konfiguriert werden, dass bei Auslösung eines Alarms die Sicherung einer evtl. konfigurierten Voralarm-Aufzeichnung und eine Erhöhung der Bildwiederholrate für die Dauer des Alarms erfolgen.

Die Auslösung eines Alarms bei einem bestimmten **vimacc®** Device kann also durch dieses Kommando aktiviert werden.

Der Zustand 'Alarm' bei einem **vimacc®** Device bleibt solange bestehen, bis die Alarmierung beispielsweise durch das Kommando `cleardevicealarm` wieder zurückgesetzt wird (siehe Kapitel 3.7.4.7).

Der Parameter `source` gibt die ID des Devices an, bei dem der Zustand 'Alarm' ausgelöst werden soll.

Die Parameter `alarmid` dient zur Identifizierung des Alarms innerhalb des Devices. Es können mehrere Alarme für ein Device ausgelöst werden, wobei jeder Alarm muss allerdings über eine andere `alarmid` verfügen muss.

Jeder ausgelöste Alarm muss wieder durch das Kommando `cleardevicealarm` mit der entsprechenden `alarmid` zurückgesetzt werden. Der Zustand 'Alarm' wieder von dem Device erst dann wieder zurückgenommen, wenn alle Alarme wieder zurückgesetzt worden sind (siehe Kapitel 3.7.4.7).

Die Parameter `alarmtime` dient zur Kennzeichnung des Alarmzeitpunktes. Dieser Parameter ist optional. Wird hier ein Zeitstempel übergeben, so wird dieser als UTC (Coordinated Universal Time) Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` erwartet.
Wird kein Zeitstempel übergeben, so wird das aktuelle Datum und die aktuelle Uhrzeit als Alarmzeitpunkt festgelegt.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der vimacc Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Hinweis:
Die Aktivierung eines Alarms wird von **vimacc®** automatisch bei den Devices durchgeführt, wenn ein Kommando zum Erzeugen (siehe Kapitel 3.7.4.2 und 3.7.4.3) eines Alarms übergeben wurden.

Hinweis:
Es ist wichtig zu beachten, dass dieses Kommando ausschließlich den Alarmierungszustand eines einzelnen **vimacc®** Devices auslösen kann. Die Reaktion auf den Zustand 'Alarm' erfolgt ausschließlich innerhalb eines Devices auf die oben beschriebene Weise.
Durch dieses Kommando kann kein Alarm innerhalb des **vimacc®** Systems ausgelöst werden. Hierzu müssen die Kommandos zum Erzeugen eines Alarmes (siehe Kapitel 3.7.4.2 und 3.7.4.3) verwendet werden.

Beispiel:
MMS→vimacc:
```
cmd=triggerdevicealarm;source=cam_demostreams_maneki-
neko_0001;
alarmid=testalarm;alarmtime=20220320T182037.125Z;userdata=test
;contextid=1234
```

vimacc→MMS:
```
resp=triggerdevicealarm;source=cam_demostreams_maneki-
neko_0001;alarmid=testalarm;
alarmtime=20220320T182037.125Z;userdata=test;contextid=1234
;answer=ok
```

3.7.4.7 Zurücksetzen des Alarmzustandes eines vimacc Devices

Kommando:
```
cmd=cleardevicealarm;source=<sourceID>;alarmid=<text>;userdata
=<text>;contextid=<text>
```

Antwort:
```
resp=cleardevicealarm;source=<sourceID>;alarmid=<text>;userdat
a=<text>;contextid=<text>;answer=ok|failed
```

Mit diesem Kommando kann bei einem bestimmten **vimacc®** Device ein vorab ausgelöster Alarm wieder zurückgesetzt werden.
Der Gesamtzustand 'Alarm' wird allerdings erst dann zurückgenommen, wenn alle innerhalb des Devices ausgelösten Alarme wieder zurückgesetzt worden sind.

Das Zurücksetzen des Zustandes 'Alarm' kann je nach Device unterschiedliche Reaktionen auslösen.

Bei den in Kapitel 3.7.4.6 beschriebenen Videokameras führt das Zurücksetzen des Zustandes 'Alarm' zum Beenden der gestarteten Alarmaufzeichnung.

Der Parameter `source` gibt die ID des Devices an, bei dem der Zustand 'Alarm' zurückgesetzt werden soll.

Die Parameter `alarmid` dient zur Identifizierung des Alarms innerhalb des Devices.

Die Parameter `alarmtime` dient zur Kennzeichnung des Alarmzeitpunktes. Dieser Parameter ist optional. Wird hier ein Zeitstempel übergeben, so wird dieser als UTC (Coordinated Universal Time) Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` erwartet.
Wird kein Zeitstempel übergeben, so wird das aktuelle Datum und die aktuelle Uhrzeit als Alarmzeitpunkt festgelegt.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird

nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc®** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Hinweis:
Das Zurücksetzen eines Alarmes wird von **vimacc®** automatisch bei den Devices aktiviert, wenn ein Kommando zum Beenden eines Alarms (siehe Kapitel 3.7.4.5) übergeben wurden.

Hinweis:
Es ist wichtig zu beachten, dass dieses Kommando ausschließlich den Alarmierungszustand eines einzelnen **vimacc®** Devices zurücksetzt. Durch dieses Kommando kann kein Alarm innerhalb des **vimacc®** Systems beendet werden. Hierzu muss das Kommando zum Beenden eines Alarms (siehe Kapitel 3.7.4.5) verwendet werden.

Beispiel:
MMS→vimacc:
```
cmd=cleardevicealarm;source=cam_demostreams_maneki-neko_0001;
alarmid=testalarm;userdata=test;contextid=1234
```

vimacc→MMS:
```
resp=cleardevicealarm;source=cam_demostreams_maneki-neko_0001;
alarmid=testalarm;userdata=test;contextid=1234;answer=ok
```

3.7.4.8 Löschschutz für Zeitbereiches eines Wiedergabe-Streams

Kommando:
```
cmd=addstreamprotection;playbackid=<playbackid;begintime=<utc
timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz>;endtime=<utc
timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz>;
userdata=<text>;contextid=<text>
```

Antwort:
```
resp=addstreamprotection;playbackid=<playbackid>;begintime=<ti
mestamp>;endtime=<timestamp>;userdata=<text>;userdata=<text>;c
ontextid=<text>;answer=ok|failed
```

Mit diesem Kommando kann ein bestimmter Zeitbereich innerhalb eines Wiedergabe-Streams vor dem Überschreiben geschützt werden (Löschschutz). Dies ist dann sinnvoll, wenn der zugehörige Stream normalerweise in einem Ring aufgezeichnet wird, bestimmte Bereiche aber bei dem nächsten Umlauf nicht gelöscht werden soll.

Sollte sich der aufgezeichnete Inhalt auf mehreren Aufzeichnungsservern befinden, so werden die Zeitbereiche auf allen Servern geschützt.

Der Parameter `playbackid` gibt die ID des Wiedergabe-Streams an, bei dem Bereiche geschützt werden sollen.

Die Parameter `begintime` und `endtime` definieren den zu schützenden Zeitbereich. Die übergebenen Zeitstempel werden als UTC (Coordinated Universal Time) Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` erwartet.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc®** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Hinweis:
Sind einem Stream weitere Kanäle zugeordnet (siehe `associations` in Kapitel 3.7.3.15), so werden die angegebenen Zeitbereiche in diesen Kanälen nicht automatisch geschützt. Hierzu müssen die entsprechenden Wiedergabe-Streams durch erneutes Aufrufen des Kommandos `addstreamprotection` explizit geschützt werden.

Beispiel:
MMS→vimacc:

```
cmd=addstreamprotection;playbackid=cam0231_archiv;contextid=co
ntext#1;begintime=2014-03-31T18:00:37.890;endtime=2014-03-
31T18:10:37.890;userdata=test;contextid=<text>
```

vimacc→MMS:

```
resp=addstreamprotection;playbackid=cam0231_archiv;contextid=c
ontext#1;begintime=2014-03-31T18:00:37.890;endtime=2014-03-
31T18:10:37.890;userdata=test;answer=ok
```

3.7.4.9 Entfernen des Löschschutzes für einen Zeitbereich eines Wiedergabe-Streams

Kommando:

```
cmd=removestreamprotection;playbackid=<playbackid>;begintime=<
utc timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz>;
endtime=<utc timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz>;
userdata=<text>;contextid=<text>
```

Antwort:

```
resp=removestreamprotection;playbackid=<playbackid>;begintime=
<timestamp>;endtime=<timestamp>;userdata=<text>;userdata=<text
>;contextid=<text>;answer=ok|failed
```

Mit diesem Kommando kann ein Zeitbereich eines Wiedergabe-Streams, der vorab durch das Kommando `addstreamprotection` (siehe Kapitel 3.7.4.8) vor dem Überschreiben geschützt wurde, wieder freigegeben werden, so dass er beim nächsten Umlauf des Ringspeichers wieder gelöscht werden kann.

Sollte sich der aufgezeichnete Inhalt auf mehreren Aufzeichnungsservern befinden, so wird der Schreibschutz auf allen Servern aufgehoben.

Der Parameter `playbackid` gibt die ID des Wiedergabe-Streams an, bei dem ein geschützter Bereich wieder freigegeben werden soll.

Die Parameter `begintime` und `endtime` definieren den geschützten Zeitbereich. Die übergebenen Zeitstempel werden als UTC (Coordinated Universal Time) Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` erwartet.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc®** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:
MMS→vimacc:

```
cmd=removestreamprotection;playbackid=cam0231_archiv;contextid
=context#1;begintime=2014-03-31T18:00:37.890;endtime=2014-03-
31T18:10:37.890;userdata=test
```

vimacc→MMS:

```
resp=removestreamprotection;playbackid=cam0231_archiv;contexti
d=context#1;begintime=2014-03-31T18:00:37.890;endtime=2014-03-
31T18:10:37.890;userdata=test;answer=ok
```

3.7.4.10 Abfragen der geschützten Bereiche eines Wiedergabe-Streams

Kommando:

```
cmd=getstreamprotectionlist;playbackid=<playbackid>;userdata=<
text>
```

Antwort:

```
resp=getstreamprotectionlist;playbackid=<playbackid>;userdata=
<text>;answer=ok|failed;parameterlist{\r\n
begintime=<utc timestamp iso 8601: yyyy-MM-
dd'T'hh:mm:ss.zzz>;endtime=<utc timestamp iso 8601: yyyy-MM-
dd'T'hh:mm:ss.zzz>\r\n
begintime=<utc timestamp iso 8601: yyyy-MM-
dd'T'hh:mm:ss.zzz>;endtime=<utc timestamp iso 8601: yyyy-MM-
dd'T'hh:mm:ss.zzz>\r\n …
\r\n}
```

Mit diesem Kommando können die geschützten Bereiche eines Wiedergabe-Streams abgefragt werden. Geschützte Bereiche können durch das Kommando `addstreamprotection` festgelegt werden (siehe Kapitel 3.7.4.8).

Sollte sich der aufgezeichnete Inhalt auf mehreren Aufzeichnungsservern befinden, so werden die geschützten Zeitbereiche auf allen Servern ermittelt und alle gefundenen Zeitbereiche zurückgegeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Der Parameter `playbackid` gibt die ID des Wiedergabe-Streams an, dessen Grenzen ermittelt werden sollen.

Die ermittelten Grenzen werden als UTC (Coordinated Universal Time) Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` zurückgeliefert.

Für jeden geschützten Zeitabschnitt wird innerhalb des Antworttextes eine Zeile in der Form `begintime=<timestamp>\;endtime=<timestamp>\r\n` zurückgegeben.

Beispiel:
MMS→vimacc:

```
cmd=getstreamprotectionlist;playbackid=cam0231_archiv;userdata
=test
```

vimacc→MMS:

```
resp=getstreamprotectionlist;playbackid=cam0231_archiv;userdat
a=test;answer=ok;parameterlist
{\r\nbegintime=2014-04-01T11:42:19.246\;endtime=2014-04-
01T11:49:38.727\r\n}
```

3.7.4.11 Löschen eines Zeitbereiches aus einem Wiedergabe-Stream

Kommando:

```
cmd=removetimespanfromstream;playbackid=<playbackid>;begintime
=<utc timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz>;
endtime=<utc timestamp iso 8601: yyyy-MM-dd'T'hh:mm:ss.zzz>;
userdata=<text>;contextid=<text>
```

Antwort:

```
resp=removetimespanfromstream;playbackid=<playbackid>;begintim
e=<timestamp>;endtime=<timestamp>;userdata=<text>;userdata=<te
xt>;contextid=<text>;answer=ok|failed
```

Mit diesem Kommando kann ein bestimmter Zeitbereich aus einem aufgezeichneten Wiedergabe-Stream gelöscht werden.

Sollte sich der aufgezeichnete Inhalt auf mehreren Aufzeichnungsservern befinden, so wird der angegebene Zeitbereich auf allen Servern gelöscht.

Der Parameter `playbackid` gibt die ID des Wiedergabe-Streams an, aus dem der angegebene Zeitbereich gelöscht werden soll.

Die Parameter `begintime` und `endtime` definieren den zu löschenden Zeitbereich. Die übergebenen Zeitstempel werden als UTC (Coordinated Universal Time) Zeitstempel in der Form `yyyy-MM-dd'T'hh:mm:ss.zzz` erwartet.

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:
MMS→vimacc:

```
cmd=removetimespanfromstream;playbackid=cam0231_archiv;context
id=context#1;begintime=2014-03-31T18:00:37.890;endtime=2014-
03-31T18:10:37.890;userdata=test
```

vimacc→MMS:

```
resp=removetimespanfromstream;playbackid=cam0231_archiv;contex
tid=context#1;begintime=2014-03-31T18:00:37.890;endtime=2014-
03-31T18:10:37.890;userdata=test;answer=ok
```

3.7.4.12 Setzen einer Textmarke für einen Live-Stream

Kommando:
`cmd=setbookmarkforstream;deviceid=<deviceid>;text=<text>;userdata=<text>;contextid=<text>`

Antwort:
`resp=setbookmarkforstream;deviceid=<deviceid>;text=<text>;userdata=<text>;contextid=<text>;answer=ok|failed`

Mit diesem Kommando kann eine Textmarke (Lesezeichen) zu einem bestimmten Stream gespeichert werden, so dass bei einer späteren Auswertung die entsprechende Stelle schneller wiedergefunden werden kann.
Der übergeben Text wird dabei mit dem zugehörigen Zeitpunkt an die **vimacc®** Dokumentationsebene übergeben.

Der Parameter `deviceid` gibt die ID des Streams an, dem zu übergebene `text` zugewiesen werden soll.

Der im Parameter `text` übergebene Eintrag sollte in Hochkommata (`"<text>"`) eingeschlossen werden, damit er auch Leerzeichen und andere Sonderzeichen enthalten kann. (Semikolon, Doppelpunkt und Hochkommata müssen escaped werden).

Der Parameter `contextid` dient zur Identifizierung und Referenzierung des Kommandos zu vorherigen Kommandos oder Ereignissen. Der Parameter selbst wird nicht ausgewertet, sondern als Zuordnungsmerkmal bei der Protokollierung des Kommandos in der **vimacc®** Dokumentationsebene übergeben.

Der Parameter `userdata` wird nicht ausgewertet, sondern in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:
MMS→vimacc:
```
cmd=setbookmarkforstream;deviceid=cam0231;contextid=context#1;
text="Dies ist der Text eines Lesezeichens";userdata=test
```

vimacc→MMS:
```
resp=setbookmarkforstream;deviceid=cam0231;contextid=context#1
;text="Dies ist der Text eines
Lesezeichens";userdata=test;answer=ok
```

3.7.5 VIMACC_CONTROL_ALL

3.7.5.1 Allgemein

Dieses Protokoll erweitert das Protokoll VIMACC_CONTROL_DEVICES_ALARMS_SCENARIOS um Kommandos zum Schreiben von beliebigen "Datenpunkten" des **vimacc** Systems.

3.7.5.2 Schreiben eines beliebigen Datenpunktes

Kommando:
`cmd=writedp;contextid=<text>;datapointname=<text>;datapointvalue=<text>`

Antwort:
`resp=writedp;contextid=<text>;datapointname=<text>;datapointvalue=<text>;answer=ok|failed`

Mit diesem Kommando kann ein beliebiger Datenpunkt des **vimacc** Systems geschrieben werden.

Der Parameter `datapointname` definiert den zu schreibenden Datenpunkt in der **vimacc** Config, beginnend vom Wurzelknoten. Daher muss der komplette Pfad des Datenpunktes angegeben werden.

Der Parameter `datapointvalue` legt den Inhalt fest, der in den übergebenen Datenpunkt geschrieben werden soll.

Der Parameter `contextid` wird nicht ausgewertet, sondern dient der Protokollierung und wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Hinweis:
Als Inhalt für den Parameter `datapointvalue` müssen hier ebenfalls Schlüssel-Werte-Paare übergeben werden, wie z.B. das Kommando `cmd=show;source=cam0231;videodlg=1.`
Für die korrekte Protokollauswertung muss der zu übergebene Text des Parameters `datapointvalue` dreimal mit dem Sonderzeichen '\' 'escaped' werden.
Der Grund dafür ist, dass ansonsten keine Sonderzeichen wie etwa ('), (\), (\r), (\n) und (\t) übergeben und enthaltene Schlüssel-Werte-Paare nicht korrekt ermittelt werden könnten.
Da die Schlüssel-Werte-Paare dieses Parameters genau wie das gesamte Kommando mit einem Gleichheitszeichen (=) aufgebaut sind und die einzelnen Schlüssel-Werte-Paare auch mit einem Semikolon (;) voneinander getrennt sind, könnte ohne die Escape-Zeichen diese Struktur nicht korrekt aus dem Kommando ermittelt werden können.

Daher muss der übergebenen Text in diesem Parameter auf die folgende Weise 'escaped' werden:

1. Da die Kommandos mit der Schlüssel-Werte-Paar `cmd=<Kommando>` identifiziert werden, muss zunächst das Zeichen '=' escaped werden.
2. Da die Schlüssel-Werte-Paar mit einem ';' voneinander getrennt werden, muss in dem resultierenden Text das Zeichen ';' escaped werden.
3. Da der Protokoll-Parser ebenfalls den empfangen Text einmal 'de-escaped', müssen in dem resultierenden Text erneut alle Sonderzeichen escaped werden.

Soll beispielsweise in dem Parameter `datapointvalue` das Kommando `cmd=show;source=cam0231;videodlg=1` übergeben werden, so ergeben diese drei Arbeitsschritte den folgenden zu übergebenen Text:

1. Zeichen '=' escapen:
   ➔ `cmd\=show;source\=cam0231;videodlg\=1`
2. In dem resultierenden Text das Zeichen ';' escapen:
   ➔ `cmd\\=show\;source\\=cam0231\;videodlg\\=1`
3. In dem resultierenden Text erneut alle Sonderzeichen escapen:
   ➔ `cmd\\\\=show\\;source\\\\=cam0231\\;videodlg\\\\=1`

Beispiel:

- MMS→vimacc:
  ```
  cmd=writedp;contextid=1234;datapointname=ActiveDeviceList
  /4000/command/request;datapointvalue=cmd\\\\=show\\;sourc
  e\\\\=10000\\;videodlg\\\\=VD1
  ```

  vimacc→MMS:
  ```
  resp=writedp;contextid=1234;datapointname=ActiveDeviceLis
  t/4000/command/request;datapointvalue=cmd\\\\=show\\;sour
  ce\\\\=10000\\;videodlg\\\\=VD1;answer=ok
  ```

3.7.5.3 Schreiben des Kommando-Datenpunktes eines vimacc Devices

Kommando:

```
cmd=writecommanddp;contextid=<text>;deviceid=<deviceid>;datapo
intvalue=<text>
```

Antwort:

```
resp=writecommanddp;contextid=<text>;deviceid=<deviceid>;datap
ointvalue=<text>;answer=ok|failed
```

Mit diesem Kommando kann der Kommando-Datenpunkt (genauer der `command/request` Datenpunkt) eines **vimacc®** Devices geschrieben werden.

Der Parameter `deviceid` definiert das entsprechende **vimacc®** Device. Es wird überprüft, ob das Device innerhalb des **vimacc®** Systems bereits aktiviert worden ist. Ist dies nicht der Fall, so wird `answer=failed,device not available` zurückgegeben.

Der Parameter `datapointvalue` legt den Inhalt fest, der in den Kommando-Datenpunkt geschrieben werden soll. Auch hier muss der übergebenen Text 'escaped' werden (→Befehl `writedp`).

Der Parameter `contextid` wird nicht ausgewertet, sondern dient der Protokollierung und wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Beispiel:

- MMS→vimacc:
  ```
  cmd=writecommanddp;contextid=1234;deviceid=4000;datapoint
  value=cmd\\\\=show\\;source\\\\=10000\\;videodlg\\\\=VD1
  ```

  vimacc→MMS:
  ```
  resp=writecommanddp;contextid=1234;deviceid=4000;datapoin
  tvalue=cmd\\\\=show\\;source\\\\=10000\\;videodlg\\\\=VD1
  ;answer=ok
  ```

3.7.5.4 Lesen eines beliebigen Datenpunktes

Kommando:

```
cmd=readdp;userdata=<text>;datapointname=<text>
```

Antwort:

```
resp=readdp;userdata=<text>;datapointname=<text>;answer=ok,<da
tapoint value>|failed
```

Mit diesem Kommando kann ein beliebiger Datenpunkt des **vimacc®** Systems gelesen werden.

Der Parameter `datapointname` definiert den zu lesenden Datenpunkt in der **vimacc®** Config, beginnend vom Wurzelknoten. Daher muss der komplette Pfad des Datenpunktes angegeben werden.

Der Parameter `userdata` wird nicht ausgewertet, sondern wird in den Antworten zur besseren Unterscheidung eintreffender Nachrichten wieder zurückgesendet.

Der Wert des gelesenen Datenpunktes wird in der Antwort zurückgegeben.

Hinweis:

Damit der zurückgegebene Inhalt des gelesenen Datenpunktes aus dem Antworttext extrahiert werden kann, wird etwaigen Sonderzeichen wie etwa ("), (\), (\r), (\n) und (\t) und den Zeichen (=), (,) und (;) das Escape-Zeichen (\) vorangestellt.

Beispiel:

- MMS→vimacc:
  ```
  cmd=readdp;userdata=1234;datapointname=ActiveDeviceList/401/command/response
  ```

  vimacc→MMS:
  ```
  resp=readdp;userdata=1234;datapointname=ActiveDeviceList/401/command/response;answer=ok,cmd\=acceptalarm\;contextid\=11\;error\=rejected
  ```

3.7.6 VIMACC_CONTROL_FALLBACK

3.7.6.1 Allgemein

Dieses Protokoll bietet nur einen sehr eingeschränkten Satz Kommandos. Es dient dazu, einige Informationen über den Zustand eines **vimacc®** Systems zu ermitteln, auch wenn dieses nicht mehr ordnungsgemäß betrieben werden kann.

Dieser Fall tritt beispielsweise ein, wenn die zu Grunde liegende Lizenz noch nicht aktiviert oder abgelaufen ist. Demo-Setups laufen üblicherweise bis zu 12 Wochen. Nach Ablauf dieser Zeit werden die **vimacc®** Komponenten deaktiviert.
Damit aber ein übergeordnetes Management System zumindest über die wesentlichen Fehlerzustände informiert werden kann, wird automatisch auf dieses Protokoll zurückgeschaltet.

3.7.6.2 Abfragen der verfügbaren Befehle

Kommando:
`cmd=help;userdata=<text>`

Antwort:
`resp=help;userdata=<text>;answer=ok,parameterlist{\r\nbefehl#1 \r\n...befehl#n\r\n}`

Details siehe Kapitel 3.7.3.2.

3.7.6.3 Überwachung der Steuerverbindung

Kommando:
`cmd=keepalive;userdata=<text>`

Antwort:
`resp=keepalive;userdata=<text>;answer=ok`

Details siehe Kapitel 3.7.3.3.

3.7.6.4 Anfordern von Status-Informationen des vimacc Systems

Kommando:
`cmd=subscribesystemstatus;userdata=<text>;activate=<0|1>`

Antwort:
`resp=subscribesystemstatus;userdata=<text>;activate=<0|1>;answer=ok|failed`

Statusmeldung:
`resp=systemstatus;userdata=<text>;property=<property>;content=<status text>`

Details siehe Kapitel 3.7.3.26.

4 vimacc Live

Der Zugriff auf Live-Streams des Systems kann über eine Implementierung des **vimacc®** Video-Widgets oder über den **vimacc®** RTSP-Server erfolgen.
Siehe auch *Kapitel 0*

Abfragen der konfigurierten Kameras

4.1 Video-Widget

Das Video Widget ist eine C++/QT-Basisimplementierung zur Anzeige von Video-Streams eines **vimacc®** Systems. Video-Streams werden per TCP/IP direkt vom jeweiligen **vimacc®** Interface-Prozess abgerufen und angezeigt. Die entsprechenden Codecs sind bereits enthalten. Die implementierende Stelle muss bei MPEGLA die entsprechenden Patenportfolio-Nutzungsrechte anmelden und ggf. abrechnen.

Eine Implementierungsvariante des **vimacc®** Video-Widgets ist das Video-EWO aus SIMATIC WinCC OA Video.
Ein autonomes Video-Widget unterliegt bei der Anzeige von Streams nicht dem **vimacc®** Rechtemanagement, da es keine Verbindung zur zentralen **vimacc®** Konfiguration hat und damit keine Zuordnung von benutzerrechten vorgenommen werden kann.
Die abrufende Stelle hat über entsprechende Mechanismen für einen Zugriffsschutz zu sorgen.

4.2 RTSP-Server

Der **vimacc®** RTSP-Server ist Bestandteil der Basis-Setups aller **vimacc®** Editionen. Er muss über die Lizenzfiles entsprechend aktiviert und mittel **vimacc®** AdministrationCenter konfiguriert werden.
Siehe **vimacc®** Administratorhandbuch

Der Zugriff erfolgt in üblicher Zugriffsnotation über diese URL-Struktur:
`rtsp://<Rechnername>:<Port>/<StreamID>/live`

`<Rechnername>`	Name oder IP-Adresse des Servers/PCs
`<Port>`	Portnummer (default: 5544)
`<StreamID>`	**vimacc®** Stream-ID

Siehe auch *Kapitel 0*

Abfragen der konfigurierten Kameras

Bsp.: rtsp://VideoServer1:5544/cam_axis2025_0001/**live**

Über den RTSP-Server können nur Streams im H.264- oder MPEG4-Format abgerufen werden. Für den Zugriff auf Streams mit anderen Codecs steht der **vimacc®** http-Server zur Verfügung. Dieser stellt alle Stream-Typen in einem einheitlichen, transcodierten MJPEG-Stream bereit.

5 vimacc Playback

Der Zugriff auf Live-Streams des Systems kann über eine Implementierung des **vimacc®** Video-Widgets oder über den **vimacc®** RTSP-Server erfolgen.
Siehe auch *Kapitel 3.7.3.15 Abfragen der verfügbaren Wiedergabe-Streams*

5.1 Video-Widget

Das Video Widget ist eine C++/QT-Basisimplementierung zur Anzeige von Video-Streams eines **vimacc®** Systems. Video-Streams werden per TCP/IP direkt vom jeweiligen **vimacc®** Server-Prozess abgerufen und angezeigt. Die entsprechenden Codecs sind bereits enthalten. Die implementierende Stelle muss bei MPEGLA die entsprechenden Patenportfolio-Nutzungsrechte anmelden und ggf. abrechnen.

Eine Implementierungsvariante des **vimacc®** Video-Widgets ist das Video-EWO aus SIMATIC WinCC OA Video.
Ein autonomes Video-Widget unterliegt bei der Anzeige von Streams nicht dem **vimacc®** Rechtemanagement, da es keine Verbindung zur zentralen **vimacc®** Konfiguration hat.
Die abrufende Stelle hat über entsprechende Mechanismen für einen Zugriffsschutz zu sorgen.

5.2 RTSP-Server

Der **vimacc®** RTSP-Server ist Bestandteil der Basis-Setups aller **vimacc®** Editionen. Er muss über die Lizenzfiles entsprechend aktiviert und mittel **vimacc®** AdministrationCenter konfiguriert werden.
Siehe **vimacc®** Administratorhandbuch

Der Zugriff erfolgt in üblicher Zugriffsnotation über diese URL-Struktur:
`rtsp://<Rechnername>:<Port>/<StreamID>/`**`playback`**

`<Rechnername>`	Name oder IP-Adresse des Servers/PCs
`<Port>`	Portnummer (default: 5544)
`<StreamID>`	**vimacc®** Stream-ID

Siehe auch *Kapitel 3.7.3.15 Abfragen der verfügbaren Wiedergabe-Streams*

Bsp.: rtsp://VideoServer1:5544/cam_axis2025_0001/**playback**

Über den RTSP-Server können nur Streams im H.264- oder MPEG4-Format abgerufen werden. Für den Zugriff auf Streams mit anderen Codecs steht der **vimacc®** http-Server zur Verfügung. Dieser stellt alle Stream-Typen in einem einheitlichen, transcodierten MJPEG-Stream bereit.

6 Support / Hotline

Weitere Informationen unter www.vimacc.de

Haben Sie noch Fragen zu **vimacc®**?

Kontaktieren Sie unser Support-Team werktags von 9:00-17:00 MEZ/MESZ

- per Email an **support@accellence.de**
  oder
- telefonisch unter **+49 (0)511 277 2490**

Index