GEUTEBRÜCK

GEVISCOPE actions reference

english
v. 2.95, September 5, 2013

Dmitri Schamschurko

DRAFT
for internal use only

GEUTEBRÜCK GmbH

Im Nassen 7–9
D–53578 Windhagen
Deutschland

Tel: +49 (0) 2645/137–0
Fax: +49 (0) 2645/137–999
Web: www.geutebrueck.de

Abstract

This document is the GEVISCOPE actions reference in the current version 2.95 from September 5, 2013.

This document is part of the GEVISCOPE SDK.

GEUTEBRÜCK

Contents

GEUTEBRÜCK

GEUTEBRÜCK

GEUTEBRÜCK

GEUTEBRÜCK

GEUTEBRÜCK

1 Using GscDBI.DLL

1.1 General

GscDBI.DLL is one of the central communication parts of the GEVISCOPE system. It provides interface for many different purposes like:

- create a connection handle to the GEVISCOPE server
- connect to the server
- read and write system setup
- work with database engine
- send and receive notifications
- etc.

In this manual we describe briefly only that part of this interface, which is requested to process GEVISCOPE actions.

More detailed information can be found in the GEVISCOPE SDK documentation.

1.2 Interface

To establish connection to the GEVISCOPE server and exchange actions between the server and client application following steps should be performed:

- create connection handle
- connect to server
- instantiate PLC object
- instantiate PLC callback
- subscribe requested actions

The following code demonstrates how this can be done in C++:

```
#include <GscDBI.h>            // include import header
using namespace GeviScope;     // and open its namespace

// global objects we use for server and plc
HGscServer    GscServer = 0;
HGscPLC       PlcObject = 0;

void CreateConnectionHandle()
{
  // create global objects
  GscServer = DBICreateRemoteServer(L"our connection");
  PlcObject = GscServer->CreatePLC();
```

```
}

void DestroyConnectionHandle()
{
  // destroy the PLC object
  if (PlcObject)
  {
    delete PlcObject;
    PlcObject = 0;
  }

  // disconnect from server and destroy the server object
  if (GscServer)
  {
    if (GscServer->Connected())
      GscServer->Disconnect(2000);

    delete GscServer;
    GscServer = 0;
  }
}

void ConnectToServer()
{
  // set connection parameter...
  TGscServerConnectParams ConnectParams;
  ConnectParams.Address  = L"localhost";
  ConnectParams.Username = L"sysadmin";
  ConnectParams.Password = DBIEncodeString(L"masterkey");
  GscServer->SetConnectParams(ConnectParams);

  // ...and connect to server
  TConnectResult ConnectResult = GscServer->Connect(0, 0);
  if (ConnectResult != connectOk)
    throw Exception("cannot connect");

  // open push callback...
  PlcObject->OpenPushCallback(PLCCallback, Handle);

  // ...and register all actions, events, and blocking filters
  PlcObject->SubscribeActionsAll();
  PlcObject->SubscribeEventsAll();
  PlcObject->SubscribeBlockingFiltersAll();
}

void DisconnectFromServer()
{
  // close callback...
  PlcObject->CloseCallback();

  // ...and disconnect
  if (GscServer->Connected())
    GscServer->Disconnect(2000);
}
```

The following code demonstrates how this can be done in Delphi:

```
uses ..., GscDBI, ...;          // include import header

// global objects we use for server and plc
var
  GscServer  : HGscServer;
  PlcObject  : HGscPLC;

procedure CreateConnectionHandle;
begin
  // create global objects
  GscServer := DBICreateRemoteServer('our connection');
  PlcObject := GscServer.CreatePLC;
end;

procedure DestroyConnectionHandle;
begin
  // destroy PLC object
  if Assigned(PlcObject) then
  begin
    PlcObject.Free;
    PlcObject := Nil;
  end;

  // disconnect from server and destroy the server object
  if Assigned(GscServer) then
  begin
    if GscServer.Connected then GscServer.Disconnect(2000);

    GscServer.Free;
    GscServer := Nil;
  end;
end;

procedure ConnectToServer;
var
  ConnectResult    : TConnectResult;
  ConnectParams    : TGscServerConnectParams;
begin
  // set connection parameter...
  ClearTGscServerConnectParams(ConnectParams);
  ConnectParams.Address  := PWideChar('localhost');
  ConnectParams.Username := PWideChar('sysadmin');
  ConnectParams.Password := DBIEncodeString('masterkey');
  GscServer.SetConnectParams(ConnectParams);

  // ...and connect to server
  ConnectResult := GscServer.Connect(Nil);
  if ConnectResult <> connectOk then
    raise Exception.Create('cannot connect');

  // open push callback...
  PlcObject.OpenPushCallback(PLCCallback);
```

```
  // ...and register all actions, events, and blocking filters
  PlcObject.SubscribeActionsAll;
  PlcObject.SubscribeEventsAll;
  PlcObject.SubscribeBlockingFiltersAll;
end;

void DisconnectFromServer()
{
  // close callback...
  PlcObject.CloseCallback;

  // ...and disconnect
  if (GscServer.Connected)
    GscServer.Disconnect(2000);
}
```

2 Using GscActions.DLL

2.1 General

GscActions.DLL is responsible for actions. It can be used to

- create action
- decode action's parameters
- destroy action
- marshal and unmarshal action to and from the GscDBI.DLL
- output action as a readable string
- read action from the user input
- etc.

In this manual we describe briefly only how to send actions to the GEVISCOPE server. For further information please reffer GEVISCOPE SDK.

2.2 Interface

To send an action to the server one need to perform the following steps:

- create action
- marshal it the the transport buffer
- send it to the server
- destroy action

The following code demonstrates how this can be done in C++:

```
#include <GscActions.h>            // include import header
#include <GscActionsOutput.h>      // include buffer helper
using namespace GscPLC;            // and open its namespace

void SendCustomAction(const __int64& IntPar, const wchar_t* StrPar)
{
  // create action
  HGscAction anAction = GscAct_CreateCustomAction(IntPar, StrPar);
  if (! anAction)
    throw Exception("cannot create action");

  // marshal it to the binary buffer
  std::vector<byte> Buffer;
  unsigned int Size = Buffer << anAction;

  // ...and send it
  PlcObject->SendAction(Size, Buffer.begin());

  // optionally log to debug console
  std::wstring String;
  String << anAction;
  OutputDebugString(String.c_str());

  // free action handle
  delete anAction;
}
```

The following code demonstrates how this can be done in Delphi:

```
uses ..., GscActions,              // include import header
          GscActionsOutput ...;    // include buffer helper

procedure SendCustomAction(IntPar : Int64; StrPar : PWideChar);
var
  anAction : HGscAction;
  Buffer   : TBinBuffer;
begin
  // create action
  anAction := GscAct_CreateCustomAction(IntPar, StrPar);
  if not Assigned(anAction) then
    raise Exception.Create('cannot create action');

  // marshal it to the binary buffer
  Buffer.Create;
  Buffer.Write(anAction);

  // ...and send it
  PlcObject.SendAction(Buffer.Size, Buffer.GetPtr);

  // optionally log to debug console
  OutputDebugString(anAction.AsWideString);
```

```
  // free buffer and action handle
  Buffer.Free;
  anAction.Free;
end;
```

3 General

3.1 Action classes

All GEVISCOPE actions are divided into the following action classes:

ATM/ACS
ATM and ACS actions.

Audio control
All actions to control the audio streams, also all notifications about the state change of the audio streams.

Backup actions
All actions for backup.

Camera control
All actions to remote control the PTZ heads or camera adjustments.

Cash management actions
All actions for cash management.

Device information
All actions for low-level notification of the device or media channels changes.

Digital contacts
All actions for handling digital inputs and outputs.

Imex
Image export.

LPS
LPS messages.

Lenel
Lenel.

POS
POS (point of sale).

Perimeter protection
Perimeter protection.

Remote export
Remote export.

SKIDATA
SKIDATA messages.

Supply chain security
Supply chain security.

System actions
All actions describing system behaviour.

Video control
All actions to control the video streams, also all notifications about the state change of the video streams.

GEUTEBRÜCK

Viewer actions
All actions for viewer customizing.

Viewer notifications
All viewer notifications.

3.2 Data types

This section describes all data types used as parameter types in GEVISCOPE actions.

3.2.1 __int32

32-bit integer numbers.

C++ : native type `__int32` is used.
Delphi : native type `Integer` is used.

3.2.2 __int64

64-bit integer numbers.

C++ : native type `__int64` is used.
Delphi : native type `Int64` is used.

3.2.3 double

64-bit floating-point numbers.

C++ : native type `double` is used.
Delphi : native type `Double` is used.

3.2.4 TGLibDateTime

Date and time structure. Plese refer to the GEVISCOPE SDK for complete description.

C++ : implemented as `class TGLibDateTime...`
Delphi : implemented as `TGLibDateTime = packed object...`

3.2.5 TPlcRect

PLC rectangle definition.

C++ : implemented as `struct TPlcRect...`
Delphi : implemented as `TPlcRect = packed record...`

3.2.6 TMediaChannelID

Media channel ID structure. Plese refer to the GEVISCOPE SDK for complete description.

C++ : implemented as `struct TPlcMediaChannelID...`
Delphi : implemented as `TPlcMediaChannelID = packed record...`

3.2.7 TEventTypeID

Event type ID structure. Plese refer to the GEVISCOPE SDK for complete description.

C++ : implemented as `struct TPlcEventTypeID...`
Delphi : implemented as `TPlcEventTypeID = packed record...`

3.2.8 TResourceID

Resource ID structure. Plese refer to the GEVISCOPE SDK for complete description.

C++ : implemented as `struct TPlcResourceID...`
Delphi : implemented as `TPlcResourceID = packed record...`

3.2.9 GUID

Globally unique identifier.

C++ : implemented as `GUID`
Delphi : implemented as `TGuid`

3.2.10 string

Null-terminated string of characters. Please note that some special characters (e.g. quote: `"`) are not allowed. There are no escape sequences at the moment.

C++ : native type `const char*` is used.
Delphi : native type `PAnsiChar` is used.

3.2.11 widestring

Null-terminated UNICODE string of characters. Please note that some special characters (e.g. quote: `"`) are not allowed. There are no escape sequences at the moment.

C++ : native type `const wchar_t*` is used.
Delphi : native type `PWideChar` is used.

3.2.12 bool

Boolean values with standard semantics: 0 is false, any other value is true.

C++ : native type `bool` is used.
Delphi : native type `Boolean` is used.

3.2.13 ABCapacityWarning

Auto backup capacity monitoring warning. Enum type, here is the list of possible values:

Name	Value	Description
abwOk	0	no warnings
abwFreeCapacityBelowLimit	1	free capacity below limit
abwAllocatedCapacityAboveLimit	2	allocated capacity above limit
abwBoth	3	free capacity below and allocated capacity above limit

3.2.14 ADArea

Activity detection area. Enum type, here is the list of possible values:

Name	Value	Description
adgArea1	0	area 1
adgArea2	1	area 2
adgArea3	2	area 3
adgArea4	3	area 4

3.2.15 DigitalInputState

State of the digital input contact. Enum type, here is the list of possible values:

Name	Value	Description
disLow	0	low
disMiddle	1	terminated
disHigh	2	high

3.2.16 DigitalOutputState

State of the digital output contact. Enum type, here is the list of possible values:

Name	Value	Description
dosOpen	0	open
dosClose	1	close
dosToggle	2	toggle

3.2.17 GTectClientVCAType

G-Tect client VCA type Enum type, here is the list of possible values:

Name	Value	Description
gvtClientPrivacyMasking	0	client privacy masking
gvtMotionPrivacy	1	motion privacy

3.2.18 GTectSceneAlarmReason

G-Tect alarm reason. Enum type, here is the list of possible values:

Name	Value	Description
garViewFieldObservation	0	view field observation
garContrast	1	contrast

3.2.19 IODeviceType

Specifies the type of the io device. Enum type, here is the list of possible values:

Name	Value	Description
dioDeviceTypeMIO	0	MIO84
dioDeviceTypeIOI43	1	IOI43

3.2.20 IPSwitchOps

IP switch operation. Enum type, here is the list of possible values:

Name	Value	Description
ipoPOEOn	0	PoE on
ipoPOEOff	1	PoE off
ipoPOEReset	2	PoE reset
ipoPOEDelayedReset	3	PoE delayed reset

3.2.21 LenelAccessResult

Lenel access result. Enum type, here is the list of possible values:

Name	Value	Description
larOther	0	access other
larlUnknown	1	access unknown
larGranted	2	access granted
larDenied	3	access denied
larNotApplicable	4	not applicable

GEUTEBRÜCK

3.2.22 LenelEventID

Lenel event ID. Enum type, here is the list of possible values:

Name	Value	Description
leiAccessGranted	1	access granted
leiAccessGrantedonFacilityCode	2	access granted on facility code
leiAccessGrantedNoEntryMade	3	access granted no entry made
leiAccessGrantedonFacilityCodeNoEntryMade	4	access granted on facility code, no entry made
leiInvalidCardFormat	6	invalid card format
leiDeniedCountExceeded	7	denied count exceeded
leiDeniedPINOnlyRequest	8	denied, pin only request
leiInvalidFacilityCode	9	invalid facility code
leiInvalidBadge	10	invalid badge
leiInvalidIssueCode	11	invalid issue code
leiInvalidPINNumber	12	invalid pin number
leiInvalidAccessLevel	13	invalid access level
leiInactiveBadge	14	inactive badge
leiDeniedReaderExcluded	15	denied, reader excluded
leiDeniedNoCommandAuthority	16	denied, no command authority
leiDeniedUnMaskActiveZonesinGroup	17	denied unmask - active zones in group
leiUseLimitExceeded	18	use limit exceeded
leiAccessGrantedUnderDuress	20	access granted under duress
leiAccessDeniedUnderDuress	21	access denied under duress
leiAccessGrantedUnderDuressNoEntryMade	22	access granted under duress - no entry made
leiAntiPassbackViolation	23	anti-passback violation
leiAreaLimitExceeded	24	area limit exceeded
leiTimeoutExceededNoSecondCard	25	timeout exceeded - no second card
leiAreaClosed	26	area closed
leiAccessGrantedAntiPassbackUsed	27	access granted - anti-passback used
leiAccessGrantedAntiPassbackNotUsed	28	access granted - anti-passback not used
leiAntiPassbackInvalidExitReader	29	anti-passback invalid exit reader
leiAntiPassbackInvalidEntryReader	30	anti-passback invalid entry reader
leiNotConfigured	33	not configured
leiAlarmCanceled	34	alarm canceled
leiAlarmActive	35	alarm active
leiDoorForcedOpen	41	door forced open
leiDoorForcedOpenCanceled	42	door forced open canceled
leiDoorHeldOpen	43	door held open
leiDoorHeldOpenCanceled	44	door held open canceled
leiReaderInputTamper	47	reader input tamper
leiReaderInputTamperCanceled	48	reader input tamper canceled
leiDoorContactTamper	49	door contact tamper
leiDoorContactTamperCanceled	50	door contact tamper canceled
leiCabinetTamper	51	cabinet tamper
leiCanceledCabinetTamper	52	canceled cabinet tamper
leiPowerFailure	53	power failure
leiCanceledPowerFailure	54	canceled power failure

leiRelayContactActivated	86	relay contact activated
leiRelayContactDeactivated	87	relay contact deactivated
leiCommunicationsLost	109	communications lost
leiCommunicationsRestored	110	communications restored
leiLocalIOExecutedFunctionList	111	local i/o executed function list
leiFireAlarmIn	220	fire alarm in
leiFireAlarmOut	221	fire alarm out
leiFireAlarmAcknowledge	222	fire alarm acknowledge
leiFireAlarmBlockAcknowledge	223	fire alarm block acknowledge
leiCalltoaBusySubscriber	760	call to a busy subscriber
leiCalltoaPrivateSubscriber	761	call to a private subscriber
leiCalltoanOpenSubscriber	762	call to an open subscriber
leiCallDisconnected	763	call disconnected
leiIntercomFunction	764	intercom function
leiTroubleIn	765	trouble in
leiTroubleOut	766	trouble out
leiTroubleAcknowledge	767	trouble acknowledge
leiDeniedLowBattery	801	denied low battery
leiReaderModeFacilityCode	802	reader mode facility code
leiReaderModeCardOnly	803	reader mode card only
leiReaderModeFirstCardUnlock	804	reader mode first card unlock
leiReaderModeUnlocked	805	reader mode unlocked
leiReaderModeLocked	806	reader mode locked
leiBiometricMismatch	812	biometric mismatch
leiInputMasked	831	input masked
leiInputUnmasked	832	input unmasked
leiDoorForcedOpenMasked	833	door forced open masked
leiDoorForcedOpenUnmasked	834	door forced open unmasked
leiDoorHeldOpenMasked	835	door held open masked
leiDoorHeldOpenUnmasked	836	door held open unmasked
leiReaderModePinorCard	837	reader mode pin or card
leiReaderModeCardandPin	838	reader mode card and pin
leiAcceptedBiometricScore	839	accepted biometric score
leiRejectedBiometricScore	840	rejected biometric score
leiNoBiometricTemplateData	841	no biometric template data
leiPanelOptionsMismatch	842	panel options mismatch
leiAccessGrantedReaderUnlocke	843	access granted: reader unlocked
leiAccessDeniedReaderLocked	844	access denied: reader locked
leiMaxCardholdersReached	845	max cardholders reached
leiMaxAssetsReached	846	max assets reached
leiAccessGrantedGenral	847	access granted
leiAccessDeniedGeneral	848	access denied
leiAccessDeniedDoorSecured	849	access denied door secured
leiAccessDeniedInterlock	850	access denied interlock
leiAccessDeniedPassback	851	access denied passback
leiAccessDeniedUnauthorizedArmingState	852	access denied unauthorized arming state
leiAccessDeniedUnauthorizedEntryLevel	853	access denied unauthorized entry level
leiAccessDeniedUnauthorizedTime	854	access denied unauthorized time
leiSchedulerActionFailed	1502	scheduler action failed

leiSchedulerActionExecuted	1503	scheduler action executed
leiGuardTourActionFailed	1504	guard tour action failed
leiGuardTourActionExecuted	1505	guard tour action executed
leiGlobalLinkageActionFailed	1506	global linkage action failed
leiGlobalLinkageActionExecuted	1507	global linkage action executed
leiDeniedNoHostApproval	1508	denied - no host approval
leiDeniedUnauthorizedAssets	1509	denied - unauthorized assets
leiFirstCardUnlockModeEnabled	1542	first card unlock mode enabled
leiFirstCardUnlockModeDisabled	1543	first card unlock mode disabled
leiExtendedHeldOpenModeEnabled	1544	extended held open mode enabled
leiExtendedHeldOpenModeDisabled	1545	extended held open mode disabled
leiCipherModeEnabled	1546	cipher mode enabled
leiCipherModeDisabled	1547	cipher mode disabled
leiBiometricVerifyModeEnabled	1548	biometric verify mode enabled
leiBiometricVerifyModeDisabled	1549	biometric verify mode disabled
leiAccessDeniedNoBiometricTemplate	1550	access denied: no biometric template
leiAccessDeniedBiometricReaderOffline	1551	access denied: biometric reader offline
leiExtendedHeldCommandDenied	1552	extended held command denied
leiExtendedHeldCommandSetFromReader	1553	extended held command set from reader
leiIncomingCall	1554	incoming call
leiTroubleReport	1555	trouble report
leiTestReport	1556	test report
leiAcknowledgmentActionExecuted	1785	acknowledgment action executed
leiAcknowledgmentActionFailed	1786	acknowledgment action failed
leiAlarmMonitoringActionGroupExecuted	1787	alarm monitoring action group executed
leiAlarmMonitoringActionGroupFailed	1788	alarm monitoring action group failed
leiSmartCardAuthenticationFailed	1789	smart card authentication failed
leiRinging	1794	ringing
leiCallFailed	1795	call failed
leiHold	1796	hold
leiRetrieved	1797	retrieved
leiInitiated	1798	initiated
leiGenericEvent	1799	generic event
leiInputAlarmed	1808	input alarmed
leiInputNormal	1809	input normal
leiInputLowLow	1810	input low low
leiInputLow	1811	input low
leiInputHigh	1812	input high
leiInputHighHigh	1813	input high high
leiDoorOpen	1887	door open
leiDoorClose	1888	door close
leiCommunicationsWithHostLost	1889	communications with host lost
leiCommunicationsWithHostRestored	1890	communications with host restored
leiCallEnded	1891	call ended
leiCallEstablished	1892	call established
leiUnansweredCall	1893	unanswered call
leiCallTransferred	1894	call transferred

leiCallConferenced	1895	call conferenced
leiAccessDeniedEscortTimeoutExpired	1896	access denied: escort timeout expired
leiCallQueued	1910	call queued
leiAccessDeniedAreaOccupied	1911	access denied: area occupied
leiAccessDeniedNoOccupantApproval	1912	access denied: no occupant approval
leiDeniedBadgeNotinPanel	1914	denied, badge not in panel
leiIntercomExchangeFailure	1915	intercom exchange failure
leiIntrusionCommandDenied	1920	intrusion command denied
leiIntrusionCommandAccepted	1921	intrusion command accepted
leiAlarmMaskGroupArmed	1922	alarm mask group armed
leiAlarmMaskGroupDisarmed	1923	alarm mask group disarmed
leiAlarmMaskGroupForceArmed	1924	alarm mask group force armed
leiAlarmMaskGroupMaskCountIncremented	1925	alarm mask group mask count incremented
leiAlarmMaskGroupMaskCountDecremented	1926	alarm mask group mask count decremented
leiAlarmMaskGroupArmingFailureActivePoints	1927	alarm mask group arming failure, active points
leiAccessDeniedAreaEmpty	1928	access denied: area empty
leiElevatorTerminalModeDefaultFloororUserEntryofDestinationFloor	1936	elevator terminal mode default floor or user entry of destination floor
leiUnknownElevatorTerminal	1937	unknown elevator terminal
leiOpenDoorCommandIssuedDoorUsed	1938	open door command issued - door used
leiOpenDoorCommandIssuedDoorNotUsed	1939	open door command issued - door not used
leiRequesttoExitDoorUsed	1940	request to exit - door used
leiRequesttoExitDoorNotUsed	1941	request to exit - door not used
leiElevatorTerminalModeDefaultFloor	1943	elevator terminal mode default floor
leiElevatorTerminalModeAccesstoAuthorizedFloors	1944	elevator terminal mode access to authorized floors
leiElevatorTerminalModeUserEntryofDestinationFloor	1945	elevator terminal mode user entry of destination floor
leiReaderOffline	1968	reader offline
leiReaderOfflineRestored	1969	reader offline restored
leiUnknownUserCommand	1978	unknown user command
leiAccessDeniedAssetRequired	1984	access denied: asset required
leiAccessDeniedSpecial	2289	access denied

3.2.23 LenelEventType

Lenel event type. Enum type, here is the list of possible values:

Name	Value	Description
letAccessGranted	0	access granted
letAccessDenied	1	access denied
letDuress	2	duress
letAreaAPB	3	area APB

letSystem	4	system
letAsset	5	asset
letHostMessages	6	host messages
letFireSeven	7	Fire
letFireEight	8	Fire
letFireNine	9	Fire
letIntercom	10	intercom
letVideo	11	video
letTransmitter	12	transmitter
letBiometric	18	biometric
letTrouble	19	trouble
letDigitize	20	digitize
letBurglary	21	burglary
letTemperature	22	temperature
letGas	23	gas
letRelaySounder	24	relay/sounder
letMedical	25	medical
letWater	26	water
letC900	27	C900
letOpenClose	28	open/close
letMuster	29	muster
letGeneric	30	generic
letPointofSale	31	point of sale
letPortableProgrammer	32	portable programmer

3.2.24 PPAlarmState

PP alarm state. Enum type, here is the list of possible values:

Name	Value	Description
pasNormal	0	normal
pasAlarm	1	alarm

3.2.25 PPCableKind

PP cable kind. Enum type, here is the list of possible values:

Name	Value	Description
pckCableA	0	cable A
pckCableB	1	cable B

3.2.26 PPSensorKind

PP sensor kind. Enum type, here is the list of possible values:

Name	Value	Description
pskCableAFault	0	cable A fault
pskCableBFault	1	cable B fault

pskTamper	2	tamper
pskTrap	3	trap
pskSensor	4	sensor
pskPath	5	path
pskAux	6	aux
pskService	7	service

3.2.27 PlcBacklightMode

Backlight compensation mode. Enum type, here is the list of possible values:

Name	Value	Description
pbmOff	0	off
pbmOn	1	on

3.2.28 PlcBackupFormat

Backup format. Enum type, here is the list of possible values:

Name	Value	Description
ebfDefault	0	default
ebfGBF	1	GBF
ebfMPEG2	2	MPEG2

3.2.29 PlcClientType

Client type. Enum type, here is the list of possible values:

Name	Value	Description
pctInvalid	0	invalid
pctGscView	1	GSCView
pctGscSetup	2	GSCSetup
pctGscRegEdit	3	GSCRegEdit
pctGscWeb	4	GSCWeb
pctDefault	5	default
pctGscSupplyChain	6	GSCSupplyChain

3.2.30 PlcDatabaseRing

Database ring. Enum type, here is the list of possible values:

Name	Value	Description
dbrRing1	0	ring 1
dbrRing2	1	ring 2
dbrRing3	2	ring 3
dbrRing4	3	ring 4
dbrRing5	4	ring 5

dbrRing6	5	ring 6
dbrRing7	6	ring 7
dbrRing8	7	ring 8

3.2.31 PlcDatabaseStatus

Database status. Enum type, here is the list of possible values:

Name	Value	Description
dbsEmpty	0	empty
dbsReady	1	ready
dbsFileAdded	2	file added

3.2.32 PlcDayNightMode

Day/night mode. Enum type, here is the list of possible values:

Name	Value	Description
pdnDay	0	day
pdnNight	1	night
pdnAuto	2	auto

3.2.33 PlcExportAbort

Export abort flag. Enum type, here is the list of possible values:

Name	Value	Description
exaUserAbort	0	user abort
exaLowDiscSpace	1	low disc space
exaNoUserRights	2	no user rights
exaError	3	error

3.2.34 PlcExportMarker

Export marker. Enum type, here is the list of possible values:

Name	Value	Description
exmSelectionBegin	0	selection begin
exmSelectionEnd	1	selection end

3.2.35 PlcExportSuccess

Export success state. Enum type, here is the list of possible values:

Name	Value	Description
exsSuccess	0	success

exsFailed	1	failed
exsAborted	2	aborted
exsLowDiscSpace	3	low disc space
exsNoUserRights	4	no user rights
exsQueueSizeReached	5	queue size reached
exsConnectionError	6	connection error

3.2.36 PlcFRCNotification

FRC notification. Enum type, here is the list of possible values:

Name	Value	Description
frnDataOverwritten	0	data overwritten
frnBufferActivated	1	buffer recording activated
frnReconcileStarted	2	reconcile operation started
frnReconcileProgress	3	reconcile progress
frnReconcileAborted	4	reconcile aborted
frnReconcileFinished	5	reconcile finished

3.2.37 PlcImageExportType

Image export type. Enum type, here is the list of possible values:

Name	Value	Description
ietSingleImage	0	single image
ietBackupFile	1	backup file
ietVideoFile	2	video file
ietClipboard	3	clipboard
ietPrinter	4	printer

3.2.38 PlcLpsStatus

Position query status. Enum type, here is the list of possible values:

Name	Value	Description
lstPositionOK	0	tag and position is valid
lstUnknownArea	1	tag is valid but position is not known
lstLastPosition	2	tag is valid but position is last known position
lstInvalidID	3	tag is invalid
lstError	4	system error or server not connected

3.2.39 PlcMessageCode

Code of the message. Enum type, here is the list of possible values:

Name	Value	Description

pmcInternalError	0	internal error
pmcOutOfResources	1	out of resources
pmcUnlicensed	2	unlicensed
pmcConnectionLost	3	connection lost
pmcUnknownError	4	unknown error
pmcDemoModeExpired	5	demo mode expired
pmcDatabaseFileMissing	10	database file missing
pmcFileAccessError	11	file access error
pmcFLTMInefficient	12	FLTM inefficient
pmcDBECouldNotFreeAnyPage	13	DBE could not free any page
pmcDongleFound	20	dongle found
pmcDongleMissing	21	dongle missing
pmcDongleWrong	22	wrong dongle connected
pmcDongleInsufficient	23	insufficient licences
pmcFirmwareUploadFailed	30	firmware upload failed
pmcWatchdogReset	31	watchdog reset detected
pmcHttpError	35	HTTP error
pmcHostRestrictions	36	not a white-listed host
pmcInsufficientFrameRate	40	not enough images for analysis
pmcTranscoderOverload	41	transcoder is overloaded
pmcNoControlForLongTime	50	had no control for long time
pmcBehaviouralRuleTooDeep	51	behavioural rule iteration too deep
pmcCannotOpenPort	61	cannot open port
pmcCannotEnumeratePorts	62	cannot enumerate ports
pmcCannotSetFilter	63	cannot set filter
pmcDataOverflow	64	data overflow
pmcInsufficientData	65	insufficient data
pmcIODeviceFound	70	device found
pmcIODeviceLost	71	device lost
pmcIODeviceNotFound	72	device not found
pmcIOTemperatureBelow	73	temperature below threshold
pmcIOTemperatureAbove	74	temperature above threshold
pmcIODriverNotInstalled	75	driver is not installed
pmcIOCannotEnumerate	76	cannot enumerate hardware
pmcIOCannotOpen	77	cannot open device
pmcIODuplcatedAddress	78	device with duplicated address switch
pmcIOUnusedDevice	79	unused device

3.2.40 PlcMessageSource

Source of the message. Enum type, here is the list of possible values:

Name	Value	Description
pmsServer	0	server
pmsSetup	1	setup
pmsDBE	2	DBE
pmsMediaAPI	3	media API
pmsMediaPlugin	4	media plugin
pmsPLC	5	PLC

pmsAutoBackup	6	auto backup
pmsDongle	7	dongle
pmsFRC	8	FRC
pmsATM	9	ATM
pmsSensor	10	sensor
pmsPOS	11	POS
pmsACS	12	ACS
pmsIOI43	13	IOI43
pmsInterface	14	interface
pmsTranscoder	15	transcoder
pmsLenel	16	LENEL
pmsLogistic	17	logistic
pmsLPS	18	LPS
pmsSCS	19	Supply Chain Security
pmsVAM	20	Vehicle Access Manager
pmsMIO	21	MIO
pmsPP	22	perimeter protection

3.2.41 PlcNPRRestriction

PLC POS status. Enum type, here is the list of possible values:

Name	Value	Description
nrrUnspecified	0	unspecified
nrrBlackListed	1	black-listed
nrrWhiteListed	2	white-listed

3.2.42 PlcObtrackCounterType

OBTRACK counter type. Enum type, here is the list of possible values:

Name	Value	Description
octIncoming	0	incoming
octOutcoming	1	outcoming
octCommon	2	common

3.2.43 PlcObtrackExceedingDirection

OBTRACK exceeding direction. Enum type, here is the list of possible values:

Name	Value	Description
oedExceed	0	exceed
oedFallBelow	1	fall below

3.2.44 PlcObtrackObjectClass

OBTRACK object class. Enum type, here is the list of possible values:

Name	Value	Description
oocPerson	0	person
oocCar	1	car
oocUndefined	2	unknown object

3.2.45 PlcObtrackObjectStatus

OBTRACK object status. Enum type, here is the list of possible values:

Name	Value	Description
oosStarting	0	starting
oosStationary	1	stationary
oosLeaving	2	leaving
oosUndefined	3	undefined
oosDeleted	4	deleted

3.2.46 PlcObtrackProcessSize

OBTRACK process size. Enum type, here is the list of possible values:

Name	Value	Description
opsCIF	0	CIF
opsFourCIF	1	4CIF
opsTakeWhatYouGet	2	ANY

3.2.47 PlcPOSStatus

PLC POS status. Enum type, here is the list of possible values:

Name	Value	Description
posStarted	0	started
posCleared	1	cleared
posStopped	2	stopped

3.2.48 PlcPluginError

Plugin error code. Enum type, here is the list of possible values:

Name	Value	Description
ppeInternalError	1	internal error
ppeOutOfResources	2	out of resources
ppeUnlicensed	3	unlicensed
ppeConnectionLost	4	connection lost
ppeUnknownError	5	unknown error

3.2.49 PlcPluginState

Plugin state. Enum type, here is the list of possible values:

Name	Value	Description
ppsInternalState	0	internal state
ppsConnected	1	connected
ppsDisconnected	2	disconnected

3.2.50 PlcPumpStatus

PLC pump status. Enum type, here is the list of possible values:

Name	Value	Description
pumStarted	0	filling started
pumStopped	1	filling stopped
pumCleared	2	pump released
pumAmount	3	amount message

3.2.51 PlcResourceChangeKind

PLC resource change kind. Enum type, here is the list of possible values:

Name	Value	Description
rckAdded	0	added
rckRemoved	1	removed
rckModified	2	modified

3.2.52 PlcResourceKind

PLC resource kind. Enum type, here is the list of possible values:

Name	Value	Description
prkNotSpecified	0	unknown
prkVideoInput	1	video input
prkPTZHead	2	PTZ head
prkDigitalInput	3	digital input
prkDigitalOutput	4	digital output
prkUser	5	user
prkTimeRange	6	time range
prkVCUserOptions	7	VC user options
prkVCAllProfiles	8	VC all profiles

3.2.53 PlcSceneStoreModificationType

Scene store modification type. Enum type, here is the list of possible values:

Name	Value	Description

smtCreate	0	create
smtOpen	1	open
smtDisplay	2	display
smtClose	3	close
smtDelete	4	delete

3.2.54 PlcSkidataControl

SKIDATA control message. Enum type, here is the list of possible values:

Name	Value	Description
sdcConnectionOk	1	connection ok
sdcConnectionFailed	2	connection attempt failed
sdcDisconnected	3	disconnected
sdcConnectionLost	4	connection lost

3.2.55 PlcSkidataMsgCodeEntry

SKIDATA message entry. Enum type, here is the list of possible values:

Name	Value	Description
sdenTicket	1	SKIDATA ticket
sdenCreditCard	2	credit card
sdenElectronicPurse	3	el. purse card
sdenExternalCard	4	external card

3.2.56 PlcSkidataMsgCodeExit

SKIDATA message exit. Enum type, here is the list of possible values:

Name	Value	Description
sdexTicket	1	SKIDATA ticket
sdexCreditCard	2	credit card
sdexElectronicPurse	3	el. purse card
sdexExternalCard	4	external card

3.2.57 PlcSkidataMsgCodeTransaction

SKIDATA message transaction. Enum type, here is the list of possible values:

Name	Value	Description
sdtrParking	1	parking payment
sdtrSale	2	sale payment
sdtrOther	3	other payment
sdtrCash	4	cash payment
sdtrCheck	5	cheque payment
sdtrCreditCard	6	credit card payment

sdtrInvoice	7	invoice payment
sdtrValueCard	8	value card payment
sdtrValidation	9	validation payment
sdtrToken	10	token payment
sdtrElectronicPurse	11	electronic purse payment
sdtrIsoDiscount	12	ISO discount payment

3.2.58 PlcSpecialConstants

Special constants. Enum type, here is the list of possible values:

Name	Value	Description
pscRestOfTime	0	rest of time

3.2.59 PlcTunnelAlarmReason

Tunnel alarm reason. Enum type, here is the list of possible values:

Name	Value	Description
tarPerson	0	person
tarPark	1	park
tarWrongDirection	2	wrong direction

3.2.60 PlcViewerAlarmPlayMode

Viewer alarm play mode. Enum type, here is the list of possible values:

Name	Value	Description
apmDefault	0	show alarm using default settings
apmLiveReplay	1	live replay
apmReplayEventPictures	2	replay event pictures
apmContinuousEventReplayInALoop	3	continuous event replay
apmShowFirstAlarmPictureOnly	4	show first alarm picture only

3.2.61 PlcViewerAlarmQueueSelection

Viewer alarm queue selection mode. Enum type, here is the list of possible values:

Name	Value	Description
aqsFirst	0	first
aqsLast	1	last
aqsNext	2	next
aqsPrevious	3	previous

3.2.62 PlcViewerAlarmState

Viewer alarm state notification. Enum type, here is the list of possible values:

Name	Value	Description
vasNewAlarm	0	new alarm
vasPresented	1	presented
vasStacked	2	stacked
vasConfirmed	3	confirmed
vasRemoved	4	removed
vasLastConfirmed	5	last confirmed
vasLastRemoved	6	last removed
vasListConfirmed	7	list confirmed
vasListEmpty	8	list empty

3.2.63 PlcViewerPlayMode

Viewer play mode. Enum type, here is the list of possible values:

Name	Value	Description
vpmUnknownMode	0	unknown
vpmPlayStop	1	play stop
vpmPlayForward	2	play forward
vpmPlayBackward	3	play backward
vpmPlayFastForward	4	fast forward
vpmPlayFastBackward	5	fast backward
vpmPlayStepForward	6	step forward
vpmPlayStepBackward	7	step backward
vpmPlayBOD	8	play BOD
vpmPlayEOD	9	play EOD
vpmPlayQuasiLive	10	quasi live
vpmPlayStream	11	live
vpmPlayNextEvent	12	next event
vpmPlayPrevEvent	13	prev event
vpmPeekLivePicture	14	peek live picture
vpmPlayNextMOS	17	next detected motion
vpmPlayPrevMOS	18	prev detected motion

3.2.64 SafebagStep

Safebag step. Enum type, here is the list of possible values:

Name	Value	Description
sbsOpen	0	open

3.2.65 SystemKey

GEVISCOPE system key. Enum type, here is the list of possible values:

Name	Value	Description
keyDisplay1	0	GSC display 1
keyDisplay2	1	GSC display 2
keyDisplay3	2	GSC display 3
keyDisplay4	3	GSC display 4
keyReporter1	256	reporter 1

3.2.66 SystemLED

System LED. Enum type, here is the list of possible values:

Name	Value	Description
ledError	0	error LED
ledRecord	1	record LED
ledGreen	2	green LED
ledYellow	3	yellow LED
ledIOI43a1	9	IOI43a LED 1
ledIOI43a2	10	IOI43a LED 2
ledIOI43a3	11	IOI43a LED 3
ledIOI43a4	12	IOI43a LED 4

3.2.67 TrafficDirection

Traffic direction. Enum type, here is the list of possible values:

Name	Value	Description
tfdIn	0	in
tfdOut	1	out
tfdUnspecified	2	unknown

3.2.68 UserLoginFailureCode

User login failure code. Enum type, here is the list of possible values:

Name	Value	Description
ulfUser1NotFound	1	unknown first user
ulfUser1InvalPwd	2	invalid first user
ulfUser2NotFound	3	unknown second user
ulfUser2InvalPwd	4	invalid second user
ulfMissingSecondUser	5	second user required
ulfUser2NotAdmitted	6	second user is not admitted
ulfNoLicense	7	no licence available
ulfTooManyClients	8	licence limits reached

3.2.69 VMDCycle

VMD measure cycle. Enum type, here is the list of possible values:

Name	Value	Description
vmc40ms	0	40 ms
vmc160ms	1	160 ms
vmc640ms	2	640 ms
vmc2_5s	3	2.5 s
vmc10s	4	10 s

3.2.70 VMDGroup

VMD group. Enum type, here is the list of possible values:

Name	Value	Description
vmgGroup1	0	group 1
vmgGroup2	1	group 2
vmgGroup3	2	group 3
vmgGroup4	3	group 4

3.2.71 VideoInterlaceType

Interlace kind of the video input signal. Enum type, here is the list of possible values:

Name	Value	Description
vsdInterlaced	0	interlaced
vsdNonInterlaced	1	non-interlaced

3.2.72 VideoSensorKind

Video sensor type. Enum type, here is the list of possible values:

Name	Value	Description
vskAD	0	AD
vskVMD	1	VMD
vskCPA	2	CPA
vskIPAD	3	IP-AD
vskOBTRACK	4	OBTRACK
vskDUAL	5	DUAL
vskNPR	6	NPR
vskFR	7	FR
vskVAMissing	8	VA-Missing
vskCPAExt	9	CPA extern
vskGTectVMX	10	G-Tect/VMX

3.2.73 VideoSignalNorm

Norm of the video input signal. Enum type, here is the list of possible values:

Name	Value	Description
vsnEIA	0	EIA
vsnCCIR	1	CCIR

3.2.74 VideoSignalType

Type of the video input signal. Enum type, here is the list of possible values:

Name	Value	Description
vstBAS	0	BAS
vstFBAS	1	FBAS

GEUTEBRÜCK

4 ATM/ACS

ATM and ACS actions.

4.1 ACS access denied

ACSAccessDenied (ACSName, *ACSNo*, *Account*, *BancCode*, *CardNo*, *TimeStamp*, *Reason*)

Description : *ACS access denied.*

Code : ac_ACSAccessDenied (357)

Class : ak_ATM (6)

Parameters :

ACSName (ACS) :
Type : widestring
Description : ACS name.

ACSNo (ACS no) [optional] :
Type : __int32
Description : ACS no.

Account (account) [optional] :
Type : __int64
Description : Account no.

BancCode (bank code) [optional] :
Type : __int64
Description : Bank code.

CardNo (card no) [optional] :
Type : widestring
Description : Card no.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

Reason (reason) [optional] :
Type : __int32
Description : Reason.

Text

```
ACSAccessDenied ("ACS", ACSNo:  32, Account:  64,
BancCode:  64, CardNo:  "card no", TimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00", Reason:  32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateACSAccessDenied(
  const wchar_t* aACSName,
  const __int32* aACSNo,
  const __int64* aAccount,
  const __int64* aBancCode,
  const wchar_t* aCardNo,
  const TGLibDateTime* aTimeStamp,
  const __int32* aReason);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeACSAccessDenied(
  ConstHGscAction anAction,
  const wchar_t*& aACSName,
  const __int32*& aACSNo,
  const __int64*& aAccount,
  const __int64*& aBancCode,
  const wchar_t*& aCardNo,
  const TGLibDateTime*& aTimeStamp,
  const __int32*& aReason);
```

Delphi – Create

```
function GscAct_CreateACSAccessDenied(
  aACSName: PWideChar;
  aACSNo: PInteger;
  aAccount: PInt64;
  aBancCode: PInt64;
  aCardNo: PWideChar;
  aTimeStamp: PTGLibDateTime;
  aReason: PInteger)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeACSAccessDenied(
  const anAction: HGscAction;
  out aACSName: PWideChar;
  out aACSNo: PInteger;
  out aAccount: PInt64;
  out aBancCode: PInt64;
  out aCardNo: PWideChar;
  out aTimeStamp: PTGLibDateTime;
  out aReason: PInteger)
  : Boolean; stdcall; external GscActionsDll;
```

4.2 ACS access granted

ACSAccessGranted (ACSName, *ACSNo*, *Account*, *BancCode*, *CardNo*, *TimeStamp*)

Description : *ACS access granted.*

Code : ac_ACSAccessGranted (356)

Class : ak_ATM (6)

Parameters :

ACSName (ACS) :
Type : widestring
Description : ACS name.

ACSNo (ACS no) [optional] :
Type : __int32
Description : ACS no.

Account (account) [optional] :
Type : __int64
Description : Account no.

BancCode (bank code) [optional] :
Type : __int64
Description : Bank code.

CardNo (card no) [optional] :
Type : widestring
Description : Card no.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

Text

```
ACSAccessGranted ("ACS", ACSNo:  32, Account:  64,
BancCode:  64, CardNo:  "card no", TimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00")
```

C++ Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateACSAccessGranted(
  const wchar_t* aACSName,
  const __int32* aACSNo,
  const __int64* aAccount,
  const __int64* aBancCode,
  const wchar_t* aCardNo,
  const TGLibDateTime* aTimeStamp);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeACSAccessGranted(
  ConstHGscAction anAction,
  const wchar_t*& aACSName,
  const __int32*& aACSNo,
  const __int64*& aAccount,
  const __int64*& aBancCode,
  const wchar_t*& aCardNo,
  const TGLibDateTime*& aTimeStamp);
```

Delphi – Create

```
function GscAct_CreateACSAccessGranted(
  aACSName: PWideChar;
  aACSNo: PInteger;
  aAccount: PInt64;
  aBancCode: PInt64;
  aCardNo: PWideChar;
  aTimeStamp: PTGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeACSAccessGranted(
  const anAction: HGscAction;
  out aACSName: PWideChar;
  out aACSNo: PInteger;
  out aAccount: PInt64;
  out aBancCode: PInt64;
  out aCardNo: PWideChar;
  out aTimeStamp: PTGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

4.3 ACS raw answer

ACSRawAnswer (ACSName, TimeStamp, ACSData)

Description : *ACS raw answer.*

Code : `ac_ACSRawAnswer` (355)

Class : `ak_ATM` (6)

Parameters :

ACSName (ACS) :
Type : widestring
Description : ACS name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ACSData (answer) :
Type : widestring
Description : ACS answer.

Text

```
ACSRawAnswer ("ACS", "2013/09/05 14:59:59,999 GMT+02:00",
"answer")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateACSRawAnswer(
  const wchar_t* aACSName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aACSData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeACSRawAnswer(
  ConstHGscAction anAction,
  const wchar_t*& aACSName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aACSData);
```

Delphi – Create

```
function GscAct_CreateACSRawAnswer(
  aACSName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aACSData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeACSRawAnswer(
  const anAction: HGscAction;
  out aACSName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aACSData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

4.4 ACS raw data

ACSRawData (ACSName, TimeStamp, ACSData)

Description : *ACS raw data.*

Code : ac_ACSRawData (354)

Class : ak_ATM (6)

Parameters :

ACSName (ACS) :
Type : widestring
Description : ACS name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ACSData (data) :
Type : widestring
Description : ACS data.

Text

```
ACSRawData ("ACS", "2013/09/05 14:59:59,999 GMT+02:00",
"data")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateACSRawData(
  const wchar_t* aACSName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aACSData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeACSRawData(
  ConstHGscAction anAction,
  const wchar_t*& aACSName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aACSData);
```

Delphi – Create

```
function GscAct_CreateACSRawData(
  aACSName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aACSData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeACSRawData(
  const anAction: HGscAction;
  out aACSName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aACSData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

4.5 ATM raw answer

ATMRawAnswer (ATMName, TimeStamp, ATMData)

Description : *ATM raw answer.*

Code : ac_ATMRawAnswer (351)

Class : ak_ATM (6)

Parameters :

ATMName (ATM) :
Type : widestring
Description : ATM name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ATMData (answer) :
Type : widestring
Description : ATM answer.

Text

```
ATMRawAnswer ("ATM", "2013/09/05 14:59:59,999 GMT+02:00",
"answer")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateATMRawAnswer(
  const wchar_t* aATMName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aATMData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeATMRawAnswer(
  ConstHGscAction anAction,
  const wchar_t*& aATMName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aATMData);
```

Delphi – Create

```
function GscAct_CreateATMRawAnswer(
  aATMName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aATMData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeATMRawAnswer(
  const anAction: HGscAction;
  out aATMName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aATMData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

4.6 ATM raw data

ATMRawData (ATMName, TimeStamp, ATMData)

Description : *ATM raw data.*

Code : ac_ATMRawData (350)

Class : ak_ATM (6)

Parameters :

ATMName (ATM) :
Type : widestring
Description : ATM name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ATMData (data) :
Type : widestring
Description : ATM data.

GEUTEBRÜCK

Text

```
ATMRawData ("ATM", "2013/09/05 14:59:59,999 GMT+02:00",
"data")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateATMRawData(
  const wchar_t* aATMName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aATMData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeATMRawData(
  ConstHGscAction anAction,
  const wchar_t*& aATMName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aATMData);
```

Delphi – Create

```
function GscAct_CreateATMRawData(
  aATMName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aATMData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeATMRawData(
  const anAction: HGscAction;
  out aATMName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aATMData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

4.7 ATM transaction

ATMTransaction (ATMName, NewTransaction, Photostep, *ATMNo*, *Account*, *BancCode*, *CardNo*, *TAN1*, *TAN2*, *TimeStamp1*, *TimeStamp2*, *Amount*, *Currency*)

Description : *ATM transaction.*

Code : ac_ATMTransaction (352)

Class : ak_ATM (6)

Parameters :

ATMName (ATM) :
Type : widestring
Description : ATM name.

NewTransaction (new transaction) :
Type : bool
Description : New transaction.

Photostep (photostep) :
Type : __int32
Description : Photostep.

ATMNo (ATM no) [optional] :
Type : __int32
Description : ATM no.

Account (account) [optional] :
Type : __int64
Description : Account no.

BancCode (bank code) [optional] :
Type : __int64
Description : Bank code.

CardNo (card no) [optional] :
Type : widestring
Description : Card no.

TAN1 (tan 1) [optional] :
Type : __int64
Description : TAN 1.

TAN2 (tan 2) [optional] :
Type : __int64
Description : TAN 2.

TimeStamp1 (time stamp 1) [optional] :
Type : TGLibDateTime
Description : Time stamp 1.

TimeStamp2 (time stamp 2) [optional] :
Type : TGLibDateTime
Description : Time stamp 2.

Amount (amount) [optional] :
Type : double
Description : Amount.

Currency (currency) [optional] :
Type : widestring
Description : Currency.

GEUTEBRÜCK

Text

```
ATMTransaction ("ATM", 1, 32, ATMNo: 32, Account: 64,
BancCode: 64, CardNo: "card no", TAN1: 64, TAN2:
64, TimeStamp1: "2013/09/05 14:59:59,999 GMT+02:00",
TimeStamp2: "2013/09/05 14:59:59,999 GMT+02:00", Amount:
0.0, Currency: "currency")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateATMTransaction(
  const wchar_t* aATMName,
  const bool aNewTransaction,
  const __int32 aPhotostep,
  const __int32* aATMNo,
  const __int64* aAccount,
  const __int64* aBancCode,
  const wchar_t* aCardNo,
  const __int64* aTAN1,
  const __int64* aTAN2,
  const TGLibDateTime* aTimeStamp1,
  const TGLibDateTime* aTimeStamp2,
  const double* aAmount,
  const wchar_t* aCurrency);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeATMTransaction(
  ConstHGscAction anAction,
  const wchar_t*& aATMName,
  bool& aNewTransaction,
  __int32& aPhotostep,
  const __int32*& aATMNo,
  const __int64*& aAccount,
  const __int64*& aBancCode,
  const wchar_t*& aCardNo,
  const __int64*& aTAN1,
  const __int64*& aTAN2,
  const TGLibDateTime*& aTimeStamp1,
  const TGLibDateTime*& aTimeStamp2,
  const double*& aAmount,
  const wchar_t*& aCurrency);
```

Delphi – Create

```
function GscAct_CreateATMTransaction(
  aATMName: PWideChar;
  aNewTransaction: Boolean;
  aPhotostep: Integer;
  aATMNo: PInteger;
  aAccount: PInt64;
  aBancCode: PInt64;
  aCardNo: PWideChar;
  aTAN1: PInt64;
  aTAN2: PInt64;
  aTimeStamp1: PTGLibDateTime;
  aTimeStamp2: PTGLibDateTime;
  aAmount: PDouble;
  aCurrency: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeATMTransaction(
  const anAction: HGscAction;
  out aATMName: PWideChar;
  out aNewTransaction: Boolean;
  out aPhotostep: Integer;
  out aATMNo: PInteger;
  out aAccount: PInt64;
  out aBancCode: PInt64;
  out aCardNo: PWideChar;
  out aTAN1: PInt64;
  out aTAN2: PInt64;
  out aTimeStamp1: PTGLibDateTime;
  out aTimeStamp2: PTGLibDateTime;
  out aAmount: PDouble;
  out aCurrency: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

5 Audio control

All actions to control the audio streams, also all notifications about the state change of the audio streams.

5.1 ABC connect

ABCConnect (Address)

Description : *Connect audio back channel.*

Code : ac_ABCConnect (21)

Class : ak_Audio (2)

Parameters :

Address (address) :
Type : widestring
Description : Address of the remote server.

Text

```
ABCConnect ("address")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateABCConnect(
  const wchar_t* aAddress);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeABCConnect(
  ConstHGscAction anAction,
  const wchar_t*& aAddress);
```

Delphi – Create

```
function GscAct_CreateABCConnect(
  aAddress: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeABCConnect(
  const anAction: HGscAction;
  out aAddress: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

5.2 ABC disconnect

ABCDisconnect ()

Description : *Disconnect audio back channel.*

Code : ac_ABCDisconnect (22)

Class : ak_Audio (2)

This action has no parameters.

Text

```
ABCDisconnect ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateABCDisconnect();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeABCDisconnect(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateABCDisconnect()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeABCDisconnect(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

5.3 ABC play file

ABCPlayFile (FileID, FileName, AutoRepeat)

Description : *Play file on audio back channel.*

Code : ac_ABCPlayFile (23)

Class : ak_Audio (2)

Parameters :

FileID (file ID) :
Type : __int64
Description : File ID.

FileName (file name) :
Type : widestring
Description : Name of the file.

AutoRepeat (repeat) :
Type : bool
Description : Repeat file automatically

Text

```
ABCPlayFile (64, "file name", 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateABCPlayFile(
  const __int64& aFileID,
  const wchar_t* aFileName,
  const bool aAutoRepeat);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeABCPlayFile(
  ConstHGscAction anAction,
  __int64& aFileID,
  const wchar_t*& aFileName,
  bool& aAutoRepeat);
```

Delphi – Create

```
function GscAct_CreateABCPlayFile(
  var aFileID: Int64;
  aFileName: PWideChar;
  aAutoRepeat: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeABCPlayFile(
  const anAction: HGscAction;
  out aFileID: Int64;
  out aFileName: PWideChar;
  out aAutoRepeat: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

5.4 Sensor audio alarm

SensorAudioAlarm (Channel)

Description : *Audio alarm detected.*

Code : ac_SensorAudioAlarm (20)

Class : ak_Audio (2)

Parameters :

Channel (channel) [AudioInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
SensorAudioAlarm (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSensorAudioAlarm(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSensorAudioAlarm(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateSensorAudioAlarm(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSensorAudioAlarm(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

6 Backup actions

All actions for backup.

6.1 Abort all auto backups

AbortAllAutoBackups ()

Description : *Abort all auto backups.*

Code : ac_AbortAllAutoBackups (299)

Class : ak_Backup (7)

This action has no parameters.

Text

```
AbortAllAutoBackups ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAbortAllAutoBackups();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAbortAllAutoBackups(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateAbortAllAutoBackups()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAbortAllAutoBackups(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

6.2 Abort auto backup

AbortAutoBackup (Schedule)

Description : *Abort auto backup.*

Code : ac_AbortAutoBackup (298)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

Text

```
AbortAutoBackup ("schedule")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAbortAutoBackup(
  const TPlcResourceID& aSchedule);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAbortAutoBackup(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule);
```

Delphi – Create

```
function GscAct_CreateAbortAutoBackup(
  var aSchedule: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAbortAutoBackup(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

6.3 Auto backup capacity file auto deleted

AutoBackupCapacityMonitoringFileAutoDeleted (Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByGbf, PercentFree, PercentAllocated, PercentAllocatedByGbf, FileSize, FileName)

GEUTEBRÜCK

Description : *Auto backup capacity monitoring: file auto deleted.*

Code : `ac_AutoBackupCapacityMonitoringFileAutoDeleted` (274)

Class : `ak_Backup` (7)

Parameters :

Warning (warning) :
Type : ABCapacityWarning
Description : Warning.

Destination (destination) :
Type : TResourceID
Description : Destination.

TotalCapacity (total capacity) :
Type : __int64
Description : Total capacity.

FreeCapacity (free capacity) :
Type : __int64
Description : Free capacity.

AllocatedByGbf (allocated by GBF) :
Type : __int64
Description : Allocated by GBF.

PercentFree (percent free) :
Type : __int32
Description : Percent free.

PercentAllocated (percent allocated) :
Type : __int32
Description : Percent allocated.

PercentAllocatedByGbf (percent allocated by GBF) :
Type : __int32
Description : Percent allocated by GBF.

FileSize (file size) :
Type : __int64
Description : File size.

FileName (file name) :
Type : widestring
Description : File name.

Text

```
AutoBackupCapacityMonitoringFileAutoDeleted (0,
"destination", 64, 64, 64, 32, 32, 32, 64, "file name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupCapacityMonitoringFileAutoDeleted(
  const ABCapacityWarning aWarning,
  const TPlcResourceID& aDestination,
  const __int64& aTotalCapacity,
  const __int64& aFreeCapacity,
  const __int64& aAllocatedByGbf,
  const __int32 aPercentFree,
  const __int32 aPercentAllocated,
  const __int32 aPercentAllocatedByGbf,
  const __int64& aFileSize,
  const wchar_t* aFileName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupCapacityMonitoringFileAutoDeleted(
  ConstHGscAction anAction,
  ABCapacityWarning& aWarning,
  TPlcResourceID& aDestination,
  __int64& aTotalCapacity,
  __int64& aFreeCapacity,
  __int64& aAllocatedByGbf,
  __int32& aPercentFree,
  __int32& aPercentAllocated,
  __int32& aPercentAllocatedByGbf,
  __int64& aFileSize,
  const wchar_t*& aFileName);
```

Delphi – Create

```
function GscAct_CreateAutoBackupCapacityMonitoringFileAutoDeleted(
  aWarning: ABCapacityWarning;
  var aDestination: TPlcResourceID;
  var aTotalCapacity: Int64;
  var aFreeCapacity: Int64;
  var aAllocatedByGbf: Int64;
  aPercentFree: Integer;
  aPercentAllocated: Integer;
  aPercentAllocatedByGbf: Integer;
  var aFileSize: Int64;
  aFileName: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeAutoBackupCapacityMonitoringFileAutoDeleted(
  const anAction: HGscAction;
  out aWarning: ABCapacityWarning;
  out aDestination: TPlcResourceID;
  out aTotalCapacity: Int64;
  out aFreeCapacity: Int64;
  out aAllocatedByGbf: Int64;
  out aPercentFree: Integer;
  out aPercentAllocated: Integer;
  out aPercentAllocatedByGbf: Integer;
  out aFileSize: Int64;
  out aFileName: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

6.4 Auto backup capacity out of disk space

AutoBackupCapacityMonitoringOutOfDiskSpace (Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByGbf, PercentFree, PercentAllocated, PercentAllocatedByGbf)

Description : *Auto backup capacity monitoring: out of disk space.*

Code : ac_AutoBackupCapacityMonitoringOutOfDiskSpace (270)

Class : ak_Backup (7)

Parameters :

Warning (warning) :
Type : ABCapacityWarning
Description : Warning.

Destination (destination) :
Type : TResourceID
Description : Destination.

TotalCapacity (total capacity) :
Type : __int64
Description : Total capacity.

FreeCapacity (free capacity) :
Type : __int64
Description : Free capacity.

AllocatedByGbf (allocated by GBF) :
Type : __int64
Description : Allocated by GBF.

PercentFree (percent free) :
Type : __int32
Description : Percent free.

PercentAllocated (percent allocated) :
Type : __int32
Description : Percent allocated.

PercentAllocatedByGbf (percent allocated by GBF) :
Type : _int32
Description : Percent allocated by GBF.

Text

```
AutoBackupCapacityMonitoringOutOfDiskSpace (0,
"destination", 64, 64, 64, 32, 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupCapacityMonitoringOutOfDiskSpace(
  const ABCapacityWarning aWarning,
  const TPlcResourceID& aDestination,
  const __int64& aTotalCapacity,
  const __int64& aFreeCapacity,
  const __int64& aAllocatedByGbf,
  const __int32 aPercentFree,
  const __int32 aPercentAllocated,
  const __int32 aPercentAllocatedByGbf);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupCapacityMonitoringOutOfDiskSpace(
  ConstHGscAction anAction,
  ABCapacityWarning& aWarning,
  TPlcResourceID& aDestination,
  __int64& aTotalCapacity,
  __int64& aFreeCapacity,
  __int64& aAllocatedByGbf,
  __int32& aPercentFree,
  __int32& aPercentAllocated,
  __int32& aPercentAllocatedByGbf);
```

Delphi – Create

```
function GscAct_CreateAutoBackupCapacityMonitoringOutOfDiskSpace(
  aWarning: ABCapacityWarning;
  var aDestination: TPlcResourceID;
  var aTotalCapacity: Int64;
  var aFreeCapacity: Int64;
  var aAllocatedByGbf: Int64;
  aPercentFree: Integer;
  aPercentAllocated: Integer;
  aPercentAllocatedByGbf: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeAutoBackupCapacityMonitoringOutOfDiskSpace(
  const anAction: HGscAction;
  out aWarning: ABCapacityWarning;
  out aDestination: TPlcResourceID;
  out aTotalCapacity: Int64;
  out aFreeCapacity: Int64;
  out aAllocatedByGbf: Int64;
  out aPercentFree: Integer;
  out aPercentAllocated: Integer;
  out aPercentAllocatedByGbf: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

6.5 Auto backup capacity warning

AutoBackupCapacityMonitoringCapacityWarning (Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByGbf, PercentFree, PercentAllocated, PercentAllocatedByGbf)

Description : *Auto backup capacity monitoring: capacity warning.*

Code : ac_AutoBackupCapacityMonitoringCapacityWarning (272)

Class : ak_Backup (7)

Parameters :

Warning (warning) :
Type : ABCapacityWarning
Description : Warning.

Destination (destination) :
Type : TResourceID
Description : Destination.

TotalCapacity (total capacity) :
Type : __int64
Description : Total capacity.

FreeCapacity (free capacity) :
Type : __int64
Description : Free capacity.

AllocatedByGbf (allocated by GBF) :
Type : __int64
Description : Allocated by GBF.

PercentFree (percent free) :
Type : __int32
Description : Percent free.

PercentAllocated (percent allocated) :
Type : __int32
Description : Percent allocated.

PercentAllocatedByGbf (percent allocated by GBF) :
Type : _int32
Description : Percent allocated by GBF.

Text

```
AutoBackupCapacityMonitoringCapacityWarning (0,
"destination", 64, 64, 64, 32, 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupCapacityMonitoringCapacityWarning(
  const ABCapacityWarning aWarning,
  const TPlcResourceID& aDestination,
  const __int64& aTotalCapacity,
  const __int64& aFreeCapacity,
  const __int64& aAllocatedByGbf,
  const __int32 aPercentFree,
  const __int32 aPercentAllocated,
  const __int32 aPercentAllocatedByGbf);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupCapacityMonitoringCapacityWarning(
  ConstHGscAction anAction,
  ABCapacityWarning& aWarning,
  TPlcResourceID& aDestination,
  __int64& aTotalCapacity,
  __int64& aFreeCapacity,
  __int64& aAllocatedByGbf,
  __int32& aPercentFree,
  __int32& aPercentAllocated,
  __int32& aPercentAllocatedByGbf);
```

Delphi – Create

```
function GscAct_CreateAutoBackupCapacityMonitoringCapacityWarning(
  aWarning: ABCapacityWarning;
  var aDestination: TPlcResourceID;
  var aTotalCapacity: Int64;
  var aFreeCapacity: Int64;
  var aAllocatedByGbf: Int64;
  aPercentFree: Integer;
  aPercentAllocated: Integer;
  aPercentAllocatedByGbf: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeAutoBackupCapacityMonitoringCapacityWarning(
  const anAction: HGscAction;
  out aWarning: ABCapacityWarning;
  out aDestination: TPlcResourceID;
  out aTotalCapacity: Int64;
  out aFreeCapacity: Int64;
  out aAllocatedByGbf: Int64;
  out aPercentFree: Integer;
  out aPercentAllocated: Integer;
  out aPercentAllocatedByGbf: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

6.6 Auto backup file done

AutoBackupFileDone (Schedule, *StartTime*, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination, FileSizeLimit, BandWidthLimit, FileIndex, FileName, FileSize)

Description : *Auto backup progress notification: file done.*

Code : ac_AutoBackupFileDone (294)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

StartTime (start time) [optional] :
Type : TGLibDateTime
Description : Start time, empty during event backup.

EffectiveStartTime (effective start time) :
Type : TGLibDateTime
Description : Effective schedule start time.

OperationCount (operation count) :
Type : __int32
Description : Operation count.

TimerStart (timer start) :
Type : bool
Description : Timer start.

OperationIndex (operation index) :
Type : __int32
Description : Operation index.

OperationStartTime (operation start time) :
Type : TGLibDateTime
Description : Operation start time.

Source (source) :
Type : TResourceID
Description : Source.

Destination (destination) :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

FileIndex (file index) :
Type : __int32
Description : File index.

FileName (file name) :
Type : widestring
Description : File name.

FileSize (file size) :
Type : __int64
Description : File size.

Text

```
AutoBackupFileDone ("schedule", StartTime:
"2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", 32, 1, 32,
"2013/09/05 14:59:59,999 GMT+02:00", "source",
"destination", 64, 64, 32, "file name", 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupFileDone(
  const TPlcResourceID& aSchedule,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime& aEffectiveStartTime,
  const __int32 aOperationCount,
  const bool aTimerStart,
  const __int32 aOperationIndex,
  const TGLibDateTime& aOperationStartTime,
  const TPlcResourceID& aSource,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const __int32 aFileIndex,
  const wchar_t* aFileName,
  const __int64& aFileSize);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupFileDone(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule,
  const TGLibDateTime*& aStartTime,
  TGLibDateTime& aEffectiveStartTime,
  __int32& aOperationCount,
  bool& aTimerStart,
  __int32& aOperationIndex,
  TGLibDateTime& aOperationStartTime,
  TPlcResourceID& aSource,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  __int32& aFileIndex,
  const wchar_t*& aFileName,
  __int64& aFileSize);
```

Delphi – Create

```
function GscAct_CreateAutoBackupFileDone(
  var aSchedule: TPlcResourceID;
  aStartTime: PTGLibDateTime;
  var aEffectiveStartTime: TGLibDateTime;
  aOperationCount: Integer;
  aTimerStart: Boolean;
  aOperationIndex: Integer;
  var aOperationStartTime: TGLibDateTime;
  var aSource: TPlcResourceID;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  aFileIndex: Integer;
  aFileName: PWideChar;
  var aFileSize: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAutoBackupFileDone(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID;
  out aStartTime: PTGLibDateTime;
  out aEffectiveStartTime: TGLibDateTime;
  out aOperationCount: Integer;
  out aTimerStart: Boolean;
  out aOperationIndex: Integer;
  out aOperationStartTime: TGLibDateTime;
  out aSource: TPlcResourceID;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aFileIndex: Integer;
  out aFileName: PWideChar;
  out aFileSize: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

6.7 Auto backup file progress

AutoBackupFileProgress (Schedule, *StartTime*, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination, FileSizeLimit, BandWidthLimit, FileIndex, FileName, FileSize)

Description : *Auto backup progress notification: file progress.*

Code : ac_AutoBackupFileProgress (292)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

StartTime (start time) [optional] :
Type : TGLibDateTime
Description : Start time, empty during event backup.

EffectiveStartTime (effective start time) :
Type : TGLibDateTime
Description : Effective schedule start time.

OperationCount (operation count) :
Type : __int32
Description : Operation count.

TimerStart (timer start) :
Type : bool
Description : Timer start.

OperationIndex (operation index) :
Type : __int32
Description : Operation index.

OperationStartTime (operation start time) :
Type : TGLibDateTime
Description : Operation start time.

Source (source) :
Type : TResourceID
Description : Source.

Destination (destination) :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

FileIndex (file index) :
Type : __int32
Description : File index.

FileName (file name) :
Type : widestring
Description : File name.

FileSize (file size) :
Type : __int64
Description : File size.

Text

```
AutoBackupFileProgress ("schedule", StartTime:
"2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", 32, 1, 32,
"2013/09/05 14:59:59,999 GMT+02:00", "source",
"destination", 64, 64, 32, "file name", 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupFileProgress(
  const TPlcResourceID& aSchedule,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime& aEffectiveStartTime,
  const __int32 aOperationCount,
  const bool aTimerStart,
  const __int32 aOperationIndex,
  const TGLibDateTime& aOperationStartTime,
  const TPlcResourceID& aSource,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const __int32 aFileIndex,
  const wchar_t* aFileName,
  const __int64& aFileSize);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupFileProgress(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule,
  const TGLibDateTime*& aStartTime,
  TGLibDateTime& aEffectiveStartTime,
  __int32& aOperationCount,
  bool& aTimerStart,
  __int32& aOperationIndex,
  TGLibDateTime& aOperationStartTime,
  TPlcResourceID& aSource,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  __int32& aFileIndex,
  const wchar_t*& aFileName,
  __int64& aFileSize);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateAutoBackupFileProgress(
  var aSchedule: TPlcResourceID;
  aStartTime: PTGLibDateTime;
  var aEffectiveStartTime: TGLibDateTime;
  aOperationCount: Integer;
  aTimerStart: Boolean;
  aOperationIndex: Integer;
  var aOperationStartTime: TGLibDateTime;
  var aSource: TPlcResourceID;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  aFileIndex: Integer;
  aFileName: PWideChar;
  var aFileSize: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAutoBackupFileProgress(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID;
  out aStartTime: PTGLibDateTime;
  out aEffectiveStartTime: TGLibDateTime;
  out aOperationCount: Integer;
  out aTimerStart: Boolean;
  out aOperationIndex: Integer;
  out aOperationStartTime: TGLibDateTime;
  out aSource: TPlcResourceID;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aFileIndex: Integer;
  out aFileName: PWideChar;
  out aFileSize: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

6.8 Auto backup file started

AutoBackupFileStarted (Schedule, *StartTime*, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination, FileSizeLimit, BandWidthLimit, FileIndex, FileName)

Description : *Auto backup progress notification: file started.*

Code : ac_AutoBackupFileStarted (290)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

StartTime (start time) [optional] :
Type : TGLibDateTime
Description : Start time, empty during event backup.

EffectiveStartTime (effective start time) :
Type : TGLibDateTime
Description : Effective schedule start time.

OperationCount (operation count) :
Type : __int32
Description : Operation count.

TimerStart (timer start) :
Type : bool
Description : Timer start.

OperationIndex (operation index) :
Type : __int32
Description : Operation index.

OperationStartTime (operation start time) :
Type : TGLibDateTime
Description : Operation start time.

Source (source) :
Type : TResourceID
Description : Source.

Destination (destination) :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

FileIndex (file index) :
Type : __int32
Description : File index.

FileName (file name) :
Type : widestring
Description : File name.

Text

```
AutoBackupFileStarted ("schedule", StartTime:
"2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", 32, 1, 32,
"2013/09/05 14:59:59,999 GMT+02:00", "source",
"destination", 64, 64, 32, "file name")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupFileStarted(
  const TPlcResourceID& aSchedule,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime& aEffectiveStartTime,
  const __int32 aOperationCount,
  const bool aTimerStart,
  const __int32 aOperationIndex,
  const TGLibDateTime& aOperationStartTime,
  const TPlcResourceID& aSource,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const __int32 aFileIndex,
  const wchar_t* aFileName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupFileStarted(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule,
  const TGLibDateTime*& aStartTime,
  TGLibDateTime& aEffectiveStartTime,
  __int32& aOperationCount,
  bool& aTimerStart,
  __int32& aOperationIndex,
  TGLibDateTime& aOperationStartTime,
  TPlcResourceID& aSource,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  __int32& aFileIndex,
  const wchar_t*& aFileName);
```

Delphi – Create

```
function GscAct_CreateAutoBackupFileStarted(
  var aSchedule: TPlcResourceID;
  aStartTime: PTGLibDateTime;
  var aEffectiveStartTime: TGLibDateTime;
  aOperationCount: Integer;
  aTimerStart: Boolean;
  aOperationIndex: Integer;
  var aOperationStartTime: TGLibDateTime;
  var aSource: TPlcResourceID;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  aFileIndex: Integer;
  aFileName: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAutoBackupFileStarted(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID;
  out aStartTime: PTGLibDateTime;
  out aEffectiveStartTime: TGLibDateTime;
  out aOperationCount: Integer;
  out aTimerStart: Boolean;
  out aOperationIndex: Integer;
  out aOperationStartTime: TGLibDateTime;
  out aSource: TPlcResourceID;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aFileIndex: Integer;
  out aFileName: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

6.9 Auto backup operation done

AutoBackupOperationDone (Schedule, *StartTime*, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, OperationStopTime, Source, Destination, FileSizeLimit, BandWidthLimit, *OperationErrorCode*)

Description : *Auto backup progress notification: operation done.*

Code : ac_AutoBackupOperationDone (288)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

StartTime (start time) [optional] :
Type : TGLibDateTime
Description : Start time, empty during event backup.

EffectiveStartTime (effective start time) :
Type : TGLibDateTime
Description : Effective schedule start time.

OperationCount (operation count) :
Type : __int32
Description : Operation count.

TimerStart (timer start) :
Type : bool
Description : Timer start.

OperationIndex (operation index) :
Type : __int32
Description : Operation index.

OperationStartTime (operation start time) :
Type : TGLibDateTime
Description : Operation start time.

OperationStopTime (operation stop time) :
Type : TGLibDateTime
Description : Operation stop time.

Source (source) :
Type : TResourceID
Description : Source.

Destination (destination) :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

OperationErrorCode (error code) [optional] :
Type : __int32
Description : Operation error code.

Text

```
AutoBackupOperationDone ("schedule",
StartTime:  "2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", 32, 1,
32, "2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", "source",
"destination", 64, 64, OperationErrorCode:  32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupOperationDone(
  const TPlcResourceID& aSchedule,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime& aEffectiveStartTime,
  const __int32 aOperationCount,
  const bool aTimerStart,
  const __int32 aOperationIndex,
  const TGLibDateTime& aOperationStartTime,
  const TGLibDateTime& aOperationStopTime,
  const TPlcResourceID& aSource,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const __int32* aOperationErrorCode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupOperationDone(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule,
  const TGLibDateTime*& aStartTime,
  TGLibDateTime& aEffectiveStartTime,
  __int32& aOperationCount,
  bool& aTimerStart,
  __int32& aOperationIndex,
  TGLibDateTime& aOperationStartTime,
  TGLibDateTime& aOperationStopTime,
  TPlcResourceID& aSource,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  const __int32*& aOperationErrorCode);
```

Delphi – Create

```
function GscAct_CreateAutoBackupOperationDone(
  var aSchedule: TPlcResourceID;
  aStartTime: PTGLibDateTime;
  var aEffectiveStartTime: TGLibDateTime;
  aOperationCount: Integer;
  aTimerStart: Boolean;
  aOperationIndex: Integer;
  var aOperationStartTime: TGLibDateTime;
  var aOperationStopTime: TGLibDateTime;
  var aSource: TPlcResourceID;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  aOperationErrorCode: PInteger)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeAutoBackupOperationDone(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID;
  out aStartTime: PTGLibDateTime;
  out aEffectiveStartTime: TGLibDateTime;
  out aOperationCount: Integer;
  out aTimerStart: Boolean;
  out aOperationIndex: Integer;
  out aOperationStartTime: TGLibDateTime;
  out aOperationStopTime: TGLibDateTime;
  out aSource: TPlcResourceID;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aOperationErrorCode: PInteger)
  : Boolean; stdcall; external GscActionsDll;
```

6.10 Auto backup operation started

AutoBackupOperationStarted (Schedule, *StartTime*, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination, FileSizeLimit, BandWidthLimit)

Description : *Auto backup progress notification: operation started.*

Code : ac_AutoBackupOperationStarted (286)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

StartTime (start time) [optional] :
Type : TGLibDateTime
Description : Start time, empty during event backup.

EffectiveStartTime (effective start time) :
Type : TGLibDateTime
Description : Effective schedule start time.

OperationCount (operation count) :
Type : __int32
Description : Operation count.

TimerStart (timer start) :
Type : bool
Description : Timer start.

OperationIndex (operation index) :
Type : __int32
Description : Operation index.

OperationStartTime (operation start time) :
Type : TGLibDateTime
Description : Operation start time.

Source (source) :
Type : TResourceID
Description : Source.

Destination (destination) :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

Text

```
AutoBackupOperationStarted ("schedule",
StartTime:  "2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", 32, 1, 32,
"2013/09/05 14:59:59,999 GMT+02:00", "source",
"destination", 64, 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupOperationStarted(
  const TPlcResourceID& aSchedule,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime& aEffectiveStartTime,
  const __int32 aOperationCount,
  const bool aTimerStart,
  const __int32 aOperationIndex,
  const TGLibDateTime& aOperationStartTime,
  const TPlcResourceID& aSource,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupOperationStarted(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule,
  const TGLibDateTime*& aStartTime,
  TGLibDateTime& aEffectiveStartTime,
  __int32& aOperationCount,
  bool& aTimerStart,
  __int32& aOperationIndex,
  TGLibDateTime& aOperationStartTime,
  TPlcResourceID& aSource,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit);
```

Delphi – Create

```
function GscAct_CreateAutoBackupOperationStarted(
  var aSchedule: TPlcResourceID;
  aStartTime: PTGLibDateTime;
  var aEffectiveStartTime: TGLibDateTime;
  aOperationCount: Integer;
  aTimerStart: Boolean;
  aOperationIndex: Integer;
  var aOperationStartTime: TGLibDateTime;
  var aSource: TPlcResourceID;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAutoBackupOperationStarted(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID;
  out aStartTime: PTGLibDateTime;
  out aEffectiveStartTime: TGLibDateTime;
  out aOperationCount: Integer;
  out aTimerStart: Boolean;
  out aOperationIndex: Integer;
  out aOperationStartTime: TGLibDateTime;
  out aSource: TPlcResourceID;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

6.11 Auto backup schedule done

```
AutoBackupScheduleDone (Schedule, StartTime,
EffectiveStartTime, StopTime, OperationCount, TimerStart)
```

Description : *Auto backup progress notification: schedule done.*

Code : ac_AutoBackupScheduleDone (284)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

StartTime (start time) [optional] :
Type : TGLibDateTime
Description : Start time, empty during event backup.

EffectiveStartTime (effective start time) :
Type : TGLibDateTime
Description : Effective schedule start time.

StopTime (stop time) :
Type : TGLibDateTime
Description : Schedule stop time.

OperationCount (operation count) :
Type : __int32
Description : Operation count.

TimerStart (timer start) :
Type : bool
Description : Timer start.

Text

```
AutoBackupScheduleDone ("schedule", StartTime:
"2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", 32, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateAutoBackupScheduleDone(
  const TPlcResourceID& aSchedule,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime& aEffectiveStartTime,
  const TGLibDateTime& aStopTime,
  const __int32 aOperationCount,
  const bool aTimerStart);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupScheduleDone(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule,
  const TGLibDateTime*& aStartTime,
  TGLibDateTime& aEffectiveStartTime,
  TGLibDateTime& aStopTime,
  __int32& aOperationCount,
  bool& aTimerStart);
```

Delphi – Create

```
function GscAct_CreateAutoBackupScheduleDone(
  var aSchedule: TPlcResourceID;
  aStartTime: PTGLibDateTime;
  var aEffectiveStartTime: TGLibDateTime;
  var aStopTime: TGLibDateTime;
  aOperationCount: Integer;
  aTimerStart: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAutoBackupScheduleDone(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID;
  out aStartTime: PTGLibDateTime;
  out aEffectiveStartTime: TGLibDateTime;
  out aStopTime: TGLibDateTime;
  out aOperationCount: Integer;
  out aTimerStart: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

6.12 Auto backup schedule started

AutoBackupScheduleStarted (Schedule, *StartTime*, EffectiveStartTime, OperationCount, TimerStart)

Description : *Auto backup progress notification: schedule started.*

Code : ac_AutoBackupScheduleStarted (282)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

StartTime (start time) [optional] :
Type : TGLibDateTime
Description : Start time, empty during event backup.

EffectiveStartTime (effective start time) :
Type : TGLibDateTime
Description : Effective schedule start time.

OperationCount (operation count) :
Type : __int32
Description : Operation count.

TimerStart (timer start) :
Type : bool
Description : Timer start.

Text

```
AutoBackupScheduleStarted ("schedule",
StartTime:  "2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", 32, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoBackupScheduleStarted(
  const TPlcResourceID& aSchedule,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime& aEffectiveStartTime,
  const __int32 aOperationCount,
  const bool aTimerStart);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoBackupScheduleStarted(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule,
  const TGLibDateTime*& aStartTime,
  TGLibDateTime& aEffectiveStartTime,
  __int32& aOperationCount,
  bool& aTimerStart);
```

Delphi – Create

```
function GscAct_CreateAutoBackupScheduleStarted(
  var aSchedule: TPlcResourceID;
  aStartTime: PTGLibDateTime;
  var aEffectiveStartTime: TGLibDateTime;
  aOperationCount: Integer;
  aTimerStart: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeAutoBackupScheduleStarted(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID;
  out aStartTime: PTGLibDateTime;
  out aEffectiveStartTime: TGLibDateTime;
  out aOperationCount: Integer;
  out aTimerStart: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

6.13 Backup event

BackupEvent (EventID, TypeID, Destination, *StartHintID*, *StopHintID*, *Subfolder*)

Description : *Backup event.*

Code : ac_BackupEvent (281)

Class : ak_Backup (7)

Parameters :

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

Destination (destination) [ABDestination] :
Type : TResourceID
Description : Destination.

StartHintID (start hint ID) [optional] :
Type : __int64
Description : Optional start hint ID.

StopHintID (stop hint ID) [optional] :
Type : __int64
Description : Optional stop hint ID.

Subfolder (sub folder) [optional] :
Type : widestring
Description : Sub folder to backup event.

Text

```
BackupEvent (64, "event type", "destination",
StartHintID: 64, StopHintID: 64, Subfolder:  "sub
folder")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateBackupEvent(
  const __int64& aEventID,
  const TPlcEventTypeID& aTypeID,
  const TPlcResourceID& aDestination,
  const __int64* aStartHintID,
  const __int64* aStopHintID,
  const wchar_t* aSubfolder);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeBackupEvent(
  ConstHGscAction anAction,
  __int64& aEventID,
  TPlcEventTypeID& aTypeID,
  TPlcResourceID& aDestination,
  const __int64*& aStartHintID,
  const __int64*& aStopHintID,
  const wchar_t*& aSubfolder);
```

Delphi – Create

```
function GscAct_CreateBackupEvent(
  var aEventID: Int64;
  var aTypeID: TPlcEventTypeID;
  var aDestination: TPlcResourceID;
  aStartHintID: PInt64;
  aStopHintID: PInt64;
  aSubfolder: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeBackupEvent(
  const anAction: HGscAction;
  out aEventID: Int64;
  out aTypeID: TPlcEventTypeID;
  out aDestination: TPlcResourceID;
  out aStartHintID: PInt64;
  out aStopHintID: PInt64;
  out aSubfolder: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

6.14 Event backup done

EventBackupDone (JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, StopTime, *OperationErrorCode*)

GEUTEBRÜCK

Description : *Event backup progress notification: backup done.*

Code : `ac_EventBackupDone` (312)

Class : `ak_Backup` (7)

Parameters :

JobID (job ID) :
Type : GUID
Description : Backup job ID.

EventTypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

Destination (destination) [ABDestination] :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

StartTime (start time) :
Type : TGLibDateTime
Description : Backup start time.

StopTime (stop time) :
Type : TGLibDateTime
Description : Backup stop time.

OperationErrorCode (error code) [optional] :
Type : __int32
Description : Operation error code.

Text

```
EventBackupDone ("00000000-0000-0000-0000-000000000000",
"event type", 64, "destination", 64, 64,
"2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", OperationErrorCode:
32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventBackupDone(
  const GUID& aJobID,
  const TPlcEventTypeID& aEventTypeID,
  const __int64& aEventID,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const TGLibDateTime& aStartTime,
  const TGLibDateTime& aStopTime,
  const __int32* aOperationErrorCode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventBackupDone(
  ConstHGscAction anAction,
  GUID& aJobID,
  TPlcEventTypeID& aEventTypeID,
  __int64& aEventID,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  TGLibDateTime& aStartTime,
  TGLibDateTime& aStopTime,
  const __int32*& aOperationErrorCode);
```

Delphi – Create

```
function GscAct_CreateEventBackupDone(
  var aJobID: TGuid;
  var aEventTypeID: TPlcEventTypeID;
  var aEventID: Int64;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  var aStartTime: TGLibDateTime;
  var aStopTime: TGLibDateTime;
  aOperationErrorCode: PInteger)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeEventBackupDone(
  const anAction: HGscAction;
  out aJobID: TGuid;
  out aEventTypeID: TPlcEventTypeID;
  out aEventID: Int64;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aStartTime: TGLibDateTime;
  out aStopTime: TGLibDateTime;
  out aOperationErrorCode: PInteger)
  : Boolean; stdcall; external GscActionsDll;
```

6.15 Event backup file done

EventBackupFileDone (JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, FileIndex, FileName, FileSize)

Description : *Event backup progress notification: file done.*

Code : ac_EventBackupFileDone (318)

Class : ak_Backup (7)

Parameters :

JobID (job ID) :
Type : GUID
Description : Backup job ID.

EventTypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

Destination (destination) [ABDestination] :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

StartTime (start time) :
Type : TGLibDateTime
Description : Effective backup start time.

FileIndex (file index) :
Type : __int32
Description : File index.

FileName (file name) :
Type : widestring
Description : File name.

FileSize (file size) :
Type : __int64
Description : File size.

Text

```
EventBackupFileDone ("00000000-0000-0000-0000-000000000000",
"event type", 64, "destination", 64, 64,
"2013/09/05 14:59:59,999 GMT+02:00", 32, "file name",
64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventBackupFileDone(
  const GUID& aJobID,
  const TPlcEventTypeID& aEventTypeID,
  const __int64& aEventID,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const TGLibDateTime& aStartTime,
  const __int32 aFileIndex,
  const wchar_t* aFileName,
  const __int64& aFileSize);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventBackupFileDone(
  ConstHGscAction anAction,
  GUID& aJobID,
  TPlcEventTypeID& aEventTypeID,
  __int64& aEventID,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  TGLibDateTime& aStartTime,
  __int32& aFileIndex,
  const wchar_t*& aFileName,
  __int64& aFileSize);
```

Delphi – Create

```
function GscAct_CreateEventBackupFileDone(
  var aJobID: TGuid;
  var aEventTypeID: TPlcEventTypeID;
  var aEventID: Int64;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  var aStartTime: TGLibDateTime;
  aFileIndex: Integer;
  aFileName: PWideChar;
  var aFileSize: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeEventBackupFileDone(
  const anAction: HGscAction;
  out aJobID: TGuid;
  out aEventTypeID: TPlcEventTypeID;
  out aEventID: Int64;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aStartTime: TGLibDateTime;
  out aFileIndex: Integer;
  out aFileName: PWideChar;
  out aFileSize: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

6.16 Event backup file progress

EventBackupFileProgress (JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, FileIndex, FileName, FileSize)

Description : *Event backup progress notification: file progress.*

Code : ac_EventBackupFileProgress (316)

Class : ak_Backup (7)

Parameters :

JobID (job ID) :
Type : GUID
Description : Backup job ID.

EventTypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

Destination (destination) [ABDestination] :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

StartTime (start time) :
Type : TGLibDateTime
Description : Effective backup start time.

FileIndex (file index) :
Type : __int32
Description : File index.

FileName (file name) :
Type : widestring
Description : File name.

FileSize (file size) :
Type : __int64
Description : File size.

Text

```
EventBackupFileProgress ("00000000-0000-0000-0000-000000000000",
"event type", 64, "destination", 64, 64,
"2013/09/05 14:59:59,999 GMT+02:00", 32, "file name",
64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventBackupFileProgress(
  const GUID& aJobID,
  const TPlcEventTypeID& aEventTypeID,
  const __int64& aEventID,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const TGLibDateTime& aStartTime,
  const __int32 aFileIndex,
  const wchar_t* aFileName,
  const __int64& aFileSize);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventBackupFileProgress(
  ConstHGscAction anAction,
  GUID& aJobID,
  TPlcEventTypeID& aEventTypeID,
  __int64& aEventID,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  TGLibDateTime& aStartTime,
  __int32& aFileIndex,
  const wchar_t*& aFileName,
  __int64& aFileSize);
```

Delphi – Create

```
function GscAct_CreateEventBackupFileProgress(
  var aJobID: TGuid;
  var aEventTypeID: TPlcEventTypeID;
  var aEventID: Int64;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  var aStartTime: TGLibDateTime;
  aFileIndex: Integer;
  aFileName: PWideChar;
  var aFileSize: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeEventBackupFileProgress(
  const anAction: HGscAction;
  out aJobID: TGuid;
  out aEventTypeID: TPlcEventTypeID;
  out aEventID: Int64;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aStartTime: TGLibDateTime;
  out aFileIndex: Integer;
  out aFileName: PWideChar;
  out aFileSize: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

6.17 Event backup file started

EventBackupFileStarted (JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, FileIndex, FileName)

Description : *Event backup progress notification: file started.*

Code : `ac_EventBackupFileStarted` (314)

Class : `ak_Backup` (7)

Parameters :

JobID (job ID) :
Type : GUID
Description : Backup job ID.

EventTypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

Destination (destination) [ABDestination] :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

StartTime (start time) :
Type : TGLibDateTime
Description : Effective backup start time.

FileIndex (file index) :
Type : __int32
Description : File index.

FileName (file name) :
Type : widestring
Description : File name.

Text

```
EventBackupFileStarted ("00000000-0000-0000-0000-000000000000",
"event type", 64, "destination", 64, 64,
"2013/09/05 14:59:59,999 GMT+02:00", 32, "file name")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventBackupFileStarted(
  const GUID& aJobID,
  const TPlcEventTypeID& aEventTypeID,
  const __int64& aEventID,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const TGLibDateTime& aStartTime,
  const __int32 aFileIndex,
  const wchar_t* aFileName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventBackupFileStarted(
  ConstHGscAction anAction,
  GUID& aJobID,
  TPlcEventTypeID& aEventTypeID,
  __int64& aEventID,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  TGLibDateTime& aStartTime,
  __int32& aFileIndex,
  const wchar_t*& aFileName);
```

Delphi – Create

```
function GscAct_CreateEventBackupFileStarted(
  var aJobID: TGuid;
  var aEventTypeID: TPlcEventTypeID;
  var aEventID: Int64;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  var aStartTime: TGLibDateTime;
  aFileIndex: Integer;
  aFileName: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeEventBackupFileStarted(
  const anAction: HGscAction;
  out aJobID: TGuid;
  out aEventTypeID: TPlcEventTypeID;
  out aEventID: Int64;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aStartTime: TGLibDateTime;
  out aFileIndex: Integer;
  out aFileName: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

6.18 Event backup started

EventBackupStarted (JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime)

Description : *Event backup progress notification: backup started.*

Code : ac_EventBackupStarted (310)

Class : ak_Backup (7)

Parameters :

JobID (job ID) :
Type : GUID
Description : Backup job ID.

EventTypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

Destination (destination) [ABDestination] :
Type : TResourceID
Description : Destination.

FileSizeLimit (file size limit) :
Type : __int64
Description : File size limit.

BandWidthLimit (band width limit) :
Type : __int64
Description : Band width limit.

StartTime (start time) :
Type : TGLibDateTime
Description : Backup start time.

GEUTEBRÜCK

Text

```
EventBackupStarted ("00000000-0000-0000-0000-000000000000",
"event type", 64, "destination", 64, 64,
"2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventBackupStarted(
  const GUID& aJobID,
  const TPlcEventTypeID& aEventTypeID,
  const __int64& aEventID,
  const TPlcResourceID& aDestination,
  const __int64& aFileSizeLimit,
  const __int64& aBandWidthLimit,
  const TGLibDateTime& aStartTime);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventBackupStarted(
  ConstHGscAction anAction,
  GUID& aJobID,
  TPlcEventTypeID& aEventTypeID,
  __int64& aEventID,
  TPlcResourceID& aDestination,
  __int64& aFileSizeLimit,
  __int64& aBandWidthLimit,
  TGLibDateTime& aStartTime);
```

Delphi – Create

```
function GscAct_CreateEventBackupStarted(
  var aJobID: TGuid;
  var aEventTypeID: TPlcEventTypeID;
  var aEventID: Int64;
  var aDestination: TPlcResourceID;
  var aFileSizeLimit: Int64;
  var aBandWidthLimit: Int64;
  var aStartTime: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeEventBackupStarted(
  const anAction: HGscAction;
  out aJobID: TGuid;
  out aEventTypeID: TPlcEventTypeID;
  out aEventID: Int64;
  out aDestination: TPlcResourceID;
  out aFileSizeLimit: Int64;
  out aBandWidthLimit: Int64;
  out aStartTime: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

6.19 Start auto backup

StartAutoBackup (Schedule)

Description : *Start auto backup.*

Code : ac_StartAutoBackup (280)

Class : ak_Backup (7)

Parameters :

Schedule (schedule) [ABSchedule] :
Type : TResourceID
Description : Schedule.

Text

```
StartAutoBackup ("schedule")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateStartAutoBackup(
  const TPlcResourceID& aSchedule);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeStartAutoBackup(
  ConstHGscAction anAction,
  TPlcResourceID& aSchedule);
```

Delphi – Create

```
function GscAct_CreateStartAutoBackup(
  var aSchedule: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeStartAutoBackup(
  const anAction: HGscAction;
  out aSchedule: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

7 Camera control

All actions to remote control the PTZ heads or camera adjustments.

7.1 Auto focus off

AutoFocusOff (Camera)

Description : *Disable auto focus of the camera.*

Code : ac_AutoFocusOff (152)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
AutoFocusOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoFocusOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoFocusOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateAutoFocusOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAutoFocusOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

7.2 Auto focus on

AutoFocusOn (Camera)

Description : *Enable auto focus of the camera.*

Code : ac_AutoFocusOn (151)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
AutoFocusOn (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateAutoFocusOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeAutoFocusOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateAutoFocusOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeAutoFocusOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.3 Camera RAW output

CameraRAWOutput (Camera, Output)

Description : *Output unformatted command to the camera.*

Code : `ac_CameraRAWOutput` (147)

Class : `ak_CameraControl` (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Output (output) :
Type : string
Description : RAW command to output.

Text

```
CameraRAWOutput (32, "output")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraRAWOutput(
  const TPlcMediaChannelID& aCamera,
  const char* aOutput);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraRAWOutput(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  const char*& aOutput);
```

Delphi – Create

```
function GscAct_CreateCameraRAWOutput(
  var aCamera: TPlcMediaChannelID;
  aOutput: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraRAWOutput(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aOutput: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

7.4 Camera apply profile

```
CameraApplyProfile (Camera, Profile)
```

Description : *Apply a predefined profile settings to the camera.*

Code : ac_CameraApplyProfile (159)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Profile (profile) :
Type : widestring
Description : Profile name.

Text

```
CameraApplyProfile (32, "profile")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraApplyProfile(
  const TPlcMediaChannelID& aCamera,
  const wchar_t* aProfile);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraApplyProfile(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  const wchar_t*& aProfile);
```

Delphi – Create

```
function GscAct_CreateCameraApplyProfile(
  var aCamera: TPlcMediaChannelID;
  aProfile: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraApplyProfile(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aProfile: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

7.5 Camera backlight compensation mode

CameraBacklightCompensationMode (Camera, Mode)

Description : *Change backlight compensation mode of the camera.*

Code : ac_CameraBacklightCompensationMode (158)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Mode (mode) :
Type : PlcBacklightMode
Description : Backlight compensation mode.

Text

```
CameraBacklightCompensationMode (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraBacklightCompensationMode(
  const TPlcMediaChannelID& aCamera,
  const PlcBacklightMode aMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraBacklightCompensationMode(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  PlcBacklightMode& aMode);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateCameraBacklightCompensationMode(
  var aCamera: TPlcMediaChannelID;
  aMode: PlcBacklightMode)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraBacklightCompensationMode(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aMode: PlcBacklightMode)
  : Boolean; stdcall; external GscActionsDll;
```

7.6 Camera clear preset text

CameraClearPrePosText (Camera, Position)

Description : *Clear camera text for preset position.*

Code : ac_CameraClearPrePosText (145)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Position (position) :
Type : __int32
Description : Preset position.

Text

```
CameraClearPrePosText (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraClearPrePosText(
  const TPlcMediaChannelID& aCamera,
  const __int32 aPosition);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraClearPrePosText(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aPosition);
```

Delphi – Create

```
function GscAct_CreateCameraClearPrePosText(
  var aCamera: TPlcMediaChannelID;
  aPosition: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraClearPrePosText(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aPosition: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.7 Camera day/night mode

CameraDayNightMode (Camera, Mode)

Description : *Change day/night mode of the camera.*

Code : ac_CameraDayNightMode (157)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Mode (mode) :
Type : PlcDayNightMode
Description : Day/night mode.

Text

```
CameraDayNightMode (32, 0)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraDayNightMode(
  const TPlcMediaChannelID& aCamera,
  const PlcDayNightMode aMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraDayNightMode(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  PlcDayNightMode& aMode);
```

Delphi – Create

```
function GscAct_CreateCameraDayNightMode(
  var aCamera: TPlcMediaChannelID;
  aMode: PlcDayNightMode)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraDayNightMode(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aMode: PlcDayNightMode)
  : Boolean; stdcall; external GscActionsDll;
```

7.8 Camera light off

CameraLightOff (Camera)

Description : *Turn camera light off.*

Code : ac_CameraLightOff (121)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraLightOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraLightOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraLightOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraLightOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraLightOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.9 Camera light on

CameraLightOn (Camera)

Description : *Turn camera light on.*

Code : ac_CameraLightOn (120)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraLightOn (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraLightOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraLightOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraLightOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraLightOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.10 Camera manual iris off

CameraManualIrisOff (Camera)

Description : *Disable manual iris of the camera.*

Code : ac_CameraManualIrisOff (154)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraManualIrisOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraManualIrisOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraManualIrisOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraManualIrisOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraManualIrisOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.11 Camera manual iris on

CameraManualIrisOn (Camera)

Description : *Enable manual iris of the camera.*

Code : ac_CameraManualIrisOn (153)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraManualIrisOn (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraManualIrisOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraManualIrisOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraManualIrisOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraManualIrisOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.12 Camera off

CameraOff (Camera)

Description : *Turn camera off.*

Code : ac_CameraOff (123)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.13 Camera on

CameraOn (Camera)

Description : *Turn camera on.*

Code : ac_CameraOn (122)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraOn (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.14 Camera pump off

CameraPumpOff (Camera)

Description : *Turn camera pump off.*

Code : ac_CameraPumpOff (125)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraPumpOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraPumpOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraPumpOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraPumpOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraPumpOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.15 Camera pump on

CameraPumpOn (Camera)

Description : *Turn camera pump on.*

Code : ac_CameraPumpOn (124)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraPumpOn (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraPumpOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraPumpOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraPumpOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraPumpOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.16 Camera select char mode

CameraSelectCharMode (Camera, Mode)

Description : *Select camera character mode.*

Code : ac_CameraSelectCharMode (144)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Mode (mode) :
Type : __int32
Description : Character mode.

Text

```
CameraSelectCharMode (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSelectCharMode(
  const TPlcMediaChannelID& aCamera,
  const __int32 aMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSelectCharMode(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aMode);
```

Delphi – Create

```
function GscAct_CreateCameraSelectCharMode(
  var aCamera: TPlcMediaChannelID;
  aMode: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSelectCharMode(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aMode: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.17 Camera set preset text

CameraSetPrePosText (Camera, Position)

Description : *Set camera text for preset position.*

Code : ac_CameraSetPrePosText (146)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Position (position) :
Type : __int32
Description : Preset position.

GEUTEBRÜCK

Text

```
CameraSetPrePosText (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSetPrePosText(
  const TPlcMediaChannelID& aCamera,
  const __int32 aPosition);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSetPrePosText(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aPosition);
```

Delphi – Create

```
function GscAct_CreateCameraSetPrePosText(
  var aCamera: TPlcMediaChannelID;
  aPosition: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSetPrePosText(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aPosition: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.18 Camera spec func U off

CameraSpecFuncUOff (Camera)

Description : *Turn camera special function U off.*

Code : ac_CameraSpecFuncUOff (131)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraSpecFuncUOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncUOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncUOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncUOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncUOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.19 Camera spec func U on

CameraSpecFuncUOn (Camera)

Description : *Turn camera special function U on.*

Code : ac_CameraSpecFuncUOn (130)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraSpecFuncUOn (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncUOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncUOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncUOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncUOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.20 Camera spec func V off

CameraSpecFuncVOff (Camera)

Description : *Turn camera special function V off.*

Code : ac_CameraSpecFuncVOff (133)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

CameraSpecFuncVOff (32)

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncVOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncVOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncVOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncVOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.21 Camera spec func V on

CameraSpecFuncVOn (Camera)

Description : *Turn camera special function V on.*

Code : ac_CameraSpecFuncVOn (132)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraSpecFuncVOn (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncVOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncVOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncVOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncVOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.22 Camera spec func X off

CameraSpecFuncXOff (Camera)

Description : *Turn camera special function X off.*

Code : ac_CameraSpecFuncXOff (135)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraSpecFuncXOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncXOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncXOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncXOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncXOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.23 Camera spec func X on

CameraSpecFuncXOn (Camera)

Description : *Turn camera special function X on.*

Code : ac_CameraSpecFuncXOn (134)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

CameraSpecFuncXOn (32)

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncXOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncXOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncXOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncXOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.24 Camera spec func Y off

CameraSpecFuncYOff (Camera)

Description : *Turn camera special function Y off.*

Code : ac_CameraSpecFuncYOff (137)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraSpecFuncYOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncYOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncYOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncYOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncYOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.25 Camera spec func Y on

CameraSpecFuncYOn (Camera)

Description : *Turn camera special function Y on.*

Code : ac_CameraSpecFuncYOn (136)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraSpecFuncYOn (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraSpecFuncYOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraSpecFuncYOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraSpecFuncYOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraSpecFuncYOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.26 Camera stop all

CameraStopAll (Camera)

Description : *Camera stop all.*

Code : ac_CameraStopAll (138)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraStopAll (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraStopAll(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraStopAll(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraStopAll(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraStopAll(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.27 Camera text off

CameraTextOff (Camera)

Description : *Disable camera text.*

Code : ac_CameraTextOff (156)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraTextOff (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraTextOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraTextOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraTextOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraTextOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.28 Camera text on

CameraTextOn (Camera)

Description : *Enable camera text.*

Code : ac_CameraTextOn (155)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

CameraTextOn (32)

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraTextOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraTextOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraTextOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraTextOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.29 Camera tour start

CameraTourStart (Camera, TourID, TourName)

Description : *Start camera tour.*

Code : ac_CameraTourStart (118)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

TourID (tour ID) :
Type : __int32
Description : Tour ID.

TourName (tour name) :
Type : string
Description : Tour name.

GEUTEBRÜCK

Text

```
CameraTourStart (32, 32, "tour name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraTourStart(
  const TPlcMediaChannelID& aCamera,
  const __int32 aTourID,
  const char* aTourName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraTourStart(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aTourID,
  const char*& aTourName);
```

Delphi – Create

```
function GscAct_CreateCameraTourStart(
  var aCamera: TPlcMediaChannelID;
  aTourID: Integer;
  aTourName: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraTourStart(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aTourID: Integer;
  out aTourName: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

7.30 Camera tour stop

CameraTourStop (Camera)

Description : *Stop camera tour.*

Code : ac_CameraTourStop (119)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraTourStop (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateCameraTourStop(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraTourStop(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraTourStop(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraTourStop(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.31 Camera version off

CameraVersionOff (Camera)

Description : *Hide camera version string.*

Code : ac_CameraVersionOff (149)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

GEUTEBRÜCK

Text

```
CameraVersionOff (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraVersionOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraVersionOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraVersionOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraVersionOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.32 Camera version on

CameraVersionOn (Camera)

Description : *Show camera version string.*

Code : ac_CameraVersionOn (148)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraVersionOn (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraVersionOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraVersionOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraVersionOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraVersionOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.33 Camera wash-wipe off

CameraWashOff (Camera)

Description : *Turn camera wash-wipe off.*

Code : ac_CameraWashOff (129)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraWashOff (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraWashOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraWashOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraWashOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraWashOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.34 Camera wash-wipe on

CameraWashOn (Camera)

Description : *Turn camera wash-wipe on.*

Code : ac_CameraWashOn (128)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
CameraWashOn (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCameraWashOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCameraWashOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateCameraWashOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCameraWashOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.35 Clear default position

DefaultPosClear (Camera)

Description : *Clear camera default position.*

Code : ac_DefaultPosClear (142)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
DefaultPosClear (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDefaultPosClear(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDefaultPosClear(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateDefaultPosClear(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDefaultPosClear(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.36 Clear preset position

PrePosClear (Camera, Position)

Description : *Clear camera preset position.*

Code : ac_PrePosClear (141)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Position (position) :
Type : __int32
Description : Preset position.

Text

```
PrePosClear (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePrePosClear(
  const TPlcMediaChannelID& aCamera,
  const __int32 aPosition);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePrePosClear(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aPosition);
```

Delphi – Create

```
function GscAct_CreatePrePosClear(
  var aCamera: TPlcMediaChannelID;
  aPosition: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePrePosClear(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aPosition: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.37 Fast speed off

FastSpeedOff (Camera)

Description : *Turn camera fast speed off.*

Code : ac_FastSpeedOff (127)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
FastSpeedOff (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateFastSpeedOff(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeFastSpeedOff(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateFastSpeedOff(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeFastSpeedOff(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.38 Fast speed on

FastSpeedOn (Camera)

Description : *Turn camera fast speed on.*

Code : ac_FastSpeedOn (126)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
FastSpeedOn (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateFastSpeedOn(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeFastSpeedOn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateFastSpeedOn(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeFastSpeedOn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.39 Focus far

FocusFar (Camera, Speed)

Description : *Focus camera far.*

Code : ac_FocusFar (111)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Focus speed.

Text

```
FocusFar (32, 32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateFocusFar(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeFocusFar(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

Delphi – Create

```
function GscAct_CreateFocusFar(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeFocusFar(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.40 Focus near

FocusNear (Camera, Speed)

Description : *Focus camera near.*

Code : ac_FocusNear (110)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Focus speed.

Text

```
FocusNear (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateFocusNear(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeFocusNear(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

Delphi – Create

```
function GscAct_CreateFocusNear(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeFocusNear(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.41 Focus stop

FocusStop (Camera)

Description : *Stop camera focus.*

Code : ac_FocusStop (112)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

GEUTEBRÜCK

Text

```
FocusStop (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateFocusStop(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeFocusStop(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateFocusStop(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeFocusStop(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.42 Iris close

IrisClose (Camera)

Description : *Close camera iris.*

Code : ac_IrisClose (114)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
IrisClose (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIrisClose(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIrisClose(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateIrisClose(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIrisClose(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.43 Iris open

IrisOpen (Camera)

Description : *Open camera iris.*

Code : ac_IrisOpen (113)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
IrisOpen (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIrisOpen(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIrisOpen(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateIrisOpen(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIrisOpen(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.44 Iris stop

IrisStop (Camera)

Description : *Stop camera iris.*

Code : ac_IrisStop (115)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
IrisStop (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIrisStop(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIrisStop(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateIrisStop(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIrisStop(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.45 Move by speed

MoveToBySpeed (Camera, SpeedAlpha, SpeedBeta)

Description : *Move camera by speed.*

Code : ac_MoveToBySpeed (162)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

SpeedAlpha (alpha speed) :
Type : double
Description : Speed in alpha (pan) direction (0/100 %).

SpeedBeta (beta speed) :
Type : double
Description : Speed in beta (tilt) direction (0/100 %).

GEUTEBRÜCK

Text

```
MoveToBySpeed (32, 0.0, 0.0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateMoveToBySpeed(
  const TPlcMediaChannelID& aCamera,
  const double& aSpeedAlpha,
  const double& aSpeedBeta);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeMoveToBySpeed(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  double& aSpeedAlpha,
  double& aSpeedBeta);
```

Delphi – Create

```
function GscAct_CreateMoveToBySpeed(
  var aCamera: TPlcMediaChannelID;
  var aSpeedAlpha: Double;
  var aSpeedBeta: Double)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeMoveToBySpeed(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeedAlpha: Double;
  out aSpeedBeta: Double)
  : Boolean; stdcall; external GscActionsDll;
```

7.46 Move to absolute position

MoveToAbsolutePosition (Camera, Alpha, Beta, SpeedAlpha, SpeedBeta)

Description : *Move camera to absolute position.*

Code : ac_MoveToAbsolutePosition (160)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Alpha (alpha) :
Type : double
Description : Alpha angle (-180/+180).

Beta (beta) :
Type : double
Description : Beta angle (-90/+90).

SpeedAlpha (alpha speed) :
Type : double
Description : Speed in alpha direction (0/100 %).

SpeedBeta (beta speed) :
Type : double
Description : Speed in beta direction (0/100 %).

Text

```
MoveToAbsolutePosition (32, 0.0, 0.0, 0.0, 0.0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateMoveToAbsolutePosition(
  const TPlcMediaChannelID& aCamera,
  const double& aAlpha,
  const double& aBeta,
  const double& aSpeedAlpha,
  const double& aSpeedBeta);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeMoveToAbsolutePosition(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  double& aAlpha,
  double& aBeta,
  double& aSpeedAlpha,
  double& aSpeedBeta);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateMoveToAbsolutePosition(
  var aCamera: TPlcMediaChannelID;
  var aAlpha: Double;
  var aBeta: Double;
  var aSpeedAlpha: Double;
  var aSpeedBeta: Double)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeMoveToAbsolutePosition(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aAlpha: Double;
  out aBeta: Double;
  out aSpeedAlpha: Double;
  out aSpeedBeta: Double)
  : Boolean; stdcall; external GscActionsDll;
```

7.47 Move to default position

DefaultPosCallUp (Camera)

Description : *Move camera to default position.*

Code : ac_DefaultPosCallUp (117)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
DefaultPosCallUp (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDefaultPosCallUp(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDefaultPosCallUp(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateDefaultPosCallUp(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDefaultPosCallUp(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.48 Move to preset position

PrePosCallUp (Camera, Position)

Description : *Move camera to preset position.*

Code : ac_PrePosCallUp (116)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Position (position) :
Type : __int32
Description : Preset position.

Text

```
PrePosCallUp (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePrePosCallUp(
  const TPlcMediaChannelID& aCamera,
  const __int32 aPosition);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePrePosCallUp(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aPosition);
```

Delphi – Create

```
function GscAct_CreatePrePosCallUp(
  var aCamera: TPlcMediaChannelID;
  aPosition: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePrePosCallUp(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aPosition: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.49 Move to relative position

MoveToRelativePosition (Camera, Alpha, Beta, SpeedAlpha, SpeedBeta)

Description : *Move camera to relative position.*

Code : ac_MoveToRelativePosition (161)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Alpha (alpha) :
Type : double
Description : Alpha angle (-180/+180).

Beta (beta) :
Type : double
Description : Beta angle (-90/+90).

SpeedAlpha (alpha speed) :
Type : double
Description : Speed in alpha direction (0/100 %).

SpeedBeta (beta speed) :
Type : double
Description : Speed in beta direction (0/100 %).

Text

```
MoveToRelativePosition (32, 0.0, 0.0, 0.0, 0.0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateMoveToRelativePosition(
  const TPlcMediaChannelID& aCamera,
  const double& aAlpha,
  const double& aBeta,
  const double& aSpeedAlpha,
  const double& aSpeedBeta);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeMoveToRelativePosition(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  double& aAlpha,
  double& aBeta,
  double& aSpeedAlpha,
  double& aSpeedBeta);
```

Delphi – Create

```
function GscAct_CreateMoveToRelativePosition(
  var aCamera: TPlcMediaChannelID;
  var aAlpha: Double;
  var aBeta: Double;
  var aSpeedAlpha: Double;
  var aSpeedBeta: Double)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeMoveToRelativePosition(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aAlpha: Double;
  out aBeta: Double;
  out aSpeedAlpha: Double;
  out aSpeedBeta: Double)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

7.50 Pan auto

```
PanAuto (Camera, Modus)
```

Description : *Turn auto pan of the camera.*

Code : ac_PanAuto (150)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Modus (modus) :
Type : __int32
Description : Pan modus.

Text

```
PanAuto (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePanAuto(
  const TPlcMediaChannelID& aCamera,
  const __int32 aModus);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePanAuto(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aModus);
```

Delphi – Create

```
function GscAct_CreatePanAuto(
  var aCamera: TPlcMediaChannelID;
  aModus: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePanAuto(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aModus: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.51 Pan left

PanLeft (Camera, Speed)

Description : *Pan camera to the left.*

Code : ac_PanLeft (102)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Pan speed.

Text

```
PanLeft (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePanLeft(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePanLeft(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreatePanLeft(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePanLeft(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.52 Pan right

PanRight (Camera, Speed)

Description : *Pan camera to the right.*

Code : ac_PanRight (101)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Pan speed.

Text

```
PanRight (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePanRight(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePanRight(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

Delphi – Create

```
function GscAct_CreatePanRight(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePanRight(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.53 Pan stop

PanStop (Camera)

Description : *Stop pan movement.*

Code : ac_PanStop (103)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
PanStop (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePanStop(
  const TPlcMediaChannelID& aCamera);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePanStop(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreatePanStop(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePanStop(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.54 Save default position

DefaultPosSave (Camera)

Description : *Save current camera position as default position.*

Code : ac_DefaultPosSave (140)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
DefaultPosSave (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDefaultPosSave(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDefaultPosSave(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateDefaultPosSave(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDefaultPosSave(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.55 Save preset position

PrePosSave (Camera, Position)

Description : *Save current camera position as preset position.*

Code : ac_PrePosSave (139)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Position (position) :
Type : __int32
Description : Preset position.

Text

```
PrePosSave (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePrePosSave(
  const TPlcMediaChannelID& aCamera,
  const __int32 aPosition);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePrePosSave(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aPosition);
```

Delphi – Create

```
function GscAct_CreatePrePosSave(
  var aCamera: TPlcMediaChannelID;
  aPosition: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePrePosSave(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aPosition: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.56 Set camera text

SetCameraText (Camera, Text)

Description : *Set camera text.*

Code : ac_SetCameraText (143)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text (text) :
Type : string
Description : Text to superimpose.

Text

```
SetCameraText (32, "text")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetCameraText(
  const TPlcMediaChannelID& aCamera,
  const char* aText);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetCameraText(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  const char*& aText);
```

Delphi – Create

```
function GscAct_CreateSetCameraText(
  var aCamera: TPlcMediaChannelID;
  aText: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetCameraText(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aText: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

7.57 Tilt down

TiltDown (Camera, Speed)

Description : *Tilt camera down.*

Code : ac_TiltDown (105)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Tilt speed.

GEUTEBRÜCK

Text

```
TiltDown (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTiltDown(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTiltDown(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

Delphi – Create

```
function GscAct_CreateTiltDown(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeTiltDown(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.58 Tilt stop

TiltStop (Camera)

Description : *Stop tilt movement.*

Code : ac_TiltStop (106)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
TiltStop (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTiltStop(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTiltStop(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateTiltStop(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeTiltStop(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

7.59 Tilt up

TiltUp (Camera, Speed)

Description : *Tilt camera up.*

Code : ac_TiltUp (104)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Tilt speed.

GEUTEBRÜCK

Text

```
TiltUp (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTiltUp(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTiltUp(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

Delphi – Create

```
function GscAct_CreateTiltUp(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeTiltUp(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.60 Zoom in

```
ZoomIn (Camera, Speed)
```

Description : *Zoom camera in.*

Code : `ac_ZoomIn` (107)

Class : `ak_CameraControl` (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Zoom speed.

Text

```
ZoomIn (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateZoomIn(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeZoomIn(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

Delphi – Create

```
function GscAct_CreateZoomIn(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeZoomIn(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.61 Zoom out

```
ZoomOut (Camera, Speed)
```

Description : *Zoom camera out.*

Code : ac_ZoomOut (108)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Speed (speed) :
Type : __int32
Description : Zoom speed.

Text

```
ZoomOut (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateZoomOut(
  const TPlcMediaChannelID& aCamera,
  const __int32 aSpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeZoomOut(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera,
  __int32& aSpeed);
```

Delphi – Create

```
function GscAct_CreateZoomOut(
  var aCamera: TPlcMediaChannelID;
  aSpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeZoomOut(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID;
  out aSpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

7.62 Zoom stop

```
ZoomStop (Camera)
```

Description : *Stop camera zoom.*

Code : ac_ZoomStop (109)

Class : ak_CameraControl (3)

Parameters :

Camera (PTZ head) [PTZHead] :
Type : TMediaChannelID
Description : PTZ head.

Text

```
ZoomStop (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateZoomStop(
  const TPlcMediaChannelID& aCamera);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeZoomStop(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aCamera);
```

Delphi – Create

```
function GscAct_CreateZoomStop(
  var aCamera: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeZoomStop(
  const anAction: HGscAction;
  out aCamera: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

8 Cash management actions

All actions for cash management.

8.1 Safebag close

SafebagClose (WorkingPlace, StartTime, StopTime, SafebagNo, SafebagInfo, StepID, *Debit*, *Total*, *Difference*, *HasDifference*, *Notes*, *Coins*, *Cheques*)

Description : *Safebag close.*

Code : ac_SafebagClose (405)

Class : ak_CashManagement (8)

Parameters :

WorkingPlace (working place) :
Type : __int32
Description : Working place no.

StartTime (start time) :
Type : TGLibDateTime
Description : Start time stamp.

StopTime (stop time) :
Type : TGLibDateTime
Description : Stop time stamp.

SafebagNo (safebag no.) :
Type : widestring
Description : Safebag no.

SafebagInfo (safebag info) :
Type : widestring
Description : Additional info about safebag.

StepID (step ID) :
Type : SafebagStep
Description : Processing step ID.

Debit (debit) [optional] :
Type : double
Description : Debit.

Total (total) [optional] :
Type : double
Description : Total.

Difference (difference) [optional] :
Type : double
Description : Difference between debit and total.

HasDifference (has difference) [optional] :
Type : bool
Description : Difference between debit and total is not zero.

Notes (notes) [optional] :
Type : widestring
Description : Notes.

Coins (coins) [optional] :
Type : widestring
Description : Coins.

Cheques (cheques) [optional] :
Type : widestring
Description : Cheques.

Text

```
SafebagClose (32, "2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", "safebag no.",
"safebag info", 0, Debit:  0.0, Total:  0.0, Difference:
0.0, HasDifference:  1, Notes:  "notes", Coins:  "coins",
Cheques:  "cheques")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSafebagClose(
  const __int32 aWorkingPlace,
  const TGLibDateTime& aStartTime,
  const TGLibDateTime& aStopTime,
  const wchar_t* aSafebagNo,
  const wchar_t* aSafebagInfo,
  const SafebagStep aStepID,
  const double* aDebit,
  const double* aTotal,
  const double* aDifference,
  const bool* aHasDifference,
  const wchar_t* aNotes,
  const wchar_t* aCoins,
  const wchar_t* aCheques);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSafebagClose(
  ConstHGscAction anAction,
  __int32& aWorkingPlace,
  TGLibDateTime& aStartTime,
  TGLibDateTime& aStopTime,
  const wchar_t*& aSafebagNo,
  const wchar_t*& aSafebagInfo,
  SafebagStep& aStepID,
  const double*& aDebit,
  const double*& aTotal,
  const double*& aDifference,
  const bool*& aHasDifference,
  const wchar_t*& aNotes,
  const wchar_t*& aCoins,
  const wchar_t*& aCheques);
```

Delphi – Create

```
function GscAct_CreateSafebagClose(
  aWorkingPlace: Integer;
  var aStartTime: TGLibDateTime;
  var aStopTime: TGLibDateTime;
  aSafebagNo: PWideChar;
  aSafebagInfo: PWideChar;
  aStepID: SafebagStep;
  aDebit: PDouble;
  aTotal: PDouble;
  aDifference: PDouble;
  aHasDifference: PBoolean;
  aNotes: PWideChar;
  aCoins: PWideChar;
  aCheques: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSafebagClose(
  const anAction: HGscAction;
  out aWorkingPlace: Integer;
  out aStartTime: TGLibDateTime;
  out aStopTime: TGLibDateTime;
  out aSafebagNo: PWideChar;
  out aSafebagInfo: PWideChar;
  out aStepID: SafebagStep;
  out aDebit: PDouble;
  out aTotal: PDouble;
  out aDifference: PDouble;
  out aHasDifference: PBoolean;
  out aNotes: PWideChar;
  out aCoins: PWideChar;
  out aCheques: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

8.2 Safebag data

SafebagData (WorkingPlace, StartTime, SafebagNo, SafebagInfo, StepID, *Debit*, *Total*, *Difference*, *HasDifference*, *Notes*, *Coins*, *Cheques*)

Description : *Safebag data.*

Code : ac_SafebagData (402)

Class : ak_CashManagement (8)

Parameters :

WorkingPlace (working place) :
Type : __int32
Description : Working place no.

StartTime (start time) :
Type : TGLibDateTime
Description : Start time stamp.

SafebagNo (safebag no.) :
Type : widestring
Description : Safebag no.

SafebagInfo (safebag info) :
Type : widestring
Description : Additional info about safebag.

StepID (step ID) :
Type : SafebagStep
Description : Processing step ID.

Debit (debit) [optional] :
Type : double
Description : Debit.

Total (total) [optional] :
Type : double
Description : Total.

Difference (difference) [optional] :
Type : double
Description : Difference between debit and total.

HasDifference (has difference) [optional] :
Type : bool
Description : Difference between debit and total is not zero.

Notes (notes) [optional] :
Type : widestring
Description : Notes.

Coins (coins) [optional] :
Type : widestring
Description : Coins.

Cheques (cheques) [optional] :
Type : widestring
Description : Cheques.

Text

```
SafebagData (32, "2013/09/05 14:59:59,999 GMT+02:00",
"safebag no.", "safebag info", 0, Debit: 0.0, Total:
0.0, Difference: 0.0, HasDifference: 1, Notes:
"notes", Coins: "coins", Cheques: "cheques")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateSafebagData(
  const __int32 aWorkingPlace,
  const TGLibDateTime& aStartTime,
  const wchar_t* aSafebagNo,
  const wchar_t* aSafebagInfo,
  const SafebagStep aStepID,
  const double* aDebit,
  const double* aTotal,
  const double* aDifference,
  const bool* aHasDifference,
  const wchar_t* aNotes,
  const wchar_t* aCoins,
  const wchar_t* aCheques);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSafebagData(
  ConstHGscAction anAction,
  __int32& aWorkingPlace,
  TGLibDateTime& aStartTime,
  const wchar_t*& aSafebagNo,
  const wchar_t*& aSafebagInfo,
  SafebagStep& aStepID,
  const double*& aDebit,
  const double*& aTotal,
  const double*& aDifference,
  const bool*& aHasDifference,
  const wchar_t*& aNotes,
  const wchar_t*& aCoins,
  const wchar_t*& aCheques);
```

Delphi – Create

```
function GscAct_CreateSafebagData(
  aWorkingPlace: Integer;
  var aStartTime: TGLibDateTime;
  aSafebagNo: PWideChar;
  aSafebagInfo: PWideChar;
  aStepID: SafebagStep;
  aDebit: PDouble;
  aTotal: PDouble;
  aDifference: PDouble;
  aHasDifference: PBoolean;
  aNotes: PWideChar;
  aCoins: PWideChar;
  aCheques: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSafebagData(
  const anAction: HGscAction;
  out aWorkingPlace: Integer;
  out aStartTime: TGLibDateTime;
  out aSafebagNo: PWideChar;
  out aSafebagInfo: PWideChar;
  out aStepID: SafebagStep;
  out aDebit: PDouble;
  out aTotal: PDouble;
  out aDifference: PDouble;
  out aHasDifference: PBoolean;
  out aNotes: PWideChar;
  out aCoins: PWideChar;
  out aCheques: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

8.3 Safebag open

```
SafebagOpen (WorkingPlace, StartTime, SafebagNo,
SafebagInfo, StepID)
```

Description : *Safebag open.*

Code : `ac_SafebagOpen` (400)

Class : `ak_CashManagement` (8)

Parameters :

WorkingPlace (working place) :
Type : __int32
Description : Working place no.

StartTime (start time) :
Type : TGLibDateTime
Description : Start time stamp.

SafebagNo (safebag no.) :
Type : widestring
Description : Safebag no.

SafebagInfo (safebag info) :
Type : widestring
Description : Additional info about safebag.

StepID (step ID) :
Type : SafebagStep
Description : Processing step ID.

Text

```
SafebagOpen (32, "2013/09/05 14:59:59,999 GMT+02:00",
"safebag no.", "safebag info", 0)
```

C++ - Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSafebagOpen(
  const __int32 aWorkingPlace,
  const TGLibDateTime& aStartTime,
  const wchar_t* aSafebagNo,
  const wchar_t* aSafebagInfo,
  const SafebagStep aStepID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSafebagOpen(
  ConstHGscAction anAction,
  __int32& aWorkingPlace,
  TGLibDateTime& aStartTime,
  const wchar_t*& aSafebagNo,
  const wchar_t*& aSafebagInfo,
  SafebagStep& aStepID);
```

Delphi – Create

```
function GscAct_CreateSafebagOpen(
  aWorkingPlace: Integer;
  var aStartTime: TGLibDateTime;
  aSafebagNo: PWideChar;
  aSafebagInfo: PWideChar;
  aStepID: SafebagStep)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSafebagOpen(
  const anAction: HGscAction;
  out aWorkingPlace: Integer;
  out aStartTime: TGLibDateTime;
  out aSafebagNo: PWideChar;
  out aSafebagInfo: PWideChar;
  out aStepID: SafebagStep)
  : Boolean; stdcall; external GscActionsDll;
```

8.4 Safebag passing of risk data

SafebagPassingOfRiskData (WorkingPlace, StartTime, SafebagNo, SafebagInfo, StepID, UserID1, UserID2, *TourNumber*, *TargetWorkingPlace*, *PassingOfRiskType*)

Description : *Safebag passing of risk data.*

Code : ac_SafebagPassingOfRiskData (410)

Class : ak_CashManagement (8)

Parameters :

WorkingPlace (working place) :
Type : __int32
Description : Working place no.

StartTime (start time) :
Type : TGLibDateTime
Description : Start time stamp.

SafebagNo (safebag no.) :
Type : widestring
Description : Safebag no.

SafebagInfo (safebag info) :
Type : widestring
Description : Additional info about safebag.

StepID (step ID) :
Type : SafebagStep
Description : Processing step ID.

UserID1 (user 1) :
Type : __int32
Description : User who gives the safebag.

UserID2 (user 2) :
Type : __int32
Description : User who takes the safebag.

TourNumber (tour no) [optional] :
Type : __int32
Description : Optional tour no.

TargetWorkingPlace (target working place) [optional] :
Type : widestring
Description : Target working place.

PassingOfRiskType (passing of risk type) [optional] :
Type : widestring
Description : Passing of risk type.

Text

```
SafebagPassingOfRiskData (32, "2013/09/05 14:59:59,999 GMT+02:00",
"safebag no.", "safebag info", 0, 32, 32, TourNumber:
32, TargetWorkingPlace:  "target working place",
PassingOfRiskType:  "passing of risk type")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSafebagPassingOfRiskData(
  const __int32 aWorkingPlace,
  const TGLibDateTime& aStartTime,
  const wchar_t* aSafebagNo,
  const wchar_t* aSafebagInfo,
  const SafebagStep aStepID,
  const __int32 aUserID1,
  const __int32 aUserID2,
  const __int32* aTourNumber,
  const wchar_t* aTargetWorkingPlace,
  const wchar_t* aPassingOfRiskType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSafebagPassingOfRiskData(
  ConstHGscAction anAction,
  __int32& aWorkingPlace,
  TGLibDateTime& aStartTime,
  const wchar_t*& aSafebagNo,
  const wchar_t*& aSafebagInfo,
  SafebagStep& aStepID,
  __int32& aUserID1,
  __int32& aUserID2,
  const __int32*& aTourNumber,
  const wchar_t*& aTargetWorkingPlace,
  const wchar_t*& aPassingOfRiskType);
```

Delphi – Create

```
function GscAct_CreateSafebagPassingOfRiskData(
  aWorkingPlace: Integer;
  var aStartTime: TGLibDateTime;
  aSafebagNo: PWideChar;
  aSafebagInfo: PWideChar;
  aStepID: SafebagStep;
  aUserID1: Integer;
  aUserID2: Integer;
  aTourNumber: PInteger;
  aTargetWorkingPlace: PWideChar;
  aPassingOfRiskType: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSafebagPassingOfRiskData(
  const anAction: HGscAction;
  out aWorkingPlace: Integer;
  out aStartTime: TGLibDateTime;
  out aSafebagNo: PWideChar;
  out aSafebagInfo: PWideChar;
  out aStepID: SafebagStep;
  out aUserID1: Integer;
  out aUserID2: Integer;
  out aTourNumber: PInteger;
  out aTargetWorkingPlace: PWideChar;
  out aPassingOfRiskType: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

8.5 Safebag passing of risk start

SafebagPassingOfRiskStart (WorkingPlace, StartTime, SafebagNo, SafebagInfo, StepID, UserID1, UserID2, *TourNumber*, *TargetWorkingPlace*, *PassingOfRiskType*)

GEUTEBRÜCK

Description : *Safebag passing of risk start.*

Code : `ac_SafebagPassingOfRiskStart` (407)

Class : `ak_CashManagement` (8)

Parameters :

WorkingPlace (working place) :
Type : __int32
Description : Working place no.

StartTime (start time) :
Type : TGLibDateTime
Description : Start time stamp.

SafebagNo (safebag no.) :
Type : widestring
Description : Safebag no.

SafebagInfo (safebag info) :
Type : widestring
Description : Additional info about safebag.

StepID (step ID) :
Type : SafebagStep
Description : Processing step ID.

UserID1 (user 1) :
Type : __int32
Description : User who gives the safebag.

UserID2 (user 2) :
Type : __int32
Description : User who takes the safebag.

TourNumber (tour no) [optional] :
Type : __int32
Description : Optional tour no.

TargetWorkingPlace (target working place) [optional] :
Type : widestring
Description : Target working place.

PassingOfRiskType (passing of risk type) [optional] :
Type : widestring
Description : Passing of risk type.

Text

```
SafebagPassingOfRiskStart (32, "2013/09/05 14:59:59,999 GMT+02:00",
"safebag no.", "safebag info", 0, 32, 32, TourNumber:
32, TargetWorkingPlace:  "target working place",
PassingOfRiskType:  "passing of risk type")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSafebagPassingOfRiskStart(
  const __int32 aWorkingPlace,
  const TGLibDateTime& aStartTime,
  const wchar_t* aSafebagNo,
  const wchar_t* aSafebagInfo,
  const SafebagStep aStepID,
  const __int32 aUserID1,
  const __int32 aUserID2,
  const __int32* aTourNumber,
  const wchar_t* aTargetWorkingPlace,
  const wchar_t* aPassingOfRiskType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSafebagPassingOfRiskStart(
  ConstHGscAction anAction,
  __int32& aWorkingPlace,
  TGLibDateTime& aStartTime,
  const wchar_t*& aSafebagNo,
  const wchar_t*& aSafebagInfo,
  SafebagStep& aStepID,
  __int32& aUserID1,
  __int32& aUserID2,
  const __int32*& aTourNumber,
  const wchar_t*& aTargetWorkingPlace,
  const wchar_t*& aPassingOfRiskType);
```

Delphi – Create

```
function GscAct_CreateSafebagPassingOfRiskStart(
  aWorkingPlace: Integer;
  var aStartTime: TGLibDateTime;
  aSafebagNo: PWideChar;
  aSafebagInfo: PWideChar;
  aStepID: SafebagStep;
  aUserID1: Integer;
  aUserID2: Integer;
  aTourNumber: PInteger;
  aTargetWorkingPlace: PWideChar;
  aPassingOfRiskType: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeSafebagPassingOfRiskStart(
  const anAction: HGscAction;
  out aWorkingPlace: Integer;
  out aStartTime: TGLibDateTime;
  out aSafebagNo: PWideChar;
  out aSafebagInfo: PWideChar;
  out aStepID: SafebagStep;
  out aUserID1: Integer;
  out aUserID2: Integer;
  out aTourNumber: PInteger;
  out aTargetWorkingPlace: PWideChar;
  out aPassingOfRiskType: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

8.6 Safebag passing of risk stop

SafebagPassingOfRiskStop (WorkingPlace, StartTime, StopTime, SafebagNo, SafebagInfo, StepID, UserID1, UserID2, *TourNumber*, *TargetWorkingPlace*, *PassingOfRiskType*)

Description : *Safebag passing of risk stop.*

Code : ac_SafebagPassingOfRiskStop (413)

Class : ak_CashManagement (8)

Parameters :

WorkingPlace (working place) :
Type : __int32
Description : Working place no.

StartTime (start time) :
Type : TGLibDateTime
Description : Start time stamp.

StopTime (stop time) :
Type : TGLibDateTime
Description : Stop time stamp.

SafebagNo (safebag no.) :
Type : widestring
Description : Safebag no.

SafebagInfo (safebag info) :
Type : widestring
Description : Additional info about safebag.

StepID (step ID) :
Type : SafebagStep
Description : Processing step ID.

UserID1 (user 1) :
Type : __int32
Description : User who gives the safebag.

UserID2 (user 2) :
Type : __int32
Description : User who takes the safebag.

TourNumber (tour no) [optional] :
Type : __int32
Description : Optional tour no.

TargetWorkingPlace (target working place) [optional] :
Type : widestring
Description : Target working place.

PassingOfRiskType (passing of risk type) [optional] :
Type : widestring
Description : Passing of risk type.

Text

```
SafebagPassingOfRiskStop (32, "2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", "safebag
no.", "safebag info", 0, 32, 32, TourNumber:  32,
TargetWorkingPlace:  "target working place",
PassingOfRiskType:  "passing of risk type")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSafebagPassingOfRiskStop(
  const __int32 aWorkingPlace,
  const TGLibDateTime& aStartTime,
  const TGLibDateTime& aStopTime,
  const wchar_t* aSafebagNo,
  const wchar_t* aSafebagInfo,
  const SafebagStep aStepID,
  const __int32 aUserID1,
  const __int32 aUserID2,
  const __int32* aTourNumber,
  const wchar_t* aTargetWorkingPlace,
  const wchar_t* aPassingOfRiskType);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSafebagPassingOfRiskStop(
  ConstHGscAction anAction,
  __int32& aWorkingPlace,
  TGLibDateTime& aStartTime,
  TGLibDateTime& aStopTime,
  const wchar_t*& aSafebagNo,
  const wchar_t*& aSafebagInfo,
  SafebagStep& aStepID,
  __int32& aUserID1,
  __int32& aUserID2,
  const __int32*& aTourNumber,
  const wchar_t*& aTargetWorkingPlace,
  const wchar_t*& aPassingOfRiskType);
```

Delphi – Create

```
function GscAct_CreateSafebagPassingOfRiskStop(
  aWorkingPlace: Integer;
  var aStartTime: TGLibDateTime;
  var aStopTime: TGLibDateTime;
  aSafebagNo: PWideChar;
  aSafebagInfo: PWideChar;
  aStepID: SafebagStep;
  aUserID1: Integer;
  aUserID2: Integer;
  aTourNumber: PInteger;
  aTargetWorkingPlace: PWideChar;
  aPassingOfRiskType: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSafebagPassingOfRiskStop(
  const anAction: HGscAction;
  out aWorkingPlace: Integer;
  out aStartTime: TGLibDateTime;
  out aStopTime: TGLibDateTime;
  out aSafebagNo: PWideChar;
  out aSafebagInfo: PWideChar;
  out aStepID: SafebagStep;
  out aUserID1: Integer;
  out aUserID2: Integer;
  out aTourNumber: PInteger;
  out aTargetWorkingPlace: PWideChar;
  out aPassingOfRiskType: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

9 Device information

All actions for low-level notification of the device or media channels changes.

9.1 Device found

```
DeviceFound (Type, Name, Serial)
```

Description : *This action will be fired when the USB or NET device is connected to the system. It is also fired at start-up for all detected devices.*

Code : `ac_DeviceFound` (10)

Class : `ak_Device` (5)

Parameters :

Type (device type) [DeviceType] :
Type : string
Description : Type of the device.

Name (device name) [DeviceName] :
Type : string
Description : Device name if assigned in setup, empty otherwise.

Serial (serial ID) :
Type : string
Description : Serial ID of the device.

Text

```
DeviceFound ("device type", "device name", "serial ID")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDeviceFound(
  const char* aType,
  const char* aName,
  const char* aSerial);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDeviceFound(
  ConstHGscAction anAction,
  const char*& aType,
  const char*& aName,
  const char*& aSerial);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateDeviceFound(
  aType: PAnsiChar;
  aName: PAnsiChar;
  aSerial: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDeviceFound(
  const anAction: HGscAction;
  out aType: PAnsiChar;
  out aName: PAnsiChar;
  out aSerial: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

9.2 Device plugin error

DevicePluginError (Channel, Type, *SubType*, Name, Serial, ErrorClass, *ErrorCode*, *Description*)

Description : *This action notifies device plugin error.*

Code : ac_DevicePluginError (14)

Class : ak_Device (5)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Type (device type) [DeviceType] :
Type : string
Description : Type of the device.

SubType (device subtype) [optional] [DeviceSubType] :
Type : string
Description : Subtype of the device.

Name (device name) [DeviceName] :
Type : string
Description : Device name.

Serial (serial ID) :
Type : string
Description : Serial ID of the device.

ErrorClass (error class) :
Type : PlcPluginError
Description : Error class of the error occured.

ErrorCode (error code) [optional] :
Type : __int32
Description : Plugin type specific error code.

Description (description) [optional] :
Type : string
Description : Error description.

Text

```
DevicePluginError (32, "device type", SubType: "device
subtype", "device name", "serial ID", 0, ErrorCode: 32,
Description: "description")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateDevicePluginError(
  const TPlcMediaChannelID& aChannel,
  const char* aType,
  const char* aSubType,
  const char* aName,
  const char* aSerial,
  const PlcPluginError aErrorClass,
  const __int32* aErrorCode,
  const char* aDescription);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDevicePluginError(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const char*& aType,
  const char*& aSubType,
  const char*& aName,
  const char*& aSerial,
  PlcPluginError& aErrorClass,
  const __int32*& aErrorCode,
  const char*& aDescription);
```

Delphi – Create

```
function GscAct_CreateDevicePluginError(
  var aChannel: TPlcMediaChannelID;
  aType: PAnsiChar;
  aSubType: PAnsiChar;
  aName: PAnsiChar;
  aSerial: PAnsiChar;
  aErrorClass: PlcPluginError;
  aErrorCode: PInteger;
  aDescription: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDevicePluginError(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aType: PAnsiChar;
  out aSubType: PAnsiChar;
  out aName: PAnsiChar;
  out aSerial: PAnsiChar;
  out aErrorClass: PlcPluginError;
  out aErrorCode: PInteger;
  out aDescription: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

9.3 Device plugin state

DevicePluginState (Channel, Type, *SubType*, Name, Serial, State, *InternalState*, *Description*)

Description : *This action notifies device plugin state.*

Code : ac_DevicePluginState (15)

Class : ak_Device (5)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Type (device type) [DeviceType] :
Type : string
Description : Type of the device.

SubType (device subtype) [optional] [DeviceSubType] :
Type : string
Description : Subtype of the device.

Name (device name) [DeviceName] :
Type : string
Description : Device name.

Serial (serial ID) :
Type : string
Description : Serial ID of the device.

State (plugin state) :
Type : PlcPluginState
Description : New plugin device state.

InternalState (internal state) [optional] :
Type : __int32
Description : Plugin device specific state.

Description (description) [optional] :
Type : string
Description : State description.

Text

```
DevicePluginState (32, "device type", SubType:  "device
subtype", "device name", "serial ID", 0, InternalState:
32, Description:  "description")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDevicePluginState(
  const TPlcMediaChannelID& aChannel,
  const char* aType,
  const char* aSubType,
  const char* aName,
  const char* aSerial,
  const PlcPluginState aState,
  const __int32* aInternalState,
  const char* aDescription);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDevicePluginState(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const char*& aType,
  const char*& aSubType,
  const char*& aName,
  const char*& aSerial,
  PlcPluginState& aState,
  const __int32*& aInternalState,
  const char*& aDescription);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateDevicePluginState(
  var aChannel: TPlcMediaChannelID;
  aType: PAnsiChar;
  aSubType: PAnsiChar;
  aName: PAnsiChar;
  aSerial: PAnsiChar;
  aState: PlcPluginState;
  aInternalState: PInteger;
  aDescription: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDevicePluginState(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aType: PAnsiChar;
  out aSubType: PAnsiChar;
  out aName: PAnsiChar;
  out aSerial: PAnsiChar;
  out aState: PlcPluginState;
  out aInternalState: PInteger;
  out aDescription: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

9.4 Device reattached

DeviceReattached (Type, Name, Serial)

Description : *This action will be fired when the USB or NET device is reattached to the system.*

Code : ac_DeviceReattached (13)

Class : ak_Device (5)

Parameters :

Type (device type) [DeviceType] :
Type : string
Description : Type of the device.

Name (device name) [DeviceName] :
Type : string
Description : Device name if assigned in setup, empty otherwise.

Serial (serial ID) :
Type : string
Description : Serial ID of the device.

Text

```
DeviceReattached ("device type", "device name", "serial
ID")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDeviceReattached(
  const char* aType,
  const char* aName,
  const char* aSerial);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDeviceReattached(
  ConstHGscAction anAction,
  const char*& aType,
  const char*& aName,
  const char*& aSerial);
```

Delphi – Create

```
function GscAct_CreateDeviceReattached(
  aType: PAnsiChar;
  aName: PAnsiChar;
  aSerial: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDeviceReattached(
  const anAction: HGscAction;
  out aType: PAnsiChar;
  out aName: PAnsiChar;
  out aSerial: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

9.5 Device removed

DeviceRemoved (Type, Name, Serial)

Description : *This action will be fired when the USB or NET device is disconnected from the system. It is also fired at the start-up for all parameterized but not present devices.*

Code : ac_DeviceRemoved (11)

GEUTEBRÜCK

Class : ak_Device (5)

Parameters :

Type (device type) [DeviceType] :
Type : string
Description : Type of the device.

Name (device name) [DeviceName] :
Type : string
Description : Device name if assigned in setup, empty otherwise.

Serial (serial ID) :
Type : string
Description : Serial ID of the device.

Text

```
DeviceRemoved ("device type", "device name", "serial ID")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDeviceRemoved(
  const char* aType,
  const char* aName,
  const char* aSerial);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDeviceRemoved(
  ConstHGscAction anAction,
  const char*& aType,
  const char*& aName,
  const char*& aSerial);
```

Delphi – Create

```
function GscAct_CreateDeviceRemoved(
  aType: PAnsiChar;
  aName: PAnsiChar;
  aSerial: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDeviceRemoved(
  const anAction: HGscAction;
  out aType: PAnsiChar;
  out aName: PAnsiChar;
  out aSerial: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

9.6 New firmware received

```
DeviceNewFirmware (Type, Name, Serial, Firmware)
```

Description : *This action will be fired when the USB or NET device has got the new firmware.*

Code : `ac_DeviceNewFirmware` (12)

Class : `ak_Device` (5)

Parameters :

Type (device type) [DeviceType] :
Type : string
Description : Type of the device.

Name (device name) [DeviceName] :
Type : string
Description : Device name if assigned in setup, empty otherwise.

Serial (serial ID) :
Type : string
Description : Serial ID of the device.

Firmware (firmware serial) :
Type : string
Description : Serial ID of the firmware.

Text

```
DeviceNewFirmware ("device type", "device name", "serial
ID", "firmware serial")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDeviceNewFirmware(
  const char* aType,
  const char* aName,
  const char* aSerial,
  const char* aFirmware);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDeviceNewFirmware(
  ConstHGscAction anAction,
  const char*& aType,
  const char*& aName,
  const char*& aSerial,
  const char*& aFirmware);
```

Delphi – Create

```
function GscAct_CreateDeviceNewFirmware(
  aType: PAnsiChar;
  aName: PAnsiChar;
  aSerial: PAnsiChar;
  aFirmware: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDeviceNewFirmware(
  const anAction: HGscAction;
  out aType: PAnsiChar;
  out aName: PAnsiChar;
  out aSerial: PAnsiChar;
  out aFirmware: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

10 Digital contacts

All actions for handling digital inputs and outputs.

10.1 Case has been closed

CaseClosed ()

Description : *Case has been closed.*

Code : ac_CaseClosed (446)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
CaseClosed ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCaseClosed();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCaseClosed(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateCaseClosed()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCaseClosed(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.2 Case has been opened

CaseOpened ()

Description : *Case has been opened.*

Code : ac_CaseOpened (445)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
CaseOpened ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCaseOpened();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCaseOpened(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateCaseOpened()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCaseOpened(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.3 Digital input

```
DigitalInput (Contact, State)
```

Description : *This action will be fired when the state of the digital input has changed.*

Code : ac_DigitalInput (30)

Class : ak_DigitalContacts (4)

Parameters :

Contact (contact) [InputContact] :
Type : TMediaChannelID
Description : Contact.

State (state) :
Type : DigitalInputState
Description : New state.

Text

```
DigitalInput (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDigitalInput(
  const TPlcMediaChannelID& aContact,
  const DigitalInputState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDigitalInput(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aContact,
  DigitalInputState& aState);
```

Delphi – Create

```
function GscAct_CreateDigitalInput(
  var aContact: TPlcMediaChannelID;
  aState: DigitalInputState)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDigitalInput(
  const anAction: HGscAction;
  out aContact: TPlcMediaChannelID;
  out aState: DigitalInputState)
  : Boolean; stdcall; external GscActionsDll;
```

10.4 IOI43 reset mainboard

IOI43ResetMainboard ()

Description : *Reset mainboard using IOI43a/ab USB Alarm-I/O.*

Code : ac_IOI43ResetMainboard (423)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
IOI43ResetMainboard ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIOI43ResetMainboard();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIOI43ResetMainboard(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateIOI43ResetMainboard()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIOI43ResetMainboard(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.5 IOI43 temperature notification

```
IOI43Temperature (ID, Temperature)
```

Description : *Temperature notification from IOI43a/ab USB Alarm-I/O.*

Code : `ac_IOI43Temperature` (424)

Class : `ak_DigitalContacts` (4)

Parameters :

ID (ID) :
Type : widestring
Description : ID of the IOI43 module (like IOI43-00).

Temperature (temperature) :
Type : __int32
Description : Temperature.

Text

```
IOI43Temperature ("ID", 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIOI43Temperature(
  const wchar_t* aID,
  const __int32 aTemperature);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIOI43Temperature(
  ConstHGscAction anAction,
  const wchar_t*& aID,
  __int32& aTemperature);
```

Delphi – Create

```
function GscAct_CreateIOI43Temperature(
  aID: PWideChar;
  aTemperature: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIOI43Temperature(
  const anAction: HGscAction;
  out aID: PWideChar;
  out aTemperature: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

10.6 IOI43 watchdog activate

IOI43WDActivate ()

Description : *Activate watchdog on IOI43a/ab USB Alarm-I/O.*

Code : ac_IOI43WDActivate (420)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
IOI43WDActivate ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIOI43WDActivate();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIOI43WDActivate(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateIOI43WDActivate()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIOI43WDActivate(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.7 IOI43 watchdog deactivate

IOI43WDDeactivate ()

Description : *Deactivate watchdog on IOI43a/ab USB Alarm-I/O.*

Code : ac_IOI43WDDeactivate (421)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
IOI43WDDeactivate ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIOI43WDDeactivate();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIOI43WDDeactivate(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateIOI43WDDeactivate()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIOI43WDDeactivate(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.8 IOI43 watchdog trigger

IOI43WDTrigger ()

Description : *Trigger watchdog on IOI43a/ab USB Alarm-I/O.*

Code : ac_IOI43WDTrigger (422)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

IOI43WDTrigger ()

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIOI43WDTrigger();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIOI43WDTrigger(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateIOI43WDTrigger()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIOI43WDTrigger(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

10.9 Key pressed

KeyPressed (Key)

Description : *This action is notified if one of the GEVISCOPE system keys is pressed.*

Code : ac_KeyPressed (35)

Class : ak_DigitalContacts (4)

Parameters :

Key (key) :
Type : SystemKey
Description : System key.

Text

```
KeyPressed (0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateKeyPressed(
  const SystemKey aKey);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeKeyPressed(
  ConstHGscAction anAction,
  SystemKey& aKey);
```

Delphi – Create

```
function GscAct_CreateKeyPressed(
  aKey: SystemKey)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeKeyPressed(
  const anAction: HGscAction;
  out aKey: SystemKey)
  : Boolean; stdcall; external GscActionsDll;
```

10.10 Key released

```
KeyReleased (Key)
```

Description : *This action is notified if one of the GEVISCOPE system keys is released.*

Code : ac_KeyReleased (36)

Class : ak_DigitalContacts (4)

Parameters :

Key (key) :
Type : SystemKey
Description : System key.

Text

```
KeyReleased (0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateKeyReleased(
  const SystemKey aKey);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeKeyReleased(
  ConstHGscAction anAction,
  SystemKey& aKey);
```

Delphi – Create

```
function GscAct_CreateKeyReleased(
  aKey: SystemKey)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeKeyReleased(
  const anAction: HGscAction;
  out aKey: SystemKey)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

10.11 Reset mainboard

ResetMainboard ()

Description : *Reset mainboard using IO device.*

Code : ac_ResetMainboard (444)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
ResetMainboard ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateResetMainboard();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeResetMainboard(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateResetMainboard()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeResetMainboard(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.12 Set digital output

SetDigitalOutput (Contact, State)

Description : *This action is used to modify the state of the digital output and to notify this change.*

Code : ac_SetDigitalOutput (31)

Class : ak_DigitalContacts (4)

Parameters :

Contact (contact) [OutputContact] :
Type : TMediaChannelID
Description : Contact.

State (state) :
Type : DigitalOutputState
Description : New state.

Text

```
SetDigitalOutput (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateSetDigitalOutput(
  const TPlcMediaChannelID& aContact,
  const DigitalOutputState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetDigitalOutput(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aContact,
  DigitalOutputState& aState);
```

Delphi – Create

```
function GscAct_CreateSetDigitalOutput(
  var aContact: TPlcMediaChannelID;
  aState: DigitalOutputState)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetDigitalOutput(
  const anAction: HGscAction;
  out aContact: TPlcMediaChannelID;
  out aState: DigitalOutputState)
  : Boolean; stdcall; external GscActionsDll;
```

10.13 Set system LED

```
SetLED (LED, State)
```

GEUTEBRÜCK

Description : *This action is used to turn the system LEDs on or off.*

Code : `ac_SetLED` (32)

Class : `ak_DigitalContacts` (4)

Parameters :

LED (LED) :
Type : SystemLED
Description : System LED.

State (state) :
Type : DigitalOutputState
Description : New state.

Text

```
SetLED (0, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetLED(
  const SystemLED aLED,
  const DigitalOutputState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetLED(
  ConstHGscAction anAction,
  SystemLED& aLED,
  DigitalOutputState& aState);
```

Delphi – Create

```
function GscAct_CreateSetLED(
  aLED: SystemLED;
  aState: DigitalOutputState)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetLED(
  const anAction: HGscAction;
  out aLED: SystemLED;
  out aState: DigitalOutputState)
  : Boolean; stdcall; external GscActionsDll;
```

10.14 Set system LED to blink

```
SetLEDBlink (LED, LedTimeOnMs, LedTimeOffMs)
```

Description : *This action is used to blink the system LEDs.*

Code : ac_SetLEDBlink (33)

Class : ak_DigitalContacts (4)

Parameters :

LED (LED) :
Type : SystemLED
Description : System LED.

LedTimeOnMs (LED time ON) :
Type : __int32
Description : Time in milliseconds when the LED will be switched on.

LedTimeOffMs (LED time OFF) :
Type : __int32
Description : Time in milliseconds when the LED will be switched off.

Text

```
SetLEDBlink (0, 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetLEDBlink(
  const SystemLED aLED,
  const __int32 aLedTimeOnMs,
  const __int32 aLedTimeOffMs);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetLEDBlink(
  ConstHGscAction anAction,
  SystemLED& aLED,
  __int32& aLedTimeOnMs,
  __int32& aLedTimeOffMs);
```

Delphi – Create

```
function GscAct_CreateSetLEDBlink(
  aLED: SystemLED;
  aLedTimeOnMs: Integer;
  aLedTimeOffMs: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetLEDBlink(
  const anAction: HGscAction;
  out aLED: SystemLED;
  out aLedTimeOnMs: Integer;
  out aLedTimeOffMs: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

10.15 Temperature notification

```
Temperature (SourceType, ID, Temperature)
```

Description : *Temperature notification from a/ab USB Alarm-I/O.*

Code : ac_Temperature (443)

Class : ak_DigitalContacts (4)

Parameters :

SourceType (IO device type) :
Type : IODeviceType
Description : Type of the device, which has fired the action.

ID (ID) :
Type : widestring
Description : ID of the IO module (like IOI43-00/MIO84-00/MIO168-00).

Temperature (temperature) :
Type : __int32
Description : Temperature.

Text

```
Temperature (0, "ID", 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTemperature(
  const IODeviceType aSourceType,
  const wchar_t* aID,
  const __int32 aTemperature);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTemperature(
  ConstHGscAction anAction,
  IODeviceType& aSourceType,
  const wchar_t*& aID,
  __int32& aTemperature);
```

Delphi – Create

```
function GscAct_CreateTemperature(
  aSourceType: IODeviceType;
  aID: PWideChar;
  aTemperature: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeTemperature(
  const anAction: HGscAction;
  out aSourceType: IODeviceType;
  out aID: PWideChar;
  out aTemperature: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

10.16 Watchdog activate

WatchdogActivate ()

Description : *Activate watchdog.*

Code : ac_WatchdogActivate (440)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
WatchdogActivate ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateWatchdogActivate();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeWatchdogActivate(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateWatchdogActivate()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeWatchdogActivate(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.17 Watchdog deactivate

WatchdogDeactivate ()

Description : *Deactivate watchdog.*

Code : ac_WatchdogDeactivate (441)

Class : ak_DigitalContacts (4)

This action has no parameters.

Text

```
WatchdogDeactivate ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateWatchdogDeactivate();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeWatchdogDeactivate(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateWatchdogDeactivate()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeWatchdogDeactivate(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

10.18 Watchdog trigger

WatchdogTrigger ()

Description : *Trigger watchdog.*

Code : `ac_WatchdogTrigger` (442)

Class : `ak_DigitalContacts` (4)

This action has no parameters.

Text

```
WatchdogTrigger ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateWatchdogTrigger();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeWatchdogTrigger(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateWatchdogTrigger()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeWatchdogTrigger(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

11 Imex

Image export.

11.1 Imex capacity file auto deleted

ImexCapacityFileAutoDeleted (Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByImageFile, PercentFree, PercentAllocated, PercentAllocatedByImageFile, TotalFileSize, NumberOfFiles)

Description : *Imex capacity warning: file auto deleted.*

Code : ac_ImexCapacityFileAutoDeleted (554)

Class : ak_Imex (17)

Parameters :

Warning (warning) :
Type : ABCapacityWarning
Description : Warning.

Destination (destination) :
Type : TResourceID
Description : Destination.

TotalCapacity (total capacity) :
Type : __int64
Description : Total capacity.

FreeCapacity (free capacity) :
Type : __int64
Description : Free capacity.

AllocatedByImageFile (allocated by image file) :
Type : __int64
Description : Allocated by image file.

PercentFree (percent free) :
Type : __int32
Description : Percent free.

PercentAllocated (percent allocated) :
Type : __int32
Description : Percent allocated.

PercentAllocatedByImageFile (percent allocated by image file) :
Type : __int32
Description : Percent allocated by image file.

TotalFileSize (total file size) :
Type : __int64
Description : Total file size.

NumberOfFiles (number of files) :
Type : __int32
Description : Number of files.

Text

```
ImexCapacityFileAutoDeleted (0, "destination", 64, 64,
64, 32, 32, 32, 64, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateImexCapacityFileAutoDeleted(
  const ABCapacityWarning aWarning,
  const TPlcResourceID& aDestination,
  const __int64& aTotalCapacity,
  const __int64& aFreeCapacity,
  const __int64& aAllocatedByImageFile,
  const __int32 aPercentFree,
  const __int32 aPercentAllocated,
  const __int32 aPercentAllocatedByImageFile,
  const __int64& aTotalFileSize,
  const __int32 aNumberOfFiles);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeImexCapacityFileAutoDeleted(
  ConstHGscAction anAction,
  ABCapacityWarning& aWarning,
  TPlcResourceID& aDestination,
  __int64& aTotalCapacity,
  __int64& aFreeCapacity,
  __int64& aAllocatedByImageFile,
  __int32& aPercentFree,
  __int32& aPercentAllocated,
  __int32& aPercentAllocatedByImageFile,
  __int64& aTotalFileSize,
  __int32& aNumberOfFiles);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateImexCapacityFileAutoDeleted(
  aWarning: ABCapacityWarning;
  var aDestination: TPlcResourceID;
  var aTotalCapacity: Int64;
  var aFreeCapacity: Int64;
  var aAllocatedByImageFile: Int64;
  aPercentFree: Integer;
  aPercentAllocated: Integer;
  aPercentAllocatedByImageFile: Integer;
  var aTotalFileSize: Int64;
  aNumberOfFiles: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeImexCapacityFileAutoDeleted(
  const anAction: HGscAction;
  out aWarning: ABCapacityWarning;
  out aDestination: TPlcResourceID;
  out aTotalCapacity: Int64;
  out aFreeCapacity: Int64;
  out aAllocatedByImageFile: Int64;
  out aPercentFree: Integer;
  out aPercentAllocated: Integer;
  out aPercentAllocatedByImageFile: Integer;
  out aTotalFileSize: Int64;
  out aNumberOfFiles: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

11.2 Imex capacity out of disk space.

ImexCapacityOutOfDiskSpace (Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByImageFile, PercentFree, PercentAllocated, PercentAllocatedByImageFile)

Description : *Imex capacity warning: out of disk space.*

Code : ac_ImexCapacityOutOfDiskSpace (552)

Class : ak_Imex (17)

Parameters :

Warning (warning) :
Type : ABCapacityWarning
Description : Warning.

Destination (destination) :
Type : TResourceID
Description : Destination.

TotalCapacity (total capacity) :
Type : __int64
Description : Total capacity.

FreeCapacity (free capacity) :
Type : __int64
Description : Free capacity.

AllocatedByImageFile (allocated by image file) :
Type : __int64
Description : Allocated by image file.

PercentFree (percent free) :
Type : __int32
Description : Percent free.

PercentAllocated (percent allocated) :
Type : __int32
Description : Percent allocated.

PercentAllocatedByImageFile (percent allocated by image file) :
Type : __int32
Description : Percent allocated by image file.

Text

```
ImexCapacityOutOfDiskSpace (0, "destination", 64, 64, 64,
32, 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateImexCapacityOutOfDiskSpace(
  const ABCapacityWarning aWarning,
  const TPlcResourceID& aDestination,
  const __int64& aTotalCapacity,
  const __int64& aFreeCapacity,
  const __int64& aAllocatedByImageFile,
  const __int32 aPercentFree,
  const __int32 aPercentAllocated,
  const __int32 aPercentAllocatedByImageFile);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeImexCapacityOutOfDiskSpace(
  ConstHGscAction anAction,
  ABCapacityWarning& aWarning,
  TPlcResourceID& aDestination,
  __int64& aTotalCapacity,
  __int64& aFreeCapacity,
  __int64& aAllocatedByImageFile,
  __int32& aPercentFree,
  __int32& aPercentAllocated,
  __int32& aPercentAllocatedByImageFile);
```

Delphi – Create

```
function GscAct_CreateImexCapacityOutOfDiskSpace(
  aWarning: ABCapacityWarning;
  var aDestination: TPlcResourceID;
  var aTotalCapacity: Int64;
  var aFreeCapacity: Int64;
  var aAllocatedByImageFile: Int64;
  aPercentFree: Integer;
  aPercentAllocated: Integer;
  aPercentAllocatedByImageFile: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeImexCapacityOutOfDiskSpace(
  const anAction: HGscAction;
  out aWarning: ABCapacityWarning;
  out aDestination: TPlcResourceID;
  out aTotalCapacity: Int64;
  out aFreeCapacity: Int64;
  out aAllocatedByImageFile: Int64;
  out aPercentFree: Integer;
  out aPercentAllocated: Integer;
  out aPercentAllocatedByImageFile: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

11.3 Imex capacity warning

ImexCapacityWarning (Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByImageFile, PercentFree, PercentAllocated, PercentAllocatedByImageFile)

Description : *Imex capacity warning: capacity warning.*

Code : ac_ImexCapacityWarning (550)

Class : ak_Imex (17)

Parameters :

Warning (warning) :
Type : ABCapacityWarning
Description : Warning.

Destination (destination) :
Type : TResourceID
Description : Destination.

TotalCapacity (total capacity) :
Type : __int64
Description : Total capacity.

FreeCapacity (free capacity) :
Type : __int64
Description : Free capacity.

AllocatedByImageFile (allocated by image file) :
Type : __int64
Description : Allocated by image file.

PercentFree (percent free) :
Type : __int32
Description : Percent free.

PercentAllocated (percent allocated) :
Type : __int32
Description : Percent allocated.

PercentAllocatedByImageFile (percent allocated by image file) :
Type : __int32
Description : Percent allocated by image file.

Text

```
ImexCapacityWarning (0, "destination", 64, 64, 64, 32,
32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateImexCapacityWarning(
  const ABCapacityWarning aWarning,
  const TPlcResourceID& aDestination,
  const __int64& aTotalCapacity,
  const __int64& aFreeCapacity,
  const __int64& aAllocatedByImageFile,
  const __int32 aPercentFree,
  const __int32 aPercentAllocated,
  const __int32 aPercentAllocatedByImageFile);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeImexCapacityWarning(
  ConstHGscAction anAction,
  ABCapacityWarning& aWarning,
  TPlcResourceID& aDestination,
  __int64& aTotalCapacity,
  __int64& aFreeCapacity,
  __int64& aAllocatedByImageFile,
  __int32& aPercentFree,
  __int32& aPercentAllocated,
  __int32& aPercentAllocatedByImageFile);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateImexCapacityWarning(
  aWarning: ABCapacityWarning;
  var aDestination: TPlcResourceID;
  var aTotalCapacity: Int64;
  var aFreeCapacity: Int64;
  var aAllocatedByImageFile: Int64;
  aPercentFree: Integer;
  aPercentAllocated: Integer;
  aPercentAllocatedByImageFile: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeImexCapacityWarning(
  const anAction: HGscAction;
  out aWarning: ABCapacityWarning;
  out aDestination: TPlcResourceID;
  out aTotalCapacity: Int64;
  out aFreeCapacity: Int64;
  out aAllocatedByImageFile: Int64;
  out aPercentFree: Integer;
  out aPercentAllocated: Integer;
  out aPercentAllocatedByImageFile: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

11.4 Imex export event image

ImexExportEventImage (EventID, TypeID, *Destination*, *FilePath*)

Description : *Imex export event image.*

Code : ac_ImexExportEventImage (562)

Class : ak_Imex (17)

Parameters :

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

Destination (destination) [optional] [ImexDestination] :
Type : TResourceID
Description : Destination.

FilePath (file name) [optional] :
Type : widestring
Description : File name.

Text

```
ImexExportEventImage (64, "event type", Destination:
"destination", FilePath:  "file name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateImexExportEventImage(
  const __int64& aEventID,
  const TPlcEventTypeID& aTypeID,
  const TPlcResourceID* aDestination,
  const wchar_t* aFilePath);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeImexExportEventImage(
  ConstHGscAction anAction,
  __int64& aEventID,
  TPlcEventTypeID& aTypeID,
  const TPlcResourceID*& aDestination,
  const wchar_t*& aFilePath);
```

Delphi – Create

```
function GscAct_CreateImexExportEventImage(
  var aEventID: Int64;
  var aTypeID: TPlcEventTypeID;
  aDestination: PTPlcResourceID;
  aFilePath: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeImexExportEventImage(
  const anAction: HGscAction;
  out aEventID: Int64;
  out aTypeID: TPlcEventTypeID;
  out aDestination: PTPlcResourceID;
  out aFilePath: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

11.5 Imex export image from DB

ImexExportImageFromDB (Channel, *Destination*, *FilePath*, *PictureTime*)

Description : *Imex export image from DB.*

GEUTEBRÜCK

Code : ac_ImexExportImageFromDB (560)

Class : ak_Imex (17)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Destination (destination) [optional] [ImexDestination] :
Type : TResourceID
Description : Destination.

FilePath (file name) [optional] :
Type : widestring
Description : File name.

PictureTime (picture time) [optional] :
Type : TGLibDateTime
Description : Picture time.

Text

```
ImexExportImageFromDB (32, Destination:
"destination", FilePath:  "file name", PictureTime:
"2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateImexExportImageFromDB(
  const TPlcMediaChannelID& aChannel,
  const TPlcResourceID* aDestination,
  const wchar_t* aFilePath,
  const TGLibDateTime* aPictureTime);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeImexExportImageFromDB(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const TPlcResourceID*& aDestination,
  const wchar_t*& aFilePath,
  const TGLibDateTime*& aPictureTime);
```

Delphi – Create

```
function GscAct_CreateImexExportImageFromDB(
  var aChannel: TPlcMediaChannelID;
  aDestination: PTPlcResourceID;
  aFilePath: PWideChar;
  aPictureTime: PTGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeImexExportImageFromDB(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aDestination: PTPlcResourceID;
  out aFilePath: PWideChar;
  out aPictureTime: PTGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

11.6 Imex export image from live stream

ImexExportImageFromLiveStream (Channel, *Destination*, *FilePath*)

Description : *Imex export image from live stream.*

Code : ac_ImexExportImageFromLiveStream (564)

Class : ak_Imex (17)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Destination (destination) [optional] [ImexDestination] :
Type : TResourceID
Description : Destination.

FilePath (file name) [optional] :
Type : widestring
Description : File name.

Text

```
ImexExportImageFromLiveStream (32, Destination:
"destination", FilePath:  "file name")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateImexExportImageFromLiveStream(
  const TPlcMediaChannelID& aChannel,
  const TPlcResourceID* aDestination,
  const wchar_t* aFilePath);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeImexExportImageFromLiveStream(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const TPlcResourceID*& aDestination,
  const wchar_t*& aFilePath);
```

Delphi – Create

```
function GscAct_CreateImexExportImageFromLiveStream(
  var aChannel: TPlcMediaChannelID;
  aDestination: PTPlcResourceID;
  aFilePath: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeImexExportImageFromLiveStream(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aDestination: PTPlcResourceID;
  out aFilePath: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

12 LPS

LPS messages.

12.1 LPS position data

LPSPositionData (TagID, *ScannerID*, X, Y, *Z*, *Latitude*, *Longitude*, AreaID, *CellID*, Status, *TimeStamp*, *Data*, *AreaName*)

Description : *LPS position data.*

Code : ac_LPSPositionData (460)

Class : ak_LPS (14)

Parameters :

TagID (tag ID) :
Type : __int32
Description : Tag ID.

ScannerID (scanner ID) [optional] :
Type : widestring
Description : Scanner ID or IP address.

X (X coordinate) :
Type : __int64
Description : X coordinate of cartesian coordinates.

Y (Y coordinate) :
Type : __int64
Description : Y coordinate of cartesian coordinates.

Z (Z coordinate) [optional] :
Type : __int64
Description : Z coordinate of cartesian coordinates.

Latitude (latitude) [optional] :
Type : double
Description : Latitude of geographic coordinates.

Longitude (longitude) [optional] :
Type : double
Description : Longitude of geographic coordinates.

AreaID (area ID) :
Type : __int32
Description : Area ID.

CellID (cell ID) [optional] :
Type : __int32
Description : Cell ID.

Status (status) :
Type : PlcLpsStatus
Description : Status.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

Data (data) [optional] :
Type : widestring
Description : Data.

AreaName (area name) [optional] :
Type : widestring
Description : Area Name.

Text

```
LPSPositionData (32, ScannerID: "scanner ID", 64, 64, Z:
64, Latitude:  0.0, Longitude:  0.0, 32, CellID: 32, 0,
TimeStamp:  "2013/09/05 14:59:59,999 GMT+02:00", Data:
"data", AreaName:  "area name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLPSPositionData(
  const __int32 aTagID,
  const wchar_t* aScannerID,
  const __int64& aX,
  const __int64& aY,
  const __int64* aZ,
  const double* aLatitude,
  const double* aLongitude,
  const __int32 aAreaID,
  const __int32* aCellID,
  const PlcLpsStatus aStatus,
  const TGLibDateTime* aTimeStamp,
  const wchar_t* aData,
  const wchar_t* aAreaName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLPSPositionData(
  ConstHGscAction anAction,
  __int32& aTagID,
  const wchar_t*& aScannerID,
  __int64& aX,
  __int64& aY,
  const __int64*& aZ,
  const double*& aLatitude,
  const double*& aLongitude,
  __int32& aAreaID,
  const __int32*& aCellID,
  PlcLpsStatus& aStatus,
  const TGLibDateTime*& aTimeStamp,
  const wchar_t*& aData,
  const wchar_t*& aAreaName);
```

Delphi – Create

```
function GscAct_CreateLPSPositionData(
  aTagID: Integer;
  aScannerID: PWideChar;
  var aX: Int64;
  var aY: Int64;
  aZ: PInt64;
  aLatitude: PDouble;
  aLongitude: PDouble;
  aAreaID: Integer;
  aCellID: PInteger;
  aStatus: PlcLpsStatus;
  aTimeStamp: PTGLibDateTime;
  aData: PWideChar;
  aAreaName: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLPSPositionData(
  const anAction: HGscAction;
  out aTagID: Integer;
  out aScannerID: PWideChar;
  out aX: Int64;
  out aY: Int64;
  out aZ: PInt64;
  out aLatitude: PDouble;
  out aLongitude: PDouble;
  out aAreaID: Integer;
  out aCellID: PInteger;
  out aStatus: PlcLpsStatus;
  out aTimeStamp: PTGLibDateTime;
  out aData: PWideChar;
  out aAreaName: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

12.2 LPS query position

LPSQueryPosition (TagID, *ScannerID*, *Data*)

Description : *Send position query for a Tag to LPS server.*

Code : ac_LPSQueryPosition (463)

Class : ak_LPS (14)

Parameters :

TagID (tag ID) :
Type : __int32
Description : Tag ID.

ScannerID (scanner ID) [optional] :
Type : widestring
Description : Scanner ID or IP address.

Data (data) [optional] :
Type : widestring
Description : Data.

Text

```
LPSQueryPosition (32, ScannerID: "scanner ID", Data:
"data")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLPSQueryPosition(
  const __int32 aTagID,
  const wchar_t* aScannerID,
  const wchar_t* aData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLPSQueryPosition(
  ConstHGscAction anAction,
  __int32& aTagID,
  const wchar_t*& aScannerID,
  const wchar_t*& aData);
```

Delphi – Create

```
function GscAct_CreateLPSQueryPosition(
  aTagID: Integer;
  aScannerID: PWideChar;
  aData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLPSQueryPosition(
  const anAction: HGscAction;
  out aTagID: Integer;
  out aScannerID: PWideChar;
  out aData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

13 Lenel

Lenel.

13.1 Lenel access event

LenelAccessEvent (*ID*, *Panel*, *Device*, *SecondaryDevice*, *CardNumber*, *AccessResult*, *Type*, *SubType*, Description, *SerialNumber*, *TimeStamp*, *AreaEnteredID*, *AreaExitedID*, *AssetID*, *CardholderEntered*, *Duress*, *ElevatorFloor*, *FacilityCode*, *IsReadableCard*, *IssueCode*, *CommServerHostName*, *EventText*)

Description : *Lenel OnGuard access event.*

Code : ac_LenelAccessEvent (502)

Class : ak_Lenel (16)

Parameters :

ID (ID) [optional] :
Type : LenelEventID
Description : The ID that uniquely identifies the type of this event.

Panel (panel) [optional] :
Type : widestring
Description : The name of the panel where this event originated.

Device (device) [optional] :
Type : widestring
Description : The name of the device where this event originated.

SecondaryDevice (secondary device) [optional] :
Type : __int32
Description : The ID of the secondary device where this event originated.

CardNumber (card number) [optional] :
Type : __int64
Description : The badge ID for the card that was read, if available.

AccessResult (access result) [optional] :
Type : LenelAccessResult
Description : The level of access that was granted that resulted from reading the card.

Type (type) [optional] :
Type : LenelEventType
Description : Event type i.e., duress, system, etc.

SubType (subtype) [optional] :
Type : __int32
Description : Event sub-type i.e., granted, door forced open, etc.

Description (description) :
Type : widestring
Description : A human readable, brief description of this event.

SerialNumber (serial number) [optional] :
Type : __int32
Description : A number that uniquely identifies the instance of the event for a particular panel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

AreaEnteredID (area entered) [optional] :
Type : __int32
Description : The ID of the area that was entered, if any.

AreaExitedID (area exited) [optional] :
Type : __int32
Description : The ID of the area that was exited, if any.

AssetID (asset ID) [optional] :
Type : widestring
Description : The ID of the asset related to this event, if any.

CardholderEntered (cardholder entered) [optional] :
Type : bool
Description : Whether entry was made by the cardholder.

Duress (duress) [optional] :
Type : bool
Description : Indicates whether this card access indicates an under duress/emergency state.

ElevatorFloor (elevator floor) [optional] :
Type : __int32
Description : The elevator floor on which the access event was generated, if any.

FacilityCode (facility code) [optional] :
Type : __int32
Description : The facility code for the card that was read, if available.

IsReadableCard (readable card) [optional] :
Type : bool
Description : Whether the card could be read.

IssueCode (issue code) [optional] :
Type : __int32
Description : The issue code for the card that was read, if available.

CommServerHostName (server host) [optional] :
Type : widestring
Description : Host name of the Communication server through which the event arrived.

EventText (event text) [optional] :
Type : widestring
Description : Text associated with event

GEUTEBRÜCK

Text

```
LenelAccessEvent (ID: 0, Panel: "panel", Device:
"device", SecondaryDevice: 32, CardNumber:
64, AccessResult: 0, Type: 0, SubType: 32,
"description", SerialNumber: 32, TimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00", AreaEnteredID: 32,
AreaExitedID: 32, AssetID: "asset ID", CardholderEntered:
1, Duress: 1, ElevatorFloor: 32, FacilityCode: 32,
IsReadableCard: 1, IssueCode: 32, CommServerHostName:
"server host", EventText: "event text")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateLenelAccessEvent(
  const LenelEventID* aID,
  const wchar_t* aPanel,
  const wchar_t* aDevice,
  const __int32* aSecondaryDevice,
  const __int64* aCardNumber,
  const LenelAccessResult* aAccessResult,
  const LenelEventType* aType,
  const __int32* aSubType,
  const wchar_t* aDescription,
  const __int32* aSerialNumber,
  const TGLibDateTime* aTimeStamp,
  const __int32* aAreaEnteredID,
  const __int32* aAreaExitedID,
  const wchar_t* aAssetID,
  const bool* aCardholderEntered,
  const bool* aDuress,
  const __int32* aElevatorFloor,
  const __int32* aFacilityCode,
  const bool* aIsReadableCard,
  const __int32* aIssueCode,
  const wchar_t* aCommServerHostName,
  const wchar_t* aEventText);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLenelAccessEvent(
  ConstHGscAction anAction,
  const LenelEventID*& aID,
  const wchar_t*& aPanel,
  const wchar_t*& aDevice,
  const __int32*& aSecondaryDevice,
  const __int64*& aCardNumber,
  const LenelAccessResult*& aAccessResult,
  const LenelEventType*& aType,
  const __int32*& aSubType,
  const wchar_t*& aDescription,
  const __int32*& aSerialNumber,
  const TGLibDateTime*& aTimeStamp,
  const __int32*& aAreaEnteredID,
  const __int32*& aAreaExitedID,
  const wchar_t*& aAssetID,
  const bool*& aCardholderEntered,
  const bool*& aDuress,
  const __int32*& aElevatorFloor,
  const __int32*& aFacilityCode,
  const bool*& aIsReadableCard,
  const __int32*& aIssueCode,
  const wchar_t*& aCommServerHostName,
  const wchar_t*& aEventText);
```

Delphi – Create

```
function GscAct_CreateLenelAccessEvent(
  aID: PLenelEventID;
  aPanel: PWideChar;
  aDevice: PWideChar;
  aSecondaryDevice: PInteger;
  aCardNumber: PInt64;
  aAccessResult: PLenelAccessResult;
  aType: PLenelEventType;
  aSubType: PInteger;
  aDescription: PWideChar;
  aSerialNumber: PInteger;
  aTimeStamp: PTGLibDateTime;
  aAreaEnteredID: PInteger;
  aAreaExitedID: PInteger;
  aAssetID: PWideChar;
  aCardholderEntered: PBoolean;
  aDuress: PBoolean;
  aElevatorFloor: PInteger;
  aFacilityCode: PInteger;
  aIsReadableCard: PBoolean;
  aIssueCode: PInteger;
  aCommServerHostName: PWideChar;
  aEventText: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeLenelAccessEvent(
  const anAction: HGscAction;
  out aID: PLenelEventID;
  out aPanel: PWideChar;
  out aDevice: PWideChar;
  out aSecondaryDevice: PInteger;
  out aCardNumber: PInt64;
  out aAccessResult: PLenelAccessResult;
  out aType: PLenelEventType;
  out aSubType: PInteger;
  out aDescription: PWideChar;
  out aSerialNumber: PInteger;
  out aTimeStamp: PTGLibDateTime;
  out aAreaEnteredID: PInteger;
  out aAreaExitedID: PInteger;
  out aAssetID: PWideChar;
  out aCardholderEntered: PBoolean;
  out aDuress: PBoolean;
  out aElevatorFloor: PInteger;
  out aFacilityCode: PInteger;
  out aIsReadableCard: PBoolean;
  out aIssueCode: PInteger;
  out aCommServerHostName: PWideChar;
  out aEventText: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

13.2 Lenel fire event

LenelFireEvent (*ID*, *Panel*, *Device*, *SecondaryDevice*, *TroubleCode*, *Type*, *SubType*, Description, *SerialNumber*, *TimeStamp*, *CommServerHostName*, *EventText*)

Description : *Lenel OnGuard fire event.*

Code : ac_LenelFireEvent (510)

Class : ak_Lenel (16)

Parameters :

ID (ID) [optional] :
Type : LenelEventID
Description : The ID that uniquely identifies the type of this event.

Panel (panel) [optional] :
Type : widestring
Description : The name of the panel where this event originated.

Device (device) [optional] :
Type : widestring
Description : The name of the device where this event originated.

SecondaryDevice (secondary device) [optional] :
Type : __int32
Description : The ID of the secondary device where this event originated.

TroubleCode (trouble code) [optional] :
Type : _int32
Description : A trouble code associated with the fire event.

Type (type) [optional] :
Type : LenelEventType
Description : Event type i.e., duress, system, etc.

SubType (subtype) [optional] :
Type : _int32
Description : Event sub-type i.e., granted, door forced open, etc.

Description (description) :
Type : widestring
Description : A human readable, brief description of this event.

SerialNumber (serial number) [optional] :
Type : _int32
Description : A number that uniquely identifies the instance of the event for a particular panel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

CommServerHostName (server host) [optional] :
Type : widestring
Description : Host name of the Communication server through which the event arrived.

EventText (event text) [optional] :
Type : widestring
Description : Text associated with event

Text

```
LenelFireEvent (ID: 0, Panel: "panel", Device:
"device", SecondaryDevice: 32, TroubleCode: 32,
Type: 0, SubType: 32, "description", SerialNumber:
32, TimeStamp: "2013/09/05 14:59:59,999 GMT+02:00",
CommServerHostName: "server host", EventText: "event
text")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLenelFireEvent(
  const LenelEventID* aID,
  const wchar_t* aPanel,
  const wchar_t* aDevice,
  const __int32* aSecondaryDevice,
  const __int32* aTroubleCode,
  const LenelEventType* aType,
  const __int32* aSubType,
  const wchar_t* aDescription,
  const __int32* aSerialNumber,
  const TGLibDateTime* aTimeStamp,
  const wchar_t* aCommServerHostName,
  const wchar_t* aEventText);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLenelFireEvent(
  ConstHGscAction anAction,
  const LenelEventID*& aID,
  const wchar_t*& aPanel,
  const wchar_t*& aDevice,
  const __int32*& aSecondaryDevice,
  const __int32*& aTroubleCode,
  const LenelEventType*& aType,
  const __int32*& aSubType,
  const wchar_t*& aDescription,
  const __int32*& aSerialNumber,
  const TGLibDateTime*& aTimeStamp,
  const wchar_t*& aCommServerHostName,
  const wchar_t*& aEventText);
```

Delphi – Create

```
function GscAct_CreateLenelFireEvent(
  aID: PLenelEventID;
  aPanel: PWideChar;
  aDevice: PWideChar;
  aSecondaryDevice: PInteger;
  aTroubleCode: PInteger;
  aType: PLenelEventType;
  aSubType: PInteger;
  aDescription: PWideChar;
  aSerialNumber: PInteger;
  aTimeStamp: PTGLibDateTime;
  aCommServerHostName: PWideChar;
  aEventText: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLenelFireEvent(
  const anAction: HGscAction;
  out aID: PLenelEventID;
  out aPanel: PWideChar;
  out aDevice: PWideChar;
  out aSecondaryDevice: PInteger;
  out aTroubleCode: PInteger;
  out aType: PLenelEventType;
  out aSubType: PInteger;
  out aDescription: PWideChar;
  out aSerialNumber: PInteger;
  out aTimeStamp: PTGLibDateTime;
  out aCommServerHostName: PWideChar;
  out aEventText: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

13.3 Lenel intercom event

LenelIntercomEvent (*ID*, *Panel*, *Device*, *SecondaryDevice*, *IntercomData*, *LineNumber*, *Type*, *SubType*, Description, *SerialNumber*, *TimeStamp*, *CommServerHostName*, *EventText*)

Description : *Lenel OnGuard intercom event.*

Code : `ac_LenelIntercomEvent` (506)

Class : `ak_Lenel` (16)

Parameters :

ID (ID) [optional] :
Type : LenelEventID
Description : The ID that uniquely identifies the type of this event.

Panel (panel) [optional] :
Type : widestring
Description : The name of the panel where this event originated.

Device (device) [optional] :
Type : widestring
Description : The name of the device where this event originated.

SecondaryDevice (secondary device) [optional] :
Type : __int32
Description : The ID of the secondary device where this event originated.

IntercomData (intercom data) [optional] :
Type : __int32
Description : Additional data for the intercom event that occurred.

LineNumber (line number) [optional] :
Type : __int32
Description : The line number involved in the intercom event.

Type (type) [optional] :
Type : LenelEventType
Description : Event type i.e., duress, system, etc.

SubType (subtype) [optional] :
Type : __int32
Description : Event sub-type i.e., granted, door forced open, etc.

Description (description) :
Type : widestring
Description : A human readable, brief description of this event.

SerialNumber (serial number) [optional] :
Type : __int32
Description : A number that uniquely identifies the instance of the event for a particular panel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

CommServerHostName (server host) [optional] :
Type : widestring
Description : Host name of the Communication server through which the event arrived.

EventText (event text) [optional] :
Type : widestring
Description : Text associated with event

Text

```
LenelIntercomEvent (ID: 0, Panel: "panel", Device:
"device", SecondaryDevice: 32, IntercomData:
32, LineNumber: 32, Type: 0, SubType: 32,
"description", SerialNumber: 32, TimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00", CommServerHostName:
"server host", EventText: "event text")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateLenelIntercomEvent(
  const LenelEventID* aID,
  const wchar_t* aPanel,
  const wchar_t* aDevice,
  const __int32* aSecondaryDevice,
  const __int32* aIntercomData,
  const __int32* aLineNumber,
  const LenelEventType* aType,
  const __int32* aSubType,
  const wchar_t* aDescription,
  const __int32* aSerialNumber,
  const TGLibDateTime* aTimeStamp,
  const wchar_t* aCommServerHostName,
  const wchar_t* aEventText);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLenelIntercomEvent(
  ConstHGscAction anAction,
  const LenelEventID*& aID,
  const wchar_t*& aPanel,
  const wchar_t*& aDevice,
  const __int32*& aSecondaryDevice,
  const __int32*& aIntercomData,
  const __int32*& aLineNumber,
  const LenelEventType*& aType,
  const __int32*& aSubType,
  const wchar_t*& aDescription,
  const __int32*& aSerialNumber,
  const TGLibDateTime*& aTimeStamp,
  const wchar_t*& aCommServerHostName,
  const wchar_t*& aEventText);
```

Delphi – Create

```
function GscAct_CreateLenelIntercomEvent(
  aID: PLenelEventID;
  aPanel: PWideChar;
  aDevice: PWideChar;
  aSecondaryDevice: PInteger;
  aIntercomData: PInteger;
  aLineNumber: PInteger;
  aType: PLenelEventType;
  aSubType: PInteger;
  aDescription: PWideChar;
  aSerialNumber: PInteger;
  aTimeStamp: PTGLibDateTime;
  aCommServerHostName: PWideChar;
  aEventText: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLenelIntercomEvent(
  const anAction: HGscAction;
  out aID: PLenelEventID;
  out aPanel: PWideChar;
  out aDevice: PWideChar;
  out aSecondaryDevice: PInteger;
  out aIntercomData: PInteger;
  out aLineNumber: PInteger;
  out aType: PLenelEventType;
  out aSubType: PInteger;
  out aDescription: PWideChar;
  out aSerialNumber: PInteger;
  out aTimeStamp: PTGLibDateTime;
  out aCommServerHostName: PWideChar;
  out aEventText: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

13.4 Lenel raw data

LenelRawData (TimeStamp, LenelData)

Description : *Lenel OnGuard raw data.*

Code : ac_LenelRawData (512)

Class : ak_Lenel (16)

Parameters :

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

LenelData (data) :
Type : widestring
Description : Lenel OnGuard data.

Text

```
LenelRawData ("2013/09/05 14:59:59,999 GMT+02:00",
"data")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLenelRawData(
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aLenelData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLenelRawData(
  ConstHGscAction anAction,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aLenelData);
```

Delphi – Create

```
function GscAct_CreateLenelRawData(
  var aTimeStamp: TGLibDateTime;
  aLenelData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLenelRawData(
  const anAction: HGscAction;
  out aTimeStamp: TGLibDateTime;
  out aLenelData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

13.5 Lenel refresh names

LenelRefreshNames ()

Description : *Lenel OnGuard refresh names.*

Code : ac_LenelRefreshNames (514)

Class : ak_Lenel (16)

This action has no parameters.

Text

```
LenelRefreshNames ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLenelRefreshNames();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLenelRefreshNames(
  ConstHGscAction anAction);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateLenelRefreshNames()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLenelRefreshNames(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

13.6 Lenel security event

LenelSecurityEvent (*ID*, *Panel*, *Device*, *SecondaryDevice*, *Type*, *SubType*, Description, *SerialNumber*, *TimeStamp*, *CommServerHostName*, *EventText*)

Description : *Lenel OnGuard security event.*

Code : ac_LenelSecurityEvent (500)

Class : ak_Lenel (16)

Parameters :

ID (ID) [optional] :
Type : LenelEventID
Description : The ID that uniquely identifies the type of this event.

Panel (panel) [optional] :
Type : widestring
Description : The name of the panel where this event originated.

Device (device) [optional] :
Type : widestring
Description : The name of the device where this event originated.

SecondaryDevice (secondary device) [optional] :
Type : __int32
Description : The ID of the secondary device where this event originated.

Type (type) [optional] :
Type : LenelEventType
Description : Event type i.e., duress, system, etc.

SubType (subtype) [optional] :
Type : __int32
Description : Event sub-type i.e., granted, door forced open, etc.

Description (description) :
Type : widestring
Description : A human readable, brief description of this event.

SerialNumber (serial number) [optional] :
Type : __int32
Description : A number that uniquely identifies the instance of the event for a particular panel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

CommServerHostName (server host) [optional] :
Type : widestring
Description : Host name of the Communication server through which the event arrived.

EventText (event text) [optional] :
Type : widestring
Description : Text associated with event

Text

```
LenelSecurityEvent (ID: 0, Panel: "panel", Device:
"device", SecondaryDevice: 32, Type: 0, SubType:
32, "description", SerialNumber: 32, TimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00", CommServerHostName:
"server host", EventText: "event text")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateLenelSecurityEvent(
  const LenelEventID* aID,
  const wchar_t* aPanel,
  const wchar_t* aDevice,
  const __int32* aSecondaryDevice,
  const LenelEventType* aType,
  const __int32* aSubType,
  const wchar_t* aDescription,
  const __int32* aSerialNumber,
  const TGLibDateTime* aTimeStamp,
  const wchar_t* aCommServerHostName,
  const wchar_t* aEventText);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLenelSecurityEvent(
  ConstHGscAction anAction,
  const LenelEventID*& aID,
  const wchar_t*& aPanel,
  const wchar_t*& aDevice,
  const __int32*& aSecondaryDevice,
  const LenelEventType*& aType,
  const __int32*& aSubType,
  const wchar_t*& aDescription,
  const __int32*& aSerialNumber,
  const TGLibDateTime*& aTimeStamp,
  const wchar_t*& aCommServerHostName,
  const wchar_t*& aEventText);
```

Delphi – Create

```
function GscAct_CreateLenelSecurityEvent(
  aID: PLenelEventID;
  aPanel: PWideChar;
  aDevice: PWideChar;
  aSecondaryDevice: PInteger;
  aType: PLenelEventType;
  aSubType: PInteger;
  aDescription: PWideChar;
  aSerialNumber: PInteger;
  aTimeStamp: PTGLibDateTime;
  aCommServerHostName: PWideChar;
  aEventText: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLenelSecurityEvent(
  const anAction: HGscAction;
  out aID: PLenelEventID;
  out aPanel: PWideChar;
  out aDevice: PWideChar;
  out aSecondaryDevice: PInteger;
  out aType: PLenelEventType;
  out aSubType: PInteger;
  out aDescription: PWideChar;
  out aSerialNumber: PInteger;
  out aTimeStamp: PTGLibDateTime;
  out aCommServerHostName: PWideChar;
  out aEventText: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

13.7 Lenel video event

LenelVideoEvent (*ID*, *Panel*, *Device*, *SecondaryDevice*, *Channel*, *Type*, *SubType*, Description, *SerialNumber*, *TimeStamp*, *StartTime*, *EndTime*, *CommServerHostName*, *EventText*)

Description : *Lenel OnGuard video event.*

Code : `ac_LenelVideoEvent` (508)

Class : `ak_Lenel` (16)

Parameters :

ID (ID) [optional] :
Type : LenelEventID
Description : The ID that uniquely identifies the type of this event.

Panel (panel) [optional] :
Type : widestring
Description : The name of the panel where this event originated.

Device (device) [optional] :
Type : widestring
Description : The name of the device where this event originated.

SecondaryDevice (secondary device) [optional] :
Type : __int32
Description : The ID of the secondary device where this event originated.

Channel (channel) [optional] :
Type : __int32
Description : The physical channel the camera is connected to that is creating this event.

Type (type) [optional] :
Type : LenelEventType
Description : Event type i.e., duress, system, etc.

SubType (subtype) [optional] :
Type : __int32
Description : Event sub-type i.e., granted, door forced open, etc.

Description (description) :
Type : widestring
Description : A human readable, brief description of this event.

SerialNumber (serial number) [optional] :
Type : __int32
Description : A number that uniquely identifies the instance of the event for a particular panel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

StartTime (start stamp) [optional] :
Type : TGLibDateTime
Description : The time the video event started

EndTime (end time) [optional] :
Type : TGLibDateTime
Description : The time the video event ended.

CommServerHostName (server host) [optional] :
Type : widestring
Description : Host name of the Communication server through which the event arrived.

EventText (event text) [optional] :
Type : widestring
Description : Text associated with event

Text

```
LenelVideoEvent (ID: 0, Panel: "panel", Device:
"device", SecondaryDevice: 32, Channel: 32, Type:
0, SubType: 32, "description", SerialNumber: 32,
TimeStamp: "2013/09/05 14:59:59,999 GMT+02:00",
StartTime: "2013/09/05 14:59:59,999 GMT+02:00", EndTime:
"2013/09/05 14:59:59,999 GMT+02:00", CommServerHostName:
"server host", EventText: "event text")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateLenelVideoEvent(
  const LenelEventID* aID,
  const wchar_t* aPanel,
  const wchar_t* aDevice,
  const __int32* aSecondaryDevice,
  const __int32* aChannel,
  const LenelEventType* aType,
  const __int32* aSubType,
  const wchar_t* aDescription,
  const __int32* aSerialNumber,
  const TGLibDateTime* aTimeStamp,
  const TGLibDateTime* aStartTime,
  const TGLibDateTime* aEndTime,
  const wchar_t* aCommServerHostName,
  const wchar_t* aEventText);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLenelVideoEvent(
  ConstHGscAction anAction,
  const LenelEventID*& aID,
  const wchar_t*& aPanel,
  const wchar_t*& aDevice,
  const __int32*& aSecondaryDevice,
  const __int32*& aChannel,
  const LenelEventType*& aType,
  const __int32*& aSubType,
  const wchar_t*& aDescription,
  const __int32*& aSerialNumber,
  const TGLibDateTime*& aTimeStamp,
  const TGLibDateTime*& aStartTime,
  const TGLibDateTime*& aEndTime,
  const wchar_t*& aCommServerHostName,
  const wchar_t*& aEventText);
```

Delphi – Create

```
function GscAct_CreateLenelVideoEvent(
  aID: PLenelEventID;
  aPanel: PWideChar;
  aDevice: PWideChar;
  aSecondaryDevice: PInteger;
  aChannel: PInteger;
  aType: PLenelEventType;
  aSubType: PInteger;
  aDescription: PWideChar;
  aSerialNumber: PInteger;
  aTimeStamp: PTGLibDateTime;
  aStartTime: PTGLibDateTime;
  aEndTime: PTGLibDateTime;
  aCommServerHostName: PWideChar;
  aEventText: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeLenelVideoEvent(
  const anAction: HGscAction;
  out aID: PLenelEventID;
  out aPanel: PWideChar;
  out aDevice: PWideChar;
  out aSecondaryDevice: PInteger;
  out aChannel: PInteger;
  out aType: PLenelEventType;
  out aSubType: PInteger;
  out aDescription: PWideChar;
  out aSerialNumber: PInteger;
  out aTimeStamp: PTGLibDateTime;
  out aStartTime: PTGLibDateTime;
  out aEndTime: PTGLibDateTime;
  out aCommServerHostName: PWideChar;
  out aEventText: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14 POS

POS (point of sale).

14.1 Barcode data

BarcodeData (ReaderName, TimeStamp, Barcode)

Description : *Barcode data.*

Code : ac_BarcodeData (365)

Class : ak_POS (11)

Parameters :

ReaderName (scanner) :
Type : widestring
Description : Scanner name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Barcode (code) :
Type : widestring
Description : Barcode.

Text

```
BarcodeData ("scanner", "2013/09/05 14:59:59,999 GMT+02:00",
"code")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateBarcodeData(
  const wchar_t* aReaderName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aBarcode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeBarcodeData(
  ConstHGscAction anAction,
  const wchar_t*& aReaderName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aBarcode);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateBarcodeData(
  aReaderName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aBarcode: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeBarcodeData(
  const anAction: HGscAction;
  out aReaderName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aBarcode: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14.2 Filling pump status

FillingPumpStatus (TerminalName, TimeStamp, PumpNo, Status, *Amount*, *Price*, *Details*)

Description : *Filling pump status.*

Code : ac_FillingPumpStatus (366)

Class : ak_POS (11)

Parameters :

TerminalName (terminal) :
Type : widestring
Description : Terminal name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

PumpNo (pump no) :
Type : __int32
Description : Pump no.

Status (status) :
Type : PlcPumpStatus
Description : Status.

Amount (amount) [optional] :
Type : double
Description : Amount.

Price (price) [optional] :
Type : double
Description : Price.

Details (details) [optional] :
Type : widestring
Description : Details.

Text

```
FillingPumpStatus ("terminal", "2013/09/05 14:59:59,999 GMT+02:00",
32, 0, Amount:  0.0, Price:  0.0, Details:  "details")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateFillingPumpStatus(
  const wchar_t* aTerminalName,
  const TGLibDateTime& aTimeStamp,
  const __int32 aPumpNo,
  const PlcPumpStatus aStatus,
  const double* aAmount,
  const double* aPrice,
  const wchar_t* aDetails);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeFillingPumpStatus(
  ConstHGscAction anAction,
  const wchar_t*& aTerminalName,
  TGLibDateTime& aTimeStamp,
  __int32& aPumpNo,
  PlcPumpStatus& aStatus,
  const double*& aAmount,
  const double*& aPrice,
  const wchar_t*& aDetails);
```

Delphi – Create

```
function GscAct_CreateFillingPumpStatus(
  aTerminalName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aPumpNo: Integer;
  aStatus: PlcPumpStatus;
  aAmount: PDouble;
  aPrice: PDouble;
  aDetails: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeFillingPumpStatus(
  const anAction: HGscAction;
  out aTerminalName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aPumpNo: Integer;
  out aStatus: PlcPumpStatus;
  out aAmount: PDouble;
  out aPrice: PDouble;
  out aDetails: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14.3 Interface raw answer

InterfaceRawAnswer (InterfaceName, TimeStamp, Data)

Description : *Interface raw answer.*

Code : ac_InterfaceRawAnswer (359)

Class : ak_POS (11)

Parameters :

InterfaceName (interface) :
Type : widestring
Description : Interface name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Data (answer) :
Type : widestring
Description : Interface answer.

Text

```
InterfaceRawAnswer ("interface", "2013/09/05 14:59:59,999 GMT+02:00",
"answer")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateInterfaceRawAnswer(
  const wchar_t* aInterfaceName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeInterfaceRawAnswer(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aData);
```

Delphi – Create

```
function GscAct_CreateInterfaceRawAnswer(
  aInterfaceName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeInterfaceRawAnswer(
  const anAction: HGscAction;
  out aInterfaceName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14.4 Interface raw data

InterfaceRawData (InterfaceName, TimeStamp, Data)

Description : *Interface raw data.*

Code : ac_InterfaceRawData (358)

Class : ak_POS (11)

Parameters :

InterfaceName (interface) :
Type : widestring
Description : Interface name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Data (data) :
Type : widestring
Description : Interface data.

GEUTEBRÜCK

Text

```
InterfaceRawData ("interface", "2013/09/05 14:59:59,999 GMT+02:00",
"data")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateInterfaceRawData(
  const wchar_t* aInterfaceName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeInterfaceRawData(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aData);
```

Delphi – Create

```
function GscAct_CreateInterfaceRawData(
  aInterfaceName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aData: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeInterfaceRawData(
  const anAction: HGscAction;
  out aInterfaceName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aData: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14.5 POS data

POSData (POSName, TimeStamp, *Article*, *Price*, *Units*, *PricePerUnit*, *Line1*, *Line2*, *Line3*, *Line4*, *Line5*)

Description : *POS data.*

Code : ac_POSData (361)

Class : ak_POS (11)

Parameters :

POSName (POS) :
Type : widestring
Description : POS name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Article (article) [optional] :
Type : widestring
Description : Article description.

Price (price) [optional] :
Type : double
Description : Price.

Units (units) [optional] :
Type : __int32
Description : Number of units.

PricePerUnit (price per unit) [optional] :
Type : double
Description : Price per unit.

Line1 (line 1) [optional] :
Type : widestring
Description : Line 1.

Line2 (line 2) [optional] :
Type : widestring
Description : Line 2.

Line3 (line 3) [optional] :
Type : widestring
Description : Line 3.

Line4 (line 4) [optional] :
Type : widestring
Description : Line 4.

Line5 (line 5) [optional] :
Type : widestring
Description : Line 5.

Text

```
POSData ("POS", "2013/09/05 14:59:59,999 GMT+02:00",
Article: "article", Price: 0.0, Units: 32,
PricePerUnit: 0.0, Line1: "line 1", Line2: "line 2",
Line3: "line 3", Line4: "line 4", Line5: "line 5")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePOSData(
  const wchar_t* aPOSName,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aArticle,
  const double* aPrice,
  const __int32* aUnits,
  const double* aPricePerUnit,
  const wchar_t* aLine1,
  const wchar_t* aLine2,
  const wchar_t* aLine3,
  const wchar_t* aLine4,
  const wchar_t* aLine5);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePOSData(
  ConstHGscAction anAction,
  const wchar_t*& aPOSName,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aArticle,
  const double*& aPrice,
  const __int32*& aUnits,
  const double*& aPricePerUnit,
  const wchar_t*& aLine1,
  const wchar_t*& aLine2,
  const wchar_t*& aLine3,
  const wchar_t*& aLine4,
  const wchar_t*& aLine5);
```

Delphi – Create

```
function GscAct_CreatePOSData(
  aPOSName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aArticle: PWideChar;
  aPrice: PDouble;
  aUnits: PInteger;
  aPricePerUnit: PDouble;
  aLine1: PWideChar;
  aLine2: PWideChar;
  aLine3: PWideChar;
  aLine4: PWideChar;
  aLine5: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePOSData(
  const anAction: HGscAction;
  out aPOSName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aArticle: PWideChar;
  out aPrice: PDouble;
  out aUnits: PInteger;
  out aPricePerUnit: PDouble;
  out aLine1: PWideChar;
  out aLine2: PWideChar;
  out aLine3: PWideChar;
  out aLine4: PWideChar;
  out aLine5: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14.6 POS status

POSStatus (POSName, TimeStamp, Status, *Details*)

Description : *POS status.*

Code : ac_POSStatus (360)

Class : ak_POS (11)

Parameters :

POSName (POS) :
Type : widestring
Description : POS name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Status (status) :
Type : PlcPOSStatus
Description : Status.

Details (details) [optional] :
Type : widestring
Description : Details.

Text

```
POSStatus ("POS", "2013/09/05 14:59:59,999 GMT+02:00", 0,
Details:  "details")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePOSStatus(
  const wchar_t* aPOSName,
  const TGLibDateTime& aTimeStamp,
  const PlcPOSStatus aStatus,
  const wchar_t* aDetails);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePOSStatus(
  ConstHGscAction anAction,
  const wchar_t*& aPOSName,
  TGLibDateTime& aTimeStamp,
  PlcPOSStatus& aStatus,
  const wchar_t*& aDetails);
```

Delphi – Create

```
function GscAct_CreatePOSStatus(
  aPOSName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aStatus: PlcPOSStatus;
  aDetails: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePOSStatus(
  const anAction: HGscAction;
  out aPOSName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aStatus: PlcPOSStatus;
  out aDetails: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14.7 Terminal article data

TerminalArticleData (TerminalName, TimeStamp, CashierStation, *PumpNo*, AlarmStatus, *Amount*, *Price*, *Details*)

Description : *Terminal article data.*

Code : ac_TerminalArticleData (363)

Class : ak_POS (11)

Parameters :

TerminalName (terminal) :
Type : widestring
Description : Terminal name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

CashierStation (cashier station) :
Type : __int32
Description : Cashier station.

PumpNo (pump no) [optional] :
Type : __int32
Description : Pump no.

AlarmStatus (alarm) :
Type : bool
Description : Alarm status.

Amount (amount) [optional] :
Type : double
Description : Amount.

Price (price) [optional] :
Type : double
Description : Price.

Details (details) [optional] :
Type : widestring
Description : Details.

Text

```
TerminalArticleData ("terminal", "2013/09/05 14:59:59,999 GMT+02:00",
32, PumpNo:  32, 1, Amount:  0.0, Price:  0.0, Details:
"details")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTerminalArticleData(
  const wchar_t* aTerminalName,
  const TGLibDateTime& aTimeStamp,
  const __int32 aCashierStation,
  const __int32* aPumpNo,
  const bool aAlarmStatus,
  const double* aAmount,
  const double* aPrice,
  const wchar_t* aDetails);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTerminalArticleData(
  ConstHGscAction anAction,
  const wchar_t*& aTerminalName,
  TGLibDateTime& aTimeStamp,
  __int32& aCashierStation,
  const __int32*& aPumpNo,
  bool& aAlarmStatus,
  const double*& aAmount,
  const double*& aPrice,
  const wchar_t*& aDetails);
```

Delphi – Create

```
function GscAct_CreateTerminalArticleData(
  aTerminalName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aCashierStation: Integer;
  aPumpNo: PInteger;
  aAlarmStatus: Boolean;
  aAmount: PDouble;
  aPrice: PDouble;
  aDetails: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeTerminalArticleData(
  const anAction: HGscAction;
  out aTerminalName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aCashierStation: Integer;
  out aPumpNo: PInteger;
  out aAlarmStatus: Boolean;
  out aAmount: PDouble;
  out aPrice: PDouble;
  out aDetails: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

14.8 Terminal payment data

TerminalPaymentData (TerminalName, TimeStamp, CashierStation, *PumpNo*, AlarmStatus, *Amount*, *Price*, *Details*)

Description : *Terminal payment data.*

Code : ac_TerminalPaymentData (364)

Class : ak_POS (11)

Parameters :

TerminalName (terminal) :
Type : widestring
Description : Terminal name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

CashierStation (cashier station) :
Type : __int32
Description : Cashier station.

PumpNo (pump no) [optional] :
Type : __int32
Description : Pump no.

AlarmStatus (alarm) :
Type : bool
Description : Alarm status.

Amount (amount) [optional] :
Type : double
Description : Amount.

Price (price) [optional] :
Type : double
Description : Price.

Details (details) [optional] :
Type : widestring
Description : Details.

Text

```
TerminalPaymentData ("terminal", "2013/09/05 14:59:59,999 GMT+02:00",
32, PumpNo:  32, 1, Amount:  0.0, Price:  0.0, Details:
"details")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTerminalPaymentData(
  const wchar_t* aTerminalName,
  const TGLibDateTime& aTimeStamp,
  const __int32 aCashierStation,
  const __int32* aPumpNo,
  const bool aAlarmStatus,
  const double* aAmount,
  const double* aPrice,
  const wchar_t* aDetails);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTerminalPaymentData(
  ConstHGscAction anAction,
  const wchar_t*& aTerminalName,
  TGLibDateTime& aTimeStamp,
  __int32& aCashierStation,
  const __int32*& aPumpNo,
  bool& aAlarmStatus,
  const double*& aAmount,
  const double*& aPrice,
  const wchar_t*& aDetails);
```

Delphi – Create

```
function GscAct_CreateTerminalPaymentData(
  aTerminalName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aCashierStation: Integer;
  aPumpNo: PInteger;
  aAlarmStatus: Boolean;
  aAmount: PDouble;
  aPrice: PDouble;
  aDetails: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeTerminalPaymentData(
  const anAction: HGscAction;
  out aTerminalName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aCashierStation: Integer;
  out aPumpNo: PInteger;
  out aAlarmStatus: Boolean;
  out aAmount: PDouble;
  out aPrice: PDouble;
  out aDetails: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

15 Perimeter protection

Perimeter protection.

15.1 PP device alarm

```
PPDeviceAlarm (InterfaceID, DeviceAddress, Sensor, State)
```

Description : *Perimeter protection device alarm.*

Code : `ac_PPDeviceAlarm` (581)

Class : `ak_PP` (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

DeviceAddress (device address) :
Type : __int32
Description : Device address.

Sensor (sensor) :
Type : PPSensorKind
Description : Sensor.

State (state) :
Type : PPAlarmState
Description : State.

Text

```
PPDeviceAlarm ("interface id", 32, 0, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPDeviceAlarm(
  const wchar_t* aInterfaceID,
  const __int32 aDeviceAddress,
  const PPSensorKind aSensor,
  const PPAlarmState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPDeviceAlarm(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID,
  __int32& aDeviceAddress,
  PPSensorKind& aSensor,
  PPAlarmState& aState);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreatePPDeviceAlarm(
  aInterfaceID: PWideChar;
  aDeviceAddress: Integer;
  aSensor: PPSensorKind;
  aState: PPAlarmState)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPDeviceAlarm(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar;
  out aDeviceAddress: Integer;
  out aSensor: PPSensorKind;
  out aState: PPAlarmState)
  : Boolean; stdcall; external GscActionsDll;
```

15.2 PP device input

PPDeviceInput (InterfaceID, DeviceAddress, Input, State)

Description : *Perimeter protection device input.*

Code : ac_PPDeviceInput (583)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

DeviceAddress (device address) :
Type : __int32
Description : Device address.

Input (input) :
Type : __int32
Description : Input.

State (state) :
Type : PPAlarmState
Description : State.

Text

```
PPDeviceInput ("interface id", 32, 32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPDeviceInput(
  const wchar_t* aInterfaceID,
  const __int32 aDeviceAddress,
  const __int32 aInput,
  const PPAlarmState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPDeviceInput(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID,
  __int32& aDeviceAddress,
  __int32& aInput,
  PPAlarmState& aState);
```

Delphi – Create

```
function GscAct_CreatePPDeviceInput(
  aInterfaceID: PWideChar;
  aDeviceAddress: Integer;
  aInput: Integer;
  aState: PPAlarmState)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPDeviceInput(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar;
  out aDeviceAddress: Integer;
  out aInput: Integer;
  out aState: PPAlarmState)
  : Boolean; stdcall; external GscActionsDll;
```

15.3 PP device offline

PPDeviceOffline (InterfaceID, DeviceAddress)

Description : *Perimeter protection device offline.*

Code : ac_PPDeviceOffline (589)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

DeviceAddress (device address) :
Type : __int32
Description : Device address.

Text

```
PPDeviceOffline ("interface id", 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPDeviceOffline(
  const wchar_t* aInterfaceID,
  const __int32 aDeviceAddress);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPDeviceOffline(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID,
  __int32& aDeviceAddress);
```

Delphi – Create

```
function GscAct_CreatePPDeviceOffline(
  aInterfaceID: PWideChar;
  aDeviceAddress: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPDeviceOffline(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar;
  out aDeviceAddress: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

15.4 PP device online

```
PPDeviceOnline (InterfaceID, DeviceAddress)
```

Description : *Perimeter protection device online.*

Code : ac_PPDeviceOnline (588)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

DeviceAddress (device address) :
Type : __int32
Description : Device address.

Text

```
PPDeviceOnline ("interface id", 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPDeviceOnline(
  const wchar_t* aInterfaceID,
  const __int32 aDeviceAddress);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPDeviceOnline(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID,
  __int32& aDeviceAddress);
```

Delphi – Create

```
function GscAct_CreatePPDeviceOnline(
  aInterfaceID: PWideChar;
  aDeviceAddress: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPDeviceOnline(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar;
  out aDeviceAddress: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

15.5 PP interface offline

PPInterfaceOffline (InterfaceID)

Description : *Perimeter protection interface offline.*

Code : ac_PPInterfaceOffline (587)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

Text

```
PPInterfaceOffline ("interface id")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPInterfaceOffline(
  const wchar_t* aInterfaceID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPInterfaceOffline(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID);
```

Delphi – Create

```
function GscAct_CreatePPInterfaceOffline(
  aInterfaceID: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPInterfaceOffline(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

15.6 PP interface online

PPInterfaceOnline (InterfaceID)

Description : *Perimeter protection interface online.*

Code : ac_PPInterfaceOnline (586)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

Text

```
PPInterfaceOnline ("interface id")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPInterfaceOnline(
  const wchar_t* aInterfaceID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPInterfaceOnline(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID);
```

Delphi – Create

```
function GscAct_CreatePPInterfaceOnline(
  aInterfaceID: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPInterfaceOnline(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

15.7 PP query interface

```
PPQueryInterface (InterfaceID)
```

Description : *Perimeter protection query interface.*

Code : ac_PPQueryInterface (585)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

Text

```
PPQueryInterface ("interface id")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPQueryInterface(
  const wchar_t* aInterfaceID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPQueryInterface(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID);
```

Delphi – Create

```
function GscAct_CreatePPQueryInterface(
  aInterfaceID: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPQueryInterface(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

15.8 PP set device output

PPSetDeviceOutput (InterfaceID, DeviceAddress, Output, State)

Description : *Perimeter protection set device output.*

Code : ac_PPSetDeviceOutput (584)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

DeviceAddress (device address) :
Type : __int32
Description : Device address.

Output (output) :
Type : __int32
Description : Output.

State (state) :
Type : PPAlarmState
Description : State.

Text

```
PPSetDeviceOutput ("interface id", 32, 32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPSetDeviceOutput(
  const wchar_t* aInterfaceID,
  const __int32 aDeviceAddress,
  const __int32 aOutput,
  const PPAlarmState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPSetDeviceOutput(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID,
  __int32& aDeviceAddress,
  __int32& aOutput,
  PPAlarmState& aState);
```

Delphi – Create

```
function GscAct_CreatePPSetDeviceOutput(
  aInterfaceID: PWideChar;
  aDeviceAddress: Integer;
  aOutput: Integer;
  aState: PPAlarmState)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodePPSetDeviceOutput(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar;
  out aDeviceAddress: Integer;
  out aOutput: Integer;
  out aState: PPAlarmState)
  : Boolean; stdcall; external GscActionsDll;
```

15.9 PP subcell alarm

PPSubcellAlarm (InterfaceID, DeviceAddress, Cable, Subcell, State)

Description : *Perimeter protection subcell alarm.*

Code : ac_PPSubcellAlarm (580)

Class : ak_PP (18)

Parameters :

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

DeviceAddress (device address) :
Type : __int32
Description : Device address.

Cable (cable) :
Type : PPCableKind
Description : Cable.

Subcell (subcell) :
Type : __int32
Description : Subcell.

State (state) :
Type : PPAlarmState
Description : State.

Text

```
PPSubcellAlarm ("interface id", 32, 0, 32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPSubcellAlarm(
  const wchar_t* aInterfaceID,
  const __int32 aDeviceAddress,
  const PPCableKind aCable,
  const __int32 aSubcell,
  const PPAlarmState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPSubcellAlarm(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceID,
  __int32& aDeviceAddress,
  PPCableKind& aCable,
  __int32& aSubcell,
  PPAlarmState& aState);
```

Delphi – Create

```
function GscAct_CreatePPSubcellAlarm(
  aInterfaceID: PWideChar;
  aDeviceAddress: Integer;
  aCable: PPCableKind;
  aSubcell: Integer;
  aState: PPAlarmState)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPSubcellAlarm(
  const anAction: HGscAction;
  out aInterfaceID: PWideChar;
  out aDeviceAddress: Integer;
  out aCable: PPCableKind;
  out aSubcell: Integer;
  out aState: PPAlarmState)
  : Boolean; stdcall; external GscActionsDll;
```

15.10 PP zone alarm

PPZoneAlarm (ZoneID, InterfaceID, DeviceAddress, *Cable*, *Subcell*, State)

Description : *Perimeter protection zone alarm.*

Code : ac_PPZoneAlarm (582)

GEUTEBRÜCK

Class : ak_PP (18)

Parameters :

ZoneID (zone id) :
Type : widestring
Description : Zone ID.

InterfaceID (interface id) :
Type : widestring
Description : Interface ID.

DeviceAddress (device address) :
Type : __int32
Description : Device address.

Cable (cable) [optional] :
Type : PPCableKind
Description : Cable.

Subcell (subcell) [optional] :
Type : __int32
Description : Subcell.

State (state) :
Type : PPAlarmState
Description : State.

Text

```
PPZoneAlarm ("zone id", "interface id", 32, Cable:  0,
Subcell:  32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreatePPZoneAlarm(
  const wchar_t* aZoneID,
  const wchar_t* aInterfaceID,
  const __int32 aDeviceAddress,
  const PPCableKind* aCable,
  const __int32* aSubcell,
  const PPAlarmState aState);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodePPZoneAlarm(
  ConstHGscAction anAction,
  const wchar_t*& aZoneID,
  const wchar_t*& aInterfaceID,
  __int32& aDeviceAddress,
  const PPCableKind*& aCable,
  const __int32*& aSubcell,
  PPAlarmState& aState);
```

Delphi – Create

```
function GscAct_CreatePPZoneAlarm(
  aZoneID: PWideChar;
  aInterfaceID: PWideChar;
  aDeviceAddress: Integer;
  aCable: PPPCableKind;
  aSubcell: PInteger;
  aState: PPAlarmState)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodePPZoneAlarm(
  const anAction: HGscAction;
  out aZoneID: PWideChar;
  out aInterfaceID: PWideChar;
  out aDeviceAddress: Integer;
  out aCable: PPPCableKind;
  out aSubcell: PInteger;
  out aState: PPAlarmState)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

16 Remote export

Remote export.

16.1 Cancel export

CancelExport (ExportID, AbortFlag)

Description : *Cancel export.*

Code : ac_CancelExport (456)

Class : ak_RemoteExport (12)

Parameters :

ExportID (export GUID) :
Type : GUID
Description : Export GUID.

AbortFlag (abort flag) :
Type : PlcExportAbort
Description : Abort flag.

Text

```
CancelExport ("00000000-0000-0000-0000-000000000000", 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCancelExport(
  const GUID& aExportID,
  const PlcExportAbort aAbortFlag);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCancelExport(
  ConstHGscAction anAction,
  GUID& aExportID,
  PlcExportAbort& aAbortFlag);
```

Delphi – Create

```
function GscAct_CreateCancelExport(
  var aExportID: TGuid;
  aAbortFlag: PlcExportAbort)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCancelExport(
  const anAction: HGscAction;
  out aExportID: TGuid;
  out aAbortFlag: PlcExportAbort)
  : Boolean; stdcall; external GscActionsDll;
```

16.2 Export finished

ExportFinished (ExportID, Success)

Description : *Export finished.*

Code : ac_ExportFinished (455)

Class : ak_RemoteExport (12)

Parameters :

ExportID (export GUID) :
Type : GUID
Description : Export GUID.

Success (success) :
Type : PlcExportSuccess
Description : Success.

Text

```
ExportFinished ("00000000-0000-0000-0000-000000000000",
0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateExportFinished(
  const GUID& aExportID,
  const PlcExportSuccess aSuccess);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeExportFinished(
  ConstHGscAction anAction,
  GUID& aExportID,
  PlcExportSuccess& aSuccess);
```

Delphi – Create

```
function GscAct_CreateExportFinished(
  var aExportID: TGuid;
  aSuccess: PlcExportSuccess)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeExportFinished(
  const anAction: HGscAction;
  out aExportID: TGuid;
  out aSuccess: PlcExportSuccess)
  : Boolean; stdcall; external GscActionsDll;
```

16.3 Export progress

ExportProgress (ExportID, Progress)

Description : *Export progress.*

Code : ac_ExportProgress (454)

Class : ak_RemoteExport (12)

Parameters :

ExportID (export GUID) :
Type : GUID
Description : Export GUID.

Progress (progress) :
Type : __int32
Description : Progress.

Text

```
ExportProgress ("00000000-0000-0000-0000-000000000000",
32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateExportProgress(
  const GUID& aExportID,
  const __int32 aProgress);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeExportProgress(
  ConstHGscAction anAction,
  GUID& aExportID,
  __int32& aProgress);
```

Delphi – Create

```
function GscAct_CreateExportProgress(
  var aExportID: TGuid;
  aProgress: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeExportProgress(
  const anAction: HGscAction;
  out aExportID: TGuid;
  out aProgress: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

16.4 Initialize remote export

InitializeRemoteExport (Viewer, Device)

Description : *Initialize remote export.*

Code : ac_InitializeRemoteExport (451)

Class : ak_RemoteExport (12)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Device (device GUID) :
Type : GUID
Description : Pilot device GUID.

Text

```
InitializeRemoteExport (32, "00000000-0000-0000-0000-000000000000")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateInitializeRemoteExport(
  const TPlcMediaChannelID& aViewer,
  const GUID& aDevice);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeInitializeRemoteExport(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  GUID& aDevice);
```

Delphi – Create

```
function GscAct_CreateInitializeRemoteExport(
  var aViewer: TPlcMediaChannelID;
  var aDevice: TGuid)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeInitializeRemoteExport(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aDevice: TGuid)
  : Boolean; stdcall; external GscActionsDll;
```

16.5 Set export marker

SetExportMarker (Viewer, Marker)

Description : *Set export marker.*

Code : ac_SetExportMarker (450)

Class : ak_RemoteExport (12)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Marker (marker) :
Type : PlcExportMarker
Description : Marker kind.

Text

```
SetExportMarker (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetExportMarker(
  const TPlcMediaChannelID& aViewer,
  const PlcExportMarker aMarker);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetExportMarker(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  PlcExportMarker& aMarker);
```

Delphi – Create

```
function GscAct_CreateSetExportMarker(
  var aViewer: TPlcMediaChannelID;
  aMarker: PlcExportMarker)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetExportMarker(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aMarker: PlcExportMarker)
  : Boolean; stdcall; external GscActionsDll;
```

16.6 Start remote export

StartRemoteExport (ExportID, Device, BackupFormat, Channel, SelectionBegin, SelectionEnd, JobID)

Description : *Start remote export.*

Code : ac_StartRemoteExport (452)

Class : ak_RemoteExport (12)

Parameters :

ExportID (export GUID) :
Type : GUID
Description : Export GUID.

Device (device GUID) :
Type : GUID
Description : Pilot device GUID.

BackupFormat (format) :
Type : PlcBackupFormat
Description : Backup format.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SelectionBegin (start time) :
Type : TGLibDateTime
Description : Start time.

SelectionEnd (end time) :
Type : TGLibDateTime
Description : End time.

JobID (job ID) :
Type : widestring
Description : Job ID.

Text

```
StartRemoteExport ("00000000-0000-0000-0000-000000000000",
"00000000-0000-0000-0000-000000000000", 0,
32, "2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", "job ID")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateStartRemoteExport(
  const GUID& aExportID,
  const GUID& aDevice,
  const PlcBackupFormat aBackupFormat,
  const TPlcMediaChannelID& aChannel,
  const TGLibDateTime& aSelectionBegin,
  const TGLibDateTime& aSelectionEnd,
  const wchar_t* aJobID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeStartRemoteExport(
  ConstHGscAction anAction,
  GUID& aExportID,
  GUID& aDevice,
  PlcBackupFormat& aBackupFormat,
  TPlcMediaChannelID& aChannel,
  TGLibDateTime& aSelectionBegin,
  TGLibDateTime& aSelectionEnd,
  const wchar_t*& aJobID);
```

Delphi – Create

```
function GscAct_CreateStartRemoteExport(
  var aExportID: TGuid;
  var aDevice: TGuid;
  aBackupFormat: PlcBackupFormat;
  var aChannel: TPlcMediaChannelID;
  var aSelectionBegin: TGLibDateTime;
  var aSelectionEnd: TGLibDateTime;
  aJobID: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeStartRemoteExport(
  const anAction: HGscAction;
  out aExportID: TGuid;
  out aDevice: TGuid;
  out aBackupFormat: PlcBackupFormat;
  out aChannel: TPlcMediaChannelID;
  out aSelectionBegin: TGLibDateTime;
  out aSelectionEnd: TGLibDateTime;
  out aJobID: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

16.7 Start scene store

StartSceneStore (SceneStoreID, CutList, PreHistoryLength, RecordingLength)

Description : *Start scene store.*

Code : ac_StartSceneStore (457)

Class : ak_RemoteExport (12)

Parameters :

SceneStoreID (scene store GUID) :
Type : GUID
Description : Scene store GUID.

CutList (cut-list) :
Type : widestring
Description : Cut-list.

PreHistoryLength (pre-history length) :
Type : __int64
Description : Pre-history length.

RecordingLength (recording length) :
Type : __int64
Description : Recording length.

GEUTEBRÜCK

Text

```
StartSceneStore ("00000000-0000-0000-0000-000000000000",
"cut-list", 64, 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateStartSceneStore(
  const GUID& aSceneStoreID,
  const wchar_t* aCutList,
  const __int64& aPreHistoryLength,
  const __int64& aRecordingLength);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeStartSceneStore(
  ConstHGscAction anAction,
  GUID& aSceneStoreID,
  const wchar_t*& aCutList,
  __int64& aPreHistoryLength,
  __int64& aRecordingLength);
```

Delphi – Create

```
function GscAct_CreateStartSceneStore(
  var aSceneStoreID: TGuid;
  aCutList: PWideChar;
  var aPreHistoryLength: Int64;
  var aRecordingLength: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeStartSceneStore(
  const anAction: HGscAction;
  out aSceneStoreID: TGuid;
  out aCutList: PWideChar;
  out aPreHistoryLength: Int64;
  out aRecordingLength: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

17 SKIDATA

SKIDATA messages.

17.1 SKIDATA control

SkidataControl (InterfaceName, Data)

Description : *SKIDATA control information.*

Code : ac_SkidataControl (430)

Class : ak_SkiData (13)

Parameters :

InterfaceName (interface) :
Type : widestring
Description : Interface name.

Data (state) :
Type : PlcSkidataControl
Description : Interface state.

Text

```
SkidataControl ("interface", 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSkidataControl(
  const wchar_t* aInterfaceName,
  const PlcSkidataControl aData);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSkidataControl(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceName,
  PlcSkidataControl& aData);
```

Delphi – Create

```
function GscAct_CreateSkidataControl(
  aInterfaceName: PWideChar;
  aData: PlcSkidataControl)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSkidataControl(
  const anAction: HGscAction;
  out aInterfaceName: PWideChar;
  out aData: PlcSkidataControl)
  : Boolean; stdcall; external GscActionsDll;
```

17.2 SKIDATA device event

SkidataDeviceEvent (InterfaceName, DeviceID, EventCode)

Description : *SKIDATA device event.*

Code : ac_SkidataDeviceEvent (434)

Class : ak_SkiData (13)

Parameters :

InterfaceName (interface) :
Type : widestring
Description : Interface name.

DeviceID (device) :
Type : __int32
Description : Device ID.

EventCode (event code) :
Type : __int32
Description : Event code.

Text

```
SkidataDeviceEvent ("interface", 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSkidataDeviceEvent(
  const wchar_t* aInterfaceName,
  const __int32 aDeviceID,
  const __int32 aEventCode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSkidataDeviceEvent(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceName,
  __int32& aDeviceID,
  __int32& aEventCode);
```

Delphi – Create

```
function GscAct_CreateSkidataDeviceEvent(
  aInterfaceName: PWideChar;
  aDeviceID: Integer;
  aEventCode: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSkidataDeviceEvent(
  const anAction: HGscAction;
  out aInterfaceName: PWideChar;
  out aDeviceID: Integer;
  out aEventCode: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

17.3 SKIDATA entry

SkidataEntry (InterfaceName, MessageCode, TranscactionID, CarParkNo, DeviceID)

Description : *SKIDATA entry.*

Code : ac_SkidataEntry (431)

Class : ak_SkiData (13)

Parameters :

InterfaceName (interface) :
Type : widestring
Description : Interface name.

MessageCode (message) :
Type : PlcSkidataMsgCodeEntry
Description : Message code.

TranscactionID (transaction) :
Type : widestring
Description : Transcaction ID.

CarParkNo (car park) :
Type : __int32
Description : Car park no.

DeviceID (device) :
Type : __int32
Description : Device ID.

Text

```
SkidataEntry ("interface", 0, "transaction", 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSkidataEntry(
  const wchar_t* aInterfaceName,
  const PlcSkidataMsgCodeEntry aMessageCode,
  const wchar_t* aTranscactionID,
  const __int32 aCarParkNo,
  const __int32 aDeviceID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSkidataEntry(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceName,
  PlcSkidataMsgCodeEntry& aMessageCode,
  const wchar_t*& aTranscactionID,
  __int32& aCarParkNo,
  __int32& aDeviceID);
```

Delphi – Create

```
function GscAct_CreateSkidataEntry(
  aInterfaceName: PWideChar;
  aMessageCode: PlcSkidataMsgCodeEntry;
  aTranscactionID: PWideChar;
  aCarParkNo: Integer;
  aDeviceID: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSkidataEntry(
  const anAction: HGscAction;
  out aInterfaceName: PWideChar;
  out aMessageCode: PlcSkidataMsgCodeEntry;
  out aTranscactionID: PWideChar;
  out aCarParkNo: Integer;
  out aDeviceID: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

17.4 SKIDATA exit

SkidataExit (InterfaceName, MessageCode, TranscactionID, CarParkNo, DeviceID)

Description : *SKIDATA exit.*

Code : ac_SkidataExit (432)

Class : ak_SkiData (13)

Parameters :

InterfaceName (interface) :
Type : widestring
Description : Interface name.

MessageCode (message) :
Type : PlcSkidataMsgCodeExit
Description : Message code.

TranscactionID (transaction) :
Type : widestring
Description : Transcaction ID.

CarParkNo (car park) :
Type : __int32
Description : Car park no.

DeviceID (device) :
Type : __int32
Description : Device ID.

Text

```
SkidataExit ("interface", 0, "transaction", 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSkidataExit(
  const wchar_t* aInterfaceName,
  const PlcSkidataMsgCodeExit aMessageCode,
  const wchar_t* aTranscactionID,
  const __int32 aCarParkNo,
  const __int32 aDeviceID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSkidataExit(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceName,
  PlcSkidataMsgCodeExit& aMessageCode,
  const wchar_t*& aTranscactionID,
  __int32& aCarParkNo,
  __int32& aDeviceID);
```

Delphi – Create

```
function GscAct_CreateSkidataExit(
  aInterfaceName: PWideChar;
  aMessageCode: PlcSkidataMsgCodeExit;
  aTranscactionID: PWideChar;
  aCarParkNo: Integer;
  aDeviceID: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSkidataExit(
  const anAction: HGscAction;
  out aInterfaceName: PWideChar;
  out aMessageCode: PlcSkidataMsgCodeExit;
  out aTranscactionID: PWideChar;
  out aCarParkNo: Integer;
  out aDeviceID: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

17.5 SKIDATA transaction

SkidataTransaction (InterfaceName, MessageCode, TranscactionID, CarParkNo, DeviceID)

Description : *SKIDATA transaction.*

Code : ac_SkidataTransaction (433)

Class : ak_SkiData (13)

Parameters :

InterfaceName (interface) :
Type : widestring
Description : Interface name.

MessageCode (message) :
Type : PlcSkidataMsgCodeTransaction
Description : Message code.

TranscactionID (transaction) :
Type : widestring
Description : Transcaction ID.

CarParkNo (car park) :
Type : __int32
Description : Car park no.

DeviceID (device) :
Type : __int32
Description : Device ID.

Text

```
SkidataTransaction ("interface", 0, "transaction", 32,
32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSkidataTransaction(
  const wchar_t* aInterfaceName,
  const PlcSkidataMsgCodeTransaction aMessageCode,
  const wchar_t* aTranscactionID,
  const __int32 aCarParkNo,
  const __int32 aDeviceID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSkidataTransaction(
  ConstHGscAction anAction,
  const wchar_t*& aInterfaceName,
  PlcSkidataMsgCodeTransaction& aMessageCode,
  const wchar_t*& aTranscactionID,
  __int32& aCarParkNo,
  __int32& aDeviceID);
```

Delphi – Create

```
function GscAct_CreateSkidataTransaction(
  aInterfaceName: PWideChar;
  aMessageCode: PlcSkidataMsgCodeTransaction;
  aTranscactionID: PWideChar;
  aCarParkNo: Integer;
  aDeviceID: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSkidataTransaction(
  const anAction: HGscAction;
  out aInterfaceName: PWideChar;
  out aMessageCode: PlcSkidataMsgCodeTransaction;
  out aTranscactionID: PWideChar;
  out aCarParkNo: Integer;
  out aDeviceID: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

18 Supply chain security

Supply chain security.

18.1 GSCS vehicle access denied

GSCSVehicleAccessDenied (SequenceID, PlateNo, TimeStamp, *CompanyCategory*, *CompanyName*, *Reason*, *TrafficLane*, *Channel*, *Direction*)

Description : *GSCS vehicle access denied.*

Code : ac_GSCSVehicleAccessDenied (480)

Class : ak_Logistic (15)

Parameters :

SequenceID (sequence ID) :
Type : __int32
Description : The sequence ID.

PlateNo (plate no.) :
Type : widestring
Description : Recognized plate no.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

CompanyCategory (company category) [optional] :
Type : widestring
Description : Company category.

CompanyName (company name) [optional] :
Type : widestring
Description : Company name.

Reason (reason) [optional] :
Type : widestring
Description : Reason.

TrafficLane (traffic lane) [optional] :
Type : widestring
Description : Traffic lane.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Direction (direction) [optional] :
Type : TrafficDirection
Description : Direction.

Text

```
GSCSVehicleAccessDenied (32, "plate no.",
"2013/09/05 14:59:59,999 GMT+02:00", CompanyCategory:
"company category", CompanyName:  "company name", Reason:
"reason", TrafficLane:  "traffic lane", Channel:  32,
Direction:  0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGSCSVehicleAccessDenied(
  const __int32 aSequenceID,
  const wchar_t* aPlateNo,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aCompanyCategory,
  const wchar_t* aCompanyName,
  const wchar_t* aReason,
  const wchar_t* aTrafficLane,
  const TPlcMediaChannelID* aChannel,
  const TrafficDirection* aDirection);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGSCSVehicleAccessDenied(
  ConstHGscAction anAction,
  __int32& aSequenceID,
  const wchar_t*& aPlateNo,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aCompanyCategory,
  const wchar_t*& aCompanyName,
  const wchar_t*& aReason,
  const wchar_t*& aTrafficLane,
  const TPlcMediaChannelID*& aChannel,
  const TrafficDirection*& aDirection);
```

Delphi – Create

```
function GscAct_CreateGSCSVehicleAccessDenied(
  aSequenceID: Integer;
  aPlateNo: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aCompanyCategory: PWideChar;
  aCompanyName: PWideChar;
  aReason: PWideChar;
  aTrafficLane: PWideChar;
  aChannel: PTPlcMediaChannelID;
  aDirection: PTrafficDirection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGSCSVehicleAccessDenied(
  const anAction: HGscAction;
  out aSequenceID: Integer;
  out aPlateNo: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aCompanyCategory: PWideChar;
  out aCompanyName: PWideChar;
  out aReason: PWideChar;
  out aTrafficLane: PWideChar;
  out aChannel: PTPlcMediaChannelID;
  out aDirection: PTrafficDirection)
  : Boolean; stdcall; external GscActionsDll;
```

18.2 GSCS vehicle access expired

GSCSVehicleAccessExpired (SequenceID, PlateNo, TimeStamp, ExpiringTime, *CompanyCategory*, *CompanyName*, *Reason*, *TrafficLane*, *Channel*, *Direction*)

Description : *GSCS vehicle access expired.*

Code : ac_GSCSVehicleAccessExpired (484)

Class : ak_Logistic (15)

Parameters :

SequenceID (sequence ID) :
Type : __int32
Description : The sequence ID.

PlateNo (plate no.) :
Type : widestring
Description : Recognized plate no.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ExpiringTime (expiring time) :
Type : TGLibDateTime
Description : Expiring time.

CompanyCategory (company category) [optional] :
Type : widestring
Description : Company category.

CompanyName (company name) [optional] :
Type : widestring
Description : Company name.

Reason (reason) [optional] :
Type : widestring
Description : Reason.

TrafficLane (traffic lane) [optional] :
Type : widestring
Description : Traffic lane.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Direction (direction) [optional] :
Type : TrafficDirection
Description : Direction.

Text

```
GSCSVehicleAccessExpired (32, "plate no.",
"2013/09/05 14:59:59,999 GMT+02:00",
"2013/09/05 14:59:59,999 GMT+02:00", CompanyCategory:
"company category", CompanyName:  "company name", Reason:
"reason", TrafficLane:  "traffic lane", Channel:  32,
Direction:  0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGSCSVehicleAccessExpired(
  const __int32 aSequenceID,
  const wchar_t* aPlateNo,
  const TGLibDateTime& aTimeStamp,
  const TGLibDateTime& aExpiringTime,
  const wchar_t* aCompanyCategory,
  const wchar_t* aCompanyName,
  const wchar_t* aReason,
  const wchar_t* aTrafficLane,
  const TPlcMediaChannelID* aChannel,
  const TrafficDirection* aDirection);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGSCSVehicleAccessExpired(
  ConstHGscAction anAction,
  __int32& aSequenceID,
  const wchar_t*& aPlateNo,
  TGLibDateTime& aTimeStamp,
  TGLibDateTime& aExpiringTime,
  const wchar_t*& aCompanyCategory,
  const wchar_t*& aCompanyName,
  const wchar_t*& aReason,
  const wchar_t*& aTrafficLane,
  const TPlcMediaChannelID*& aChannel,
  const TrafficDirection*& aDirection);
```

Delphi – Create

```
function GscAct_CreateGSCSVehicleAccessExpired(
  aSequenceID: Integer;
  aPlateNo: PWideChar;
  var aTimeStamp: TGLibDateTime;
  var aExpiringTime: TGLibDateTime;
  aCompanyCategory: PWideChar;
  aCompanyName: PWideChar;
  aReason: PWideChar;
  aTrafficLane: PWideChar;
  aChannel: PTPlcMediaChannelID;
  aDirection: PTrafficDirection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGSCSVehicleAccessExpired(
  const anAction: HGscAction;
  out aSequenceID: Integer;
  out aPlateNo: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aExpiringTime: TGLibDateTime;
  out aCompanyCategory: PWideChar;
  out aCompanyName: PWideChar;
  out aReason: PWideChar;
  out aTrafficLane: PWideChar;
  out aChannel: PTPlcMediaChannelID;
  out aDirection: PTrafficDirection)
  : Boolean; stdcall; external GscActionsDll;
```

18.3 GSCS vehicle access granted

GSCSVehicleAccessGranted (SequenceID, PlateNo, TimeStamp, *CompanyCategory*, *CompanyName*, *Reason*, *TrafficLane*, *Channel*, *Direction*)

Description : *GSCS vehicle access granted.*

Code : ac_GSCSVehicleAccessGranted (482)

Class : ak_Logistic (15)

Parameters :

SequenceID (sequence ID) :
Type : __int32
Description : The sequence ID.

PlateNo (plate no.) :
Type : widestring
Description : Recognized plate no.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

CompanyCategory (company category) [optional] :
Type : widestring
Description : Company category.

CompanyName (company name) [optional] :
Type : widestring
Description : Company name.

Reason (reason) [optional] :
Type : widestring
Description : Reason.

TrafficLane (traffic lane) [optional] :
Type : widestring
Description : Traffic lane.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Direction (direction) [optional] :
Type : TrafficDirection
Description : Direction.

Text

```
GSCSVehicleAccessGranted (32, "plate no.",
"2013/09/05 14:59:59,999 GMT+02:00", CompanyCategory:
"company category", CompanyName:  "company name", Reason:
"reason", TrafficLane:  "traffic lane", Channel:  32,
Direction:  0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGSCSVehicleAccessGranted(
  const __int32 aSequenceID,
  const wchar_t* aPlateNo,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aCompanyCategory,
  const wchar_t* aCompanyName,
  const wchar_t* aReason,
  const wchar_t* aTrafficLane,
  const TPlcMediaChannelID* aChannel,
  const TrafficDirection* aDirection);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGSCSVehicleAccessGranted(
  ConstHGscAction anAction,
  __int32& aSequenceID,
  const wchar_t*& aPlateNo,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aCompanyCategory,
  const wchar_t*& aCompanyName,
  const wchar_t*& aReason,
  const wchar_t*& aTrafficLane,
  const TPlcMediaChannelID*& aChannel,
  const TrafficDirection*& aDirection);
```

Delphi – Create

```
function GscAct_CreateGSCSVehicleAccessGranted(
  aSequenceID: Integer;
  aPlateNo: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aCompanyCategory: PWideChar;
  aCompanyName: PWideChar;
  aReason: PWideChar;
  aTrafficLane: PWideChar;
  aChannel: PTPlcMediaChannelID;
  aDirection: PTrafficDirection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGSCSVehicleAccessGranted(
  const anAction: HGscAction;
  out aSequenceID: Integer;
  out aPlateNo: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aCompanyCategory: PWideChar;
  out aCompanyName: PWideChar;
  out aReason: PWideChar;
  out aTrafficLane: PWideChar;
  out aChannel: PTPlcMediaChannelID;
  out aDirection: PTrafficDirection)
  : Boolean; stdcall; external GscActionsDll;
```

18.4 GSCS vehicle access pending

GSCSVehicleAccessPending (SequenceID, PlateNo, TimeStamp, *CompanyCategory*, *CompanyName*, *Reason*, *TrafficLane*, *Channel*, *Direction*)

Description : *GSCS vehicle access pending.*

Code : ac_GSCSVehicleAccessPending (486)

Class : ak_Logistic (15)

Parameters :

SequenceID (sequence ID) :
Type : __int32
Description : The sequence ID.

PlateNo (plate no.) :
Type : widestring
Description : Recognized plate no.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

CompanyCategory (company category) [optional] :
Type : widestring
Description : Company category.

CompanyName (company name) [optional] :
Type : widestring
Description : Company name.

Reason (reason) [optional] :
Type : widestring
Description : Reason.

TrafficLane (traffic lane) [optional] :
Type : widestring
Description : Traffic lane.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Direction (direction) [optional] :
Type : TrafficDirection
Description : Direction.

Text

```
GSCSVehicleAccessPending (32, "plate no.",
"2013/09/05 14:59:59,999 GMT+02:00", CompanyCategory:
"company category", CompanyName:  "company name", Reason:
"reason", TrafficLane:  "traffic lane", Channel:  32,
Direction:  0)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGSCSVehicleAccessPending(
  const __int32 aSequenceID,
  const wchar_t* aPlateNo,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aCompanyCategory,
  const wchar_t* aCompanyName,
  const wchar_t* aReason,
  const wchar_t* aTrafficLane,
  const TPlcMediaChannelID* aChannel,
  const TrafficDirection* aDirection);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGSCSVehicleAccessPending(
  ConstHGscAction anAction,
  __int32& aSequenceID,
  const wchar_t*& aPlateNo,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aCompanyCategory,
  const wchar_t*& aCompanyName,
  const wchar_t*& aReason,
  const wchar_t*& aTrafficLane,
  const TPlcMediaChannelID*& aChannel,
  const TrafficDirection*& aDirection);
```

Delphi – Create

```
function GscAct_CreateGSCSVehicleAccessPending(
  aSequenceID: Integer;
  aPlateNo: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aCompanyCategory: PWideChar;
  aCompanyName: PWideChar;
  aReason: PWideChar;
  aTrafficLane: PWideChar;
  aChannel: PTPlcMediaChannelID;
  aDirection: PTrafficDirection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGSCSVehicleAccessPending(
  const anAction: HGscAction;
  out aSequenceID: Integer;
  out aPlateNo: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aCompanyCategory: PWideChar;
  out aCompanyName: PWideChar;
  out aReason: PWideChar;
  out aTrafficLane: PWideChar;
  out aChannel: PTPlcMediaChannelID;
  out aDirection: PTrafficDirection)
  : Boolean; stdcall; external GscActionsDll;
```

18.5 Log NPR recognition

LogNPRRecognition (PlateNo, *Hash*, *Country*, *Channel*, *TimeStamp*, *Restriction*, *Category*)

Description : *Log NPR recognition.*

Code : ac_LogNPRRecognition (469)

Class : ak_Logistic (15)

Parameters :

PlateNo (plate no.) :
Type : widestring
Description : Recognized plate no.

Hash (hash value) [optional] [ForeignKey] :
Type : __int64
Description : Hash value.

Country (country) [optional] :
Type : widestring
Description : Country.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

Restriction (restriction) [optional] :
Type : PlcNPRRestriction
Description : Restriction of recognized number.

Category (category) [optional] :
Type : widestring
Description : Category of recognized number.

GEUTEBRÜCK

Text

```
LogNPRRecognition ("plate no.", Hash:  64,
Country:  "country", Channel:  32, TimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00", Restriction:  0,
Category:  "category")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLogNPRRecognition(
  const wchar_t* aPlateNo,
  const __int64* aHash,
  const wchar_t* aCountry,
  const TPlcMediaChannelID* aChannel,
  const TGLibDateTime* aTimeStamp,
  const PlcNPRRestriction* aRestriction,
  const wchar_t* aCategory);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLogNPRRecognition(
  ConstHGscAction anAction,
  const wchar_t*& aPlateNo,
  const __int64*& aHash,
  const wchar_t*& aCountry,
  const TPlcMediaChannelID*& aChannel,
  const TGLibDateTime*& aTimeStamp,
  const PlcNPRRestriction*& aRestriction,
  const wchar_t*& aCategory);
```

Delphi – Create

```
function GscAct_CreateLogNPRRecognition(
  aPlateNo: PWideChar;
  aHash: PInt64;
  aCountry: PWideChar;
  aChannel: PTPlcMediaChannelID;
  aTimeStamp: PTGLibDateTime;
  aRestriction: PPlcNPRRestriction;
  aCategory: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLogNPRRecognition(
  const anAction: HGscAction;
  out aPlateNo: PWideChar;
  out aHash: PInt64;
  out aCountry: PWideChar;
  out aChannel: PTPlcMediaChannelID;
  out aTimeStamp: PTGLibDateTime;
  out aRestriction: PPlcNPRRestriction;
  out aCategory: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

18.6 Log barcode data

LogBarcodeData (Barcode, *Hash*, *Scanner*, *AreaID*, *AreaName*, *Channel*, *TimeStamp*, *Order*, *Shipment*, *Item*)

Description : *Logistic barcode data.*

Code : ac_LogBarcodeData (465)

Class : ak_Logistic (15)

Parameters :

Barcode (barcode) :
Type : widestring
Description : Barcode.

Hash (hash value) [optional] [ForeignKey] :
Type : __int64
Description : Hash value.

Scanner (scanner name) [optional] :
Type : widestring
Description : Scanner name or IP address.

AreaID (area number) [optional] :
Type : __int64
Description : Area number.

AreaName (area name) [optional] :
Type : widestring
Description : Area name.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

Order (order number) [optional] :
Type : widestring
Description : Order number.

Shipment (shipment number) [optional] :
Type : widestring
Description : Shipment number.

Item (item number) [optional] :
Type : widestring
Description : Item number.

Text

```
LogBarcodeData ("barcode", Hash: 64, Scanner: "scanner
name", AreaID: 64, AreaName: "area name", Channel: 32,
TimeStamp: "2013/09/05 14:59:59,999 GMT+02:00", Order:
"order number", Shipment: "shipment number", Item:
"item number")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateLogBarcodeData(
  const wchar_t* aBarcode,
  const __int64* aHash,
  const wchar_t* aScanner,
  const __int64* aAreaID,
  const wchar_t* aAreaName,
  const TPlcMediaChannelID* aChannel,
  const TGLibDateTime* aTimeStamp,
  const wchar_t* aOrder,
  const wchar_t* aShipment,
  const wchar_t* aItem);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLogBarcodeData(
  ConstHGscAction anAction,
  const wchar_t*& aBarcode,
  const __int64*& aHash,
  const wchar_t*& aScanner,
  const __int64*& aAreaID,
  const wchar_t*& aAreaName,
  const TPlcMediaChannelID*& aChannel,
  const TGLibDateTime*& aTimeStamp,
  const wchar_t*& aOrder,
  const wchar_t*& aShipment,
  const wchar_t*& aItem);
```

Delphi – Create

```
function GscAct_CreateLogBarcodeData(
  aBarcode: PWideChar;
  aHash: PInt64;
  aScanner: PWideChar;
  aAreaID: PInt64;
  aAreaName: PWideChar;
  aChannel: PTPlcMediaChannelID;
  aTimeStamp: PTGLibDateTime;
  aOrder: PWideChar;
  aShipment: PWideChar;
  aItem: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLogBarcodeData(
  const anAction: HGscAction;
  out aBarcode: PWideChar;
  out aHash: PInt64;
  out aScanner: PWideChar;
  out aAreaID: PInt64;
  out aAreaName: PWideChar;
  out aChannel: PTPlcMediaChannelID;
  out aTimeStamp: PTGLibDateTime;
  out aOrder: PWideChar;
  out aShipment: PWideChar;
  out aItem: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

18.7 Log barcode data LPS

LogBarcodeDataLPS (Barcode, *Hash*, *Scanner*, *AreaID*, *AreaName*, *Channel*, *TimeStamp*, *X*, *Y*, *Z*, *LpsTagID*, *LpsStatus*, *LpsCellID*, *LpsAreaID*, *UserParam*, *Order*, *Shipment*, *Item*)

Description : *Logistic barcode and LPS data.*

Code : ac_LogBarcodeDataLPS (467)

Class : ak_Logistic (15)

Parameters :

Barcode (barcode) :
Type : widestring
Description : Barcode.

Hash (hash value) [optional] [ForeignKey] :
Type : __int64
Description : Hash value.

Scanner (scanner name) [optional] :
Type : widestring
Description : Scanner name or IP address.

AreaID (area number) [optional] :
Type : __int64
Description : Area number.

AreaName (area name) [optional] :
Type : widestring
Description : Area name.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) [optional] :
Type : TGLibDateTime
Description : Time stamp.

X (X coordinate) [optional] :
Type : double
Description : X coordinate.

Y (Y coordinate) [optional] :
Type : double
Description : Y coordinate.

Z (Z coordinate) [optional] :
Type : double
Description : Z coordinate.

LpsTagID (LPS tag ID) [optional] :
Type : __int32
Description : LPS tag ID.

LpsStatus (LPS status) [optional] :
Type : PlcLpsStatus
Description : LPS status.

LpsCellID (LPS cell ID) [optional] :
Type : __int32
Description : LPS cell ID.

LpsAreaID (LPS area ID) [optional] :
Type : __int32
Description : LPS area ID.

UserParam (user param) [optional] :
Type : __int32
Description : User param.

Order (order number) [optional] :
Type : widestring
Description : Order number.

Shipment (shipment number) [optional] :
Type : widestring
Description : Shipment number.

Item (item number) [optional] :
Type : widestring
Description : Item number.

Text

```
LogBarcodeDataLPS ("barcode", Hash:  64,
Scanner:  "scanner name", AreaID: 64,
AreaName:  "area name", Channel:  32, TimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00", X: 0.0, Y: 0.0,
Z: 0.0, LpsTagID: 32, LpsStatus:  0, LpsCellID: 32,
LpsAreaID: 32, UserParam:  32, Order:  "order number",
Shipment:  "shipment number", Item:  "item number")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLogBarcodeDataLPS(
  const wchar_t* aBarcode,
  const __int64* aHash,
  const wchar_t* aScanner,
  const __int64* aAreaID,
  const wchar_t* aAreaName,
  const TPlcMediaChannelID* aChannel,
  const TGLibDateTime* aTimeStamp,
  const double* aX,
  const double* aY,
  const double* aZ,
  const __int32* aLpsTagID,
  const PlcLpsStatus* aLpsStatus,
  const __int32* aLpsCellID,
  const __int32* aLpsAreaID,
  const __int32* aUserParam,
  const wchar_t* aOrder,
  const wchar_t* aShipment,
  const wchar_t* aItem);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLogBarcodeDataLPS(
  ConstHGscAction anAction,
  const wchar_t*& aBarcode,
  const __int64*& aHash,
  const wchar_t*& aScanner,
  const __int64*& aAreaID,
  const wchar_t*& aAreaName,
  const TPlcMediaChannelID*& aChannel,
  const TGLibDateTime*& aTimeStamp,
  const double*& aX,
  const double*& aY,
  const double*& aZ,
  const __int32*& aLpsTagID,
  const PlcLpsStatus*& aLpsStatus,
  const __int32*& aLpsCellID,
  const __int32*& aLpsAreaID,
  const __int32*& aUserParam,
  const wchar_t*& aOrder,
  const wchar_t*& aShipment,
  const wchar_t*& aItem);
```

Delphi – Create

```
function GscAct_CreateLogBarcodeDataLPS(
  aBarcode: PWideChar;
  aHash: PInt64;
  aScanner: PWideChar;
  aAreaID: PInt64;
  aAreaName: PWideChar;
  aChannel: PTPlcMediaChannelID;
  aTimeStamp: PTGLibDateTime;
  aX: PDouble;
  aY: PDouble;
  aZ: PDouble;
  aLpsTagID: PInteger;
  aLpsStatus: PPlcLpsStatus;
  aLpsCellID: PInteger;
  aLpsAreaID: PInteger;
  aUserParam: PInteger;
  aOrder: PWideChar;
  aShipment: PWideChar;
  aItem: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLogBarcodeDataLPS(
  const anAction: HGscAction;
  out aBarcode: PWideChar;
  out aHash: PInt64;
  out aScanner: PWideChar;
  out aAreaID: PInt64;
  out aAreaName: PWideChar;
  out aChannel: PTPlcMediaChannelID;
  out aTimeStamp: PTGLibDateTime;
  out aX: PDouble;
  out aY: PDouble;
  out aZ: PDouble;
  out aLpsTagID: PInteger;
  out aLpsStatus: PPlcLpsStatus;
  out aLpsCellID: PInteger;
  out aLpsAreaID: PInteger;
  out aUserParam: PInteger;
  out aOrder: PWideChar;
  out aShipment: PWideChar;
  out aItem: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

19 System actions

All actions describing system behaviour.

19.1 Blocking filter activate

BlockingFilterActivate (Filter)

Description : *Blocking filter activate.*

Code : ac_BlockingFilterActivate (68)

Class : ak_System (0)

Parameters :

Filter (filter) :
Type : widestring
Description : Blocking filter.

Text

```
BlockingFilterActivate ("filter")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateBlockingFilterActivate(
  const wchar_t* aFilter);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeBlockingFilterActivate(
  ConstHGscAction anAction,
  const wchar_t*& aFilter);
```

Delphi – Create

```
function GscAct_CreateBlockingFilterActivate(
  aFilter: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeBlockingFilterActivate(
  const anAction: HGscAction;
  out aFilter: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.2 Blocking filter deactivate

BlockingFilterDeactivate (Filter)

Description : *Blocking filter deactivate.*

Code : ac_BlockingFilterDeactivate (69)

Class : ak_System (0)

Parameters :

Filter (filter) :
Type : widestring
Description : Blocking filter.

Text

```
BlockingFilterDeactivate ("filter")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateBlockingFilterDeactivate(
  const wchar_t* aFilter);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeBlockingFilterDeactivate(
  ConstHGscAction anAction,
  const wchar_t*& aFilter);
```

Delphi – Create

```
function GscAct_CreateBlockingFilterDeactivate(
  aFilter: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeBlockingFilterDeactivate(
  const anAction: HGscAction;
  out aFilter: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.3 Custom action

CustomAction (Int, String)

GEUTEBRÜCK

Description : *This action has no side effects and can be used for customer purposes.*

Code : `ac_CustomAction` (8)

Class : `ak_System` (0)

Parameters :

Int (INT parameter) :
Type : __int64
Description : Numeric parameter.

String (STRING parameter) :
Type : widestring
Description : Literal parameter.

Text

```
CustomAction (64, "STRING parameter")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCustomAction(
  const __int64& aInt,
  const wchar_t* aString);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCustomAction(
  ConstHGscAction anAction,
  __int64& aInt,
  const wchar_t*& aString);
```

Delphi – Create

```
function GscAct_CreateCustomAction(
  var aInt: Int64;
  aString: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCustomAction(
  const anAction: HGscAction;
  out aInt: Int64;
  out aString: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.4 Database recording info per ring

DatabaseRecordingInfoRing (DatabaseRing, NoVideoRecording, NoAudioRecording, NoRecordingAtAll, VideoSamplesPerSecond, VideoMBPerSecond, AudioSamplesPerSecond, AudioMBPerSecond, WriteWaitTimesPercent, RingCapacity, OldestItem, *RecordingDepth*, *EstimatedRequiredCapacity*)

Description : *Database recording info per ring.*

Code : ac_DatabaseRecordingInfoRing (263)

Class : ak_System (0)

Parameters :

DatabaseRing (database ring) :
Type : PlcDatabaseRing
Description : Database ring.

NoVideoRecording (no video recording) :
Type : bool
Description : Video is recording or not.

NoAudioRecording (no audio recording) :
Type : bool
Description : Audio is recording or not.

NoRecordingAtAll (no recording) :
Type : bool
Description : Video and/or audio is recording or not.

VideoSamplesPerSecond (video samples/s) :
Type : double
Description : Video samples per second.

VideoMBPerSecond (video samples MB/s) :
Type : double
Description : Video MB per second.

AudioSamplesPerSecond (audio samples/s) :
Type : double
Description : Audio samples per second.

AudioMBPerSecond (audio samples MB/s) :
Type : double
Description : Audio MB per second.

WriteWaitTimesPercent (write wait %) :
Type : double
Description : Write waiting times in percent.

RingCapacity (ring capacity) :
Type : __int64
Description : Ring capacity.

OldestItem (oldest item) :
Type : TGLibDateTime
Description : Time stamp of the oldest item.

RecordingDepth (recording depth) [optional] :
Type : double
Description : Recording depth in hours.

EstimatedRequiredCapacity (estimated required capacity) [optional] :
Type : __int64
Description : Estimated required capacity.

Text

```
DatabaseRecordingInfoRing (0, 1, 1, 1, 0.0, 0.0, 0.0,
0.0, 0.0, 64, "2013/09/05 14:59:59,999 GMT+02:00",
RecordingDepth:  0.0, EstimatedRequiredCapacity:  64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDatabaseRecordingInfoRing(
  const PlcDatabaseRing aDatabaseRing,
  const bool aNoVideoRecording,
  const bool aNoAudioRecording,
  const bool aNoRecordingAtAll,
  const double& aVideoSamplesPerSecond,
  const double& aVideoMBPerSecond,
  const double& aAudioSamplesPerSecond,
  const double& aAudioMBPerSecond,
  const double& aWriteWaitTimesPercent,
  const __int64& aRingCapacity,
  const TGLibDateTime& aOldestItem,
  const double* aRecordingDepth,
  const __int64* aEstimatedRequiredCapacity);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDatabaseRecordingInfoRing(
  ConstHGscAction anAction,
  PlcDatabaseRing& aDatabaseRing,
  bool& aNoVideoRecording,
  bool& aNoAudioRecording,
  bool& aNoRecordingAtAll,
  double& aVideoSamplesPerSecond,
  double& aVideoMBPerSecond,
  double& aAudioSamplesPerSecond,
  double& aAudioMBPerSecond,
  double& aWriteWaitTimesPercent,
  __int64& aRingCapacity,
  TGLibDateTime& aOldestItem,
  const double*& aRecordingDepth,
  const __int64*& aEstimatedRequiredCapacity);
```

Delphi – Create

```
function GscAct_CreateDatabaseRecordingInfoRing(
  aDatabaseRing: PlcDatabaseRing;
  aNoVideoRecording: Boolean;
  aNoAudioRecording: Boolean;
  aNoRecordingAtAll: Boolean;
  var aVideoSamplesPerSecond: Double;
  var aVideoMBPerSecond: Double;
  var aAudioSamplesPerSecond: Double;
  var aAudioMBPerSecond: Double;
  var aWriteWaitTimesPercent: Double;
  var aRingCapacity: Int64;
  var aOldestItem: TGLibDateTime;
  aRecordingDepth: PDouble;
  aEstimatedRequiredCapacity: PInt64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDatabaseRecordingInfoRing(
  const anAction: HGscAction;
  out aDatabaseRing: PlcDatabaseRing;
  out aNoVideoRecording: Boolean;
  out aNoAudioRecording: Boolean;
  out aNoRecordingAtAll: Boolean;
  out aVideoSamplesPerSecond: Double;
  out aVideoMBPerSecond: Double;
  out aAudioSamplesPerSecond: Double;
  out aAudioMBPerSecond: Double;
  out aWriteWaitTimesPercent: Double;
  out aRingCapacity: Int64;
  out aOldestItem: TGLibDateTime;
  out aRecordingDepth: PDouble;
  out aEstimatedRequiredCapacity: PInt64)
  : Boolean; stdcall; external GscActionsDll;
```

19.5 Database recording info total

DatabaseRecordingInfoTotal (NoVideoRecording, NoAudioRecording, NoRecordingAtAll, VideoSamplesPerSecond, VideoMBPerSecond, AudioSamplesPerSecond, AudioMBPerSecond, WriteWaitTimesPercent, TotalCapacity, FreeCapacity, AllocatedCapacity, *OldestItem*, *RecordingDepth*, *EstimatedRequiredCapacity*, *RequiredCapacityFactor*, *RequiredCapacityAvailable*)

Description : *Database recording info total.*

Code : ac_DatabaseRecordingInfoTotal (260)

Class : ak_System (0)

Parameters :

NoVideoRecording (no video recording) :
Type : bool
Description : Video is recording or not.

NoAudioRecording (no audio recording) :
Type : bool
Description : Audio is recording or not.

NoRecordingAtAll (no recording) :
Type : bool
Description : Video and/or audio is recording or not.

VideoSamplesPerSecond (video samples/s) :
Type : double
Description : Video samples per second.

VideoMBPerSecond (video samples MB/s) :
Type : double
Description : Video MB per second.

AudioSamplesPerSecond (audio samples/s) :
Type : double
Description : Audio samples per second.

AudioMBPerSecond (audio samples MB/s) :
Type : double
Description : Audio MB per second.

WriteWaitTimesPercent (write wait %) :
Type : double
Description : Write waiting times in percent.

TotalCapacity (total capacity) :
Type : __int64
Description : Total capacity.

FreeCapacity (free capacity) :
Type : __int64
Description : Free capacity.

AllocatedCapacity (allocated capacity) :
Type : __int64
Description : Allocated capacity.

OldestItem (oldest item) [optional] :
Type : TGLibDateTime
Description : Time stamp of the oldest item.

RecordingDepth (recording depth) [optional] :
Type : double
Description : Recording depth in hours.

EstimatedRequiredCapacity (estimated required capacity) [optional] :
Type : __int64
Description : Estimated required capacity.

RequiredCapacityFactor (required capacity factor) [optional] :
Type : double
Description : Required capacity factor.

RequiredCapacityAvailable (required capacity available) [optional] :
Type : bool
Description : Required capacity available.

Text

```
DatabaseRecordingInfoTotal (1, 1, 1, 0.0,
0.0, 0.0, 0.0, 0.0, 64, 64, 64, OldestItem:
"2013/09/05 14:59:59,999 GMT+02:00",
RecordingDepth:  0.0, EstimatedRequiredCapacity:  64,
RequiredCapacityFactor:  0.0, RequiredCapacityAvailable:
1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDatabaseRecordingInfoTotal(
  const bool aNoVideoRecording,
  const bool aNoAudioRecording,
  const bool aNoRecordingAtAll,
  const double& aVideoSamplesPerSecond,
  const double& aVideoMBPerSecond,
  const double& aAudioSamplesPerSecond,
  const double& aAudioMBPerSecond,
  const double& aWriteWaitTimesPercent,
  const __int64& aTotalCapacity,
  const __int64& aFreeCapacity,
  const __int64& aAllocatedCapacity,
  const TGLibDateTime* aOldestItem,
  const double* aRecordingDepth,
  const __int64* aEstimatedRequiredCapacity,
  const double* aRequiredCapacityFactor,
  const bool* aRequiredCapacityAvailable);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDatabaseRecordingInfoTotal(
  ConstHGscAction anAction,
  bool& aNoVideoRecording,
  bool& aNoAudioRecording,
  bool& aNoRecordingAtAll,
  double& aVideoSamplesPerSecond,
  double& aVideoMBPerSecond,
  double& aAudioSamplesPerSecond,
  double& aAudioMBPerSecond,
  double& aWriteWaitTimesPercent,
  __int64& aTotalCapacity,
  __int64& aFreeCapacity,
  __int64& aAllocatedCapacity,
  const TGLibDateTime*& aOldestItem,
  const double*& aRecordingDepth,
  const __int64*& aEstimatedRequiredCapacity,
  const double*& aRequiredCapacityFactor,
  const bool*& aRequiredCapacityAvailable);
```

Delphi – Create

```
function GscAct_CreateDatabaseRecordingInfoTotal(
  aNoVideoRecording: Boolean;
  aNoAudioRecording: Boolean;
  aNoRecordingAtAll: Boolean;
  var aVideoSamplesPerSecond: Double;
  var aVideoMBPerSecond: Double;
  var aAudioSamplesPerSecond: Double;
  var aAudioMBPerSecond: Double;
  var aWriteWaitTimesPercent: Double;
  var aTotalCapacity: Int64;
  var aFreeCapacity: Int64;
  var aAllocatedCapacity: Int64;
  aOldestItem: PTGLibDateTime;
  aRecordingDepth: PDouble;
  aEstimatedRequiredCapacity: PInt64;
  aRequiredCapacityFactor: PDouble;
  aRequiredCapacityAvailable: PBoolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDatabaseRecordingInfoTotal(
  const anAction: HGscAction;
  out aNoVideoRecording: Boolean;
  out aNoAudioRecording: Boolean;
  out aNoRecordingAtAll: Boolean;
  out aVideoSamplesPerSecond: Double;
  out aVideoMBPerSecond: Double;
  out aAudioSamplesPerSecond: Double;
  out aAudioMBPerSecond: Double;
  out aWriteWaitTimesPercent: Double;
  out aTotalCapacity: Int64;
  out aFreeCapacity: Int64;
  out aAllocatedCapacity: Int64;
  out aOldestItem: PTGLibDateTime;
  out aRecordingDepth: PDouble;
  out aEstimatedRequiredCapacity: PInt64;
  out aRequiredCapacityFactor: PDouble;
  out aRequiredCapacityAvailable: PBoolean)
  : Boolean; stdcall; external GscActionsDll;
```

19.6 Database started

DatabaseStarted (Status, TotalSize)

Description : *This action will be fired at the database start-up.*

Code : ac_DatabaseStarted (9)

Class : ak_System (0)

Parameters :

Status (status) :
Type : PlcDatabaseStatus
Description : Database status message.

TotalSize (total size) :
Type : __int64
Description : Database total size.

Text

```
DatabaseStarted (0, 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateDatabaseStarted(
  const PlcDatabaseStatus aStatus,
  const __int64& aTotalSize);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeDatabaseStarted(
  ConstHGscAction anAction,
  PlcDatabaseStatus& aStatus,
  __int64& aTotalSize);
```

Delphi – Create

```
function GscAct_CreateDatabaseStarted(
  aStatus: PlcDatabaseStatus;
  var aTotalSize: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeDatabaseStarted(
  const anAction: HGscAction;
  out aStatus: PlcDatabaseStatus;
  out aTotalSize: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

19.7 Event recording changed

EventRecordingChanged (EventID, TypeID)

Description : *Event recording settings are changed.*

Code : ac_EventRecordingChanged (61)

Class : ak_System (0)

Parameters :

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

Text

```
EventRecordingChanged (64, "event type")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventRecordingChanged(
  const __int64& aEventID,
  const TPlcEventTypeID& aTypeID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventRecordingChanged(
  ConstHGscAction anAction,
  __int64& aEventID,
  TPlcEventTypeID& aTypeID);
```

Delphi – Create

```
function GscAct_CreateEventRecordingChanged(
  var aEventID: Int64;
  var aTypeID: TPlcEventTypeID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeEventRecordingChanged(
  const anAction: HGscAction;
  out aEventID: Int64;
  out aTypeID: TPlcEventTypeID)
  : Boolean; stdcall; external GscActionsDll;
```

19.8 Event started

EventStarted (EventID, TypeID, *ForeignKey*)

Description : *Event has started.*

Attention : *This action has extended create and decode functions!*

Code : ac_EventStarted (59)

Class : ak_System (0)

Parameters :

EventID (instance ID) :
Type : _int64
Description : Instance ID of the event.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

ForeignKey (foreign key) [optional] [ForeignKey] :
Type : __int64
Description : Optional foreign key used to start the alarm.

Text

```
EventStarted (64, "event type", ForeignKey:  64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventStartedEx(
  const __int64& aEventID,
  const TPlcEventTypeID& aTypeID,
  const __int64* aForeignKey);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventStartedEx(
  ConstHGscAction anAction,
  __int64& aEventID,
  TPlcEventTypeID& aTypeID,
  const __int64*& aForeignKey);
```

Delphi – Create

```
function GscAct_CreateEventStartedEx(
  var aEventID: Int64;
  var aTypeID: TPlcEventTypeID;
  aForeignKey: PInt64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeEventStartedEx(
  const anAction: HGscAction;
  out aEventID: Int64;
  out aTypeID: TPlcEventTypeID;
  out aForeignKey: PInt64)
  : Boolean; stdcall; external GscActionsDll;
```

19.9 Event stopped

```
EventStopped (EventID, TypeID)
```

Description : *Event has stopped.*

Code : ac_EventStopped (60)

Class : ak_System (0)

Parameters :

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

Text

```
EventStopped (64, "event type")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateEventStopped(
  const __int64& aEventID,
  const TPlcEventTypeID& aTypeID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeEventStopped(
  ConstHGscAction anAction,
  __int64& aEventID,
  TPlcEventTypeID& aTypeID);
```

Delphi – Create

```
function GscAct_CreateEventStopped(
  var aEventID: Int64;
  var aTypeID: TPlcEventTypeID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeEventStopped(
  const anAction: HGscAction;
  out aEventID: Int64;
  out aTypeID: TPlcEventTypeID)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

19.10 FRC notification

FRCNotification (Notification, *Param*, *Description*, *XMLInfo*)

Description : *FRC notification.*

Code : ac_FRCNotification (19)

Class : ak_System (0)

Parameters :

Notification (notification) :
Type : PlcFRCNotification
Description : Notification reason.

Param (param) [optional] :
Type : __int32
Description : Additional parameter.

Description (description) [optional] :
Type : widestring
Description : Optional notification text.

XMLInfo (additional info) [optional] :
Type : string
Description : Optional additional info (usually as XML string).

Text

```
FRCNotification (0, Param:  32, Description:
"description", XMLInfo:  "additional info")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateFRCNotification(
  const PlcFRCNotification aNotification,
  const __int32* aParam,
  const wchar_t* aDescription,
  const char* aXMLInfo);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeFRCNotification(
  ConstHGscAction anAction,
  PlcFRCNotification& aNotification,
  const __int32*& aParam,
  const wchar_t*& aDescription,
  const char*& aXMLInfo);
```

Delphi – Create

```
function GscAct_CreateFRCNotification(
  aNotification: PlcFRCNotification;
  aParam: PInteger;
  aDescription: PWideChar;
  aXMLInfo: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeFRCNotification(
  const anAction: HGscAction;
  out aNotification: PlcFRCNotification;
  out aParam: PInteger;
  out aDescription: PWideChar;
  out aXMLInfo: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.11 GEMOS alarm

GEMOSalarm (GEMOSkey, GEMOSint, GEMOSstr)

Description : *GEMOS alarm notification.*

Code : ac_GEMOSalarm (300)

Class : ak_System (0)

Parameters :

GEMOSkey (GEMOS key) [ForeignKey] :
Type : __int64
Description : GEMOS alarm key.

GEMOSint (GEMOS int) :
Type : __int32
Description : GEMOS alarm integer parameter.

GEMOSstr (GEMOS str) :
Type : widestring
Description : GEMOS alarm string parameter.

Text

```
GEMOSalarm (64, 32, "GEMOS str")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGEMOSalarm(
  const __int64& aGEMOSkey,
  const __int32 aGEMOSint,
  const wchar_t* aGEMOSstr);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGEMOSalarm(
  ConstHGscAction anAction,
  __int64& aGEMOSkey,
  __int32& aGEMOSint,
  const wchar_t*& aGEMOSstr);
```

Delphi – Create

```
function GscAct_CreateGEMOSalarm(
  var aGEMOSkey: Int64;
  aGEMOSint: Integer;
  aGEMOSstr: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGEMOSalarm(
  const anAction: HGscAction;
  out aGEMOSkey: Int64;
  out aGEMOSint: Integer;
  out aGEMOSstr: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.12 IP switch operation

IPSwitchOperation (Operation, Port, *Param*)

Description : *IP switch operation.*

Code : ac_IPSwitchOperation (190)

Class : ak_System (0)

Parameters :

Operation (operation) :
Type : IPSwitchOps
Description : Operation.

Port (port no) :
Type : widestring
Description : Port no, IP address, MAC address.

Param (parameter) [optional] :
Type : __int32
Description : Optional parameter.

Text

```
IPSwitchOperation (0, "port no", Param:  32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIPSwitchOperation(
  const IPSwitchOps aOperation,
  const wchar_t* aPort,
  const __int32* aParam);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIPSwitchOperation(
  ConstHGscAction anAction,
  IPSwitchOps& aOperation,
  const wchar_t*& aPort,
  const __int32*& aParam);
```

Delphi – Create

```
function GscAct_CreateIPSwitchOperation(
  aOperation: IPSwitchOps;
  aPort: PWideChar;
  aParam: PInteger)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIPSwitchOperation(
  const anAction: HGscAction;
  out aOperation: IPSwitchOps;
  out aPort: PWideChar;
  out aParam: PInteger)
  : Boolean; stdcall; external GscActionsDll;
```

19.13 Kill all events

```
KillAllEvents ()
```

GEUTEBRÜCK

Description : *Kill all active events.*

Code : `ac_KillAllEvents` (65)

Class : `ak_System` (0)

This action has no parameters.

Text

```
KillAllEvents ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateKillAllEvents();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeKillAllEvents(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateKillAllEvents()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeKillAllEvents(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

19.14 Kill event

```
KillEvent (TypeID)
```

Description : *Kill event.*

Code : `ac_KillEvent` (63)

Class : `ak_System` (0)

Parameters :

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

Text

```
KillEvent ("event type")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateKillEvent(
  const TPlcEventTypeID& aTypeID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeKillEvent(
  ConstHGscAction anAction,
  TPlcEventTypeID& aTypeID);
```

Delphi – Create

```
function GscAct_CreateKillEvent(
  var aTypeID: TPlcEventTypeID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeKillEvent(
  const anAction: HGscAction;
  out aTypeID: TPlcEventTypeID)
  : Boolean; stdcall; external GscActionsDll;
```

19.15 Kill event by instance

KillEventByID (EventID)

Description : *Kill event by instance ID.*

Code : ac_KillEventByID (64)

Class : ak_System (0)

Parameters :

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

Text

```
KillEventByID (64)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateKillEventByID(
  const __int64& aEventID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeKillEventByID(
  ConstHGscAction anAction,
  __int64& aEventID);
```

Delphi – Create

```
function GscAct_CreateKillEventByID(
  var aEventID: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeKillEventByID(
  const anAction: HGscAction;
  out aEventID: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

19.16 Live check

LiveCheck (Counter, Date)

Description : *This action will be fired every 10 seconds and is intended for use as live check.*

Code : ac_LiveCheck (7)

Class : ak_System (0)

Parameters :

Counter (counter) :
Type : __int64
Description : This is the number of already fired live check actions.

Date (time stamp) :
Type : TGLibDateTime
Description : Current server time.

Text

```
LiveCheck (64, "2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateLiveCheck(
  const __int64& aCounter,
  const TGLibDateTime& aDate);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeLiveCheck(
  ConstHGscAction anAction,
  __int64& aCounter,
  TGLibDateTime& aDate);
```

Delphi – Create

```
function GscAct_CreateLiveCheck(
  var aCounter: Int64;
  var aDate: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeLiveCheck(
  const anAction: HGscAction;
  out aCounter: Int64;
  out aDate: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

19.17 Redundant power failure

RedundantPowerFailure ()

Description : *This action is notified if a failure of the redundant power supply is detected.*

Code : ac_RedundantPowerFailure (37)

Class : ak_System (0)

This action has no parameters.

Text

```
RedundantPowerFailure ()
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateRedundantPowerFailure();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeRedundantPowerFailure(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateRedundantPowerFailure()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeRedundantPowerFailure(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

19.18 Redundant power ok

RedundantPowerOk ()

Description : *This action is notified if a failure of the redundant power supply is removed.*

Code : ac_RedundantPowerOk (38)

Class : ak_System (0)

This action has no parameters.

Text

RedundantPowerOk ()

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateRedundantPowerOk();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeRedundantPowerOk(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateRedundantPowerOk()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeRedundantPowerOk(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

19.19 SMRP viewer cleared

SMRPViewerCleared ()

Description : *SMRP viewer cleared.*

Code : ac_SMRPViewerCleared (341)

Class : ak_System (0)

This action has no parameters.

Text

```
SMRPViewerCleared ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSMRPViewerCleared();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSMRPViewerCleared(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateSMRPViewerCleared()
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeSMRPViewerCleared(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

19.20 SMRP viewer connected

SMRPViewerConnected (Server, Channel)

Description : *SMRP viewer connected to the camera.*

Code : ac_SMRPViewerConnected (340)

Class : ak_System (0)

Parameters :

Server (server) :
Type : widestring
Description : Server name.

Channel (channel) :
Type : TMediaChannelID
Description : Channel.

Text

```
SMRPViewerConnected ("server", 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSMRPViewerConnected(
  const wchar_t* aServer,
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSMRPViewerConnected(
  ConstHGscAction anAction,
  const wchar_t*& aServer,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateSMRPViewerConnected(
  aServer: PWideChar;
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSMRPViewerConnected(
  const anAction: HGscAction;
  out aServer: PWideChar;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

19.21 SMTP mail

SMTPMailSend (Subject, To, Cc, Body, *Channel*)

Description : *This action will send a user defined email if GSCMail is connected*

Code : ac_SMTPMailSend (303)

Class : ak_System (0)

Parameters :

Subject (subject) :
Type : widestring
Description : Mail subject.

To (to) :
Type : widestring
Description : Mail recipients.

Cc (cc) :
Type : widestring
Description : Carbon copy recipients.

Body (body) :
Type : widestring
Description : Mail body.

Channel (channel) [optional] [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
SMTPMailSend ("subject", "to", "cc", "body", Channel:
32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSMTPMailSend(
  const wchar_t* aSubject,
  const wchar_t* aTo,
  const wchar_t* aCc,
  const wchar_t* aBody,
  const TPlcMediaChannelID* aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSMTPMailSend(
  ConstHGscAction anAction,
  const wchar_t*& aSubject,
  const wchar_t*& aTo,
  const wchar_t*& aCc,
  const wchar_t*& aBody,
  const TPlcMediaChannelID*& aChannel);
```

Delphi – Create

```
function GscAct_CreateSMTPMailSend(
  aSubject: PWideChar;
  aTo: PWideChar;
  aCc: PWideChar;
  aBody: PWideChar;
  aChannel: PTPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSMTPMailSend(
  const anAction: HGscAction;
  out aSubject: PWideChar;
  out aTo: PWideChar;
  out aCc: PWideChar;
  out aBody: PWideChar;
  out aChannel: PTPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

19.22 Set clock

SetClock (Date)

Description : *Set clock.*

Code : ac_SetClock (25)

Class : ak_System (0)

Parameters :

Date (current time) :
Type : TGLibDateTime
Description : Current time.

Text

```
SetClock ("2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetClock(
  const TGLibDateTime& aDate);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetClock(
  ConstHGscAction anAction,
  TGLibDateTime& aDate);
```

Delphi – Create

```
function GscAct_CreateSetClock(
  var aDate: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetClock(
  const anAction: HGscAction;
  out aDate: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

19.23 Set watchdog

SetWatchdog (Timeout)

Description : *Set watchdog.*

Code : ac_SetWatchdog (53)

Class : ak_System (0)

Parameters :

Timeout (timeout) :
Type : __int32
Description : Timeout in seconds, before the watchdog must be retriggered and before the hardware watchdog will set the hardware contact.

Text

```
SetWatchdog (32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetWatchdog(
  const __int32 aTimeout);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetWatchdog(
  ConstHGscAction anAction,
  __int32& aTimeout);
```

Delphi – Create

```
function GscAct_CreateSetWatchdog(
  aTimeout: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetWatchdog(
  const anAction: HGscAction;
  out aTimeout: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

19.24 Setup changed

SetupChanged (User, Host, Date, ResourceKind, ResourceID, ChangeKind, *Details*, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Setup changed.*

Code : ac_SetupChanged (17)

Class : ak_System (0)

Parameters :

User (user name) [UserName] :
Type : widestring
Description : Name of the user who modified the setup.

Host (remote host) :
Type : widestring
Description : Host from where the connection was done.

Date (current time) :
Type : TGLibDateTime
Description : Current time.

ResourceKind (resource kind) :
Type : PlcResourceKind
Description : Modified resource kind.

ResourceID (resource ID) :
Type : TEventTypeID
Description : Modified resource ID.

ChangeKind (change kind) :
Type : PlcResourceChangeKind
Description : Change kind.

Details (details) [optional] :
Type : widestring
Description : Details of the modification.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
SetupChanged ("user name", "remote host",
"2013/09/05 14:59:59,999 GMT+02:00", 0, "resource ID",
0, Details: "details", ClientHost: "client host",
ClientType: 0, ClientAccount: "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateSetupChanged(
  const wchar_t* aUser,
  const wchar_t* aHost,
  const TGLibDateTime& aDate,
  const PlcResourceKind aResourceKind,
  const TPlcEventTypeID& aResourceID,
  const PlcResourceChangeKind aChangeKind,
  const wchar_t* aDetails,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetupChanged(
  ConstHGscAction anAction,
  const wchar_t*& aUser,
  const wchar_t*& aHost,
  TGLibDateTime& aDate,
  PlcResourceKind& aResourceKind,
  TPlcEventTypeID& aResourceID,
  PlcResourceChangeKind& aChangeKind,
  const wchar_t*& aDetails,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateSetupChanged(
  aUser: PWideChar;
  aHost: PWideChar;
  var aDate: TGLibDateTime;
  aResourceKind: PlcResourceKind;
  var aResourceID: TPlcEventTypeID;
  aChangeKind: PlcResourceChangeKind;
  aDetails: PWideChar;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetupChanged(
  const anAction: HGscAction;
  out aUser: PWideChar;
  out aHost: PWideChar;
  out aDate: TGLibDateTime;
  out aResourceKind: PlcResourceKind;
  out aResourceID: TPlcEventTypeID;
  out aChangeKind: PlcResourceChangeKind;
  out aDetails: PWideChar;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.25 Setup upload progress

SetupUploadProgress (*User1*, *User2*, *Host*, Progress, Date)

Description : *Setup upload progress.*

Code : ac_SetupUploadProgress (16)

Class : ak_System (0)

Parameters :

User1 (first user) [optional] [UserName] :
Type : widestring
Description : Name of the user who modified the setup.

User2 (second user) [optional] [UserName] :
Type : widestring
Description : Name of the second user by four eyes authentication.

Host (remote host) [optional] :
Type : widestring
Description : Host from where the connection was done.

Progress (progress %) :
Type : __int32
Description : Progress in percent.

Date (current time) :
Type : TGLibDateTime
Description : Current stage time.

Text

```
SetupUploadProgress (User1:  "first user",
User2:  "second user", Host:  "remote host", 32,
"2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetupUploadProgress(
  const wchar_t* aUser1,
  const wchar_t* aUser2,
  const wchar_t* aHost,
  const __int32 aProgress,
  const TGLibDateTime& aDate);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetupUploadProgress(
  ConstHGscAction anAction,
  const wchar_t*& aUser1,
  const wchar_t*& aUser2,
  const wchar_t*& aHost,
  __int32& aProgress,
  TGLibDateTime& aDate);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateSetupUploadProgress(
  aUser1: PWideChar;
  aUser2: PWideChar;
  aHost: PWideChar;
  aProgress: Integer;
  var aDate: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetupUploadProgress(
  const anAction: HGscAction;
  out aUser1: PWideChar;
  out aUser2: PWideChar;
  out aHost: PWideChar;
  out aProgress: Integer;
  out aDate: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

19.26 Start event

StartEvent (TypeID, *ForeignKey*)

Description : *Start event.*

Attention : *This action has extended create and decode functions!*

Code : ac_StartEvent (54)

Class : ak_System (0)

Parameters :

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

ForeignKey (foreign key) [optional] [ForeignKey] :
Type : __int64
Description : Optional foreign key used to store with alarm.

Text

```
StartEvent ("event type", ForeignKey:  64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateStartEventEx(
  const TPlcEventTypeID& aTypeID,
  const __int64* aForeignKey);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeStartEventEx(
  ConstHGscAction anAction,
  TPlcEventTypeID& aTypeID,
  const __int64*& aForeignKey);
```

Delphi – Create

```
function GscAct_CreateStartEventEx(
  var aTypeID: TPlcEventTypeID;
  aForeignKey: PInt64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeStartEventEx(
  const anAction: HGscAction;
  out aTypeID: TPlcEventTypeID;
  out aForeignKey: PInt64)
  : Boolean; stdcall; external GscActionsDll;
```

19.27 Stop all events

StopAllEvents ()

Description : *Stop all active events.*

Code : ac_StopAllEvents (62)

Class : ak_System (0)

This action has no parameters.

Text

```
StopAllEvents ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateStopAllEvents();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeStopAllEvents(
  ConstHGscAction anAction);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateStopAllEvents()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeStopAllEvents(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

19.28 Stop event

StopEvent (TypeID)

Description : *Stop event.*

Code : ac_StopEvent (55)

Class : ak_System (0)

Parameters :

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

Text

```
StopEvent ("event type")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateStopEvent(
  const TPlcEventTypeID& aTypeID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeStopEvent(
  ConstHGscAction anAction,
  TPlcEventTypeID& aTypeID);
```

Delphi – Create

```
function GscAct_CreateStopEvent(
  var aTypeID: TPlcEventTypeID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeStopEvent(
  const anAction: HGscAction;
  out aTypeID: TPlcEventTypeID)
  : Boolean; stdcall; external GscActionsDll;
```

19.29 Stop event by instance

StopEventByID (EventID)

Description : *Stop event by instance ID.*

Code : ac_StopEventByID (56)

Class : ak_System (0)

Parameters :

EventID (instance ID) :
Type : __int64
Description : Instance ID of the event.

Text

```
StopEventByID (64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateStopEventByID(
  const __int64& aEventID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeStopEventByID(
  ConstHGscAction anAction,
  __int64& aEventID);
```

Delphi – Create

```
function GscAct_CreateStopEventByID(
  var aEventID: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeStopEventByID(
  const anAction: HGscAction;
  out aEventID: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

19.30 System error

SystemError (Source, Message, *WindowsError*, *Description*, *XMLInfo*)

Description : *Notify system error.*

Code : ac_SystemError (52)

Class : ak_System (0)

Parameters :

Source (source subsystem) :
Type : PlcMessageSource
Description : Source of the message.

Message (message code) :
Type : PlcMessageCode
Description : Kind of the message.

WindowsError (Windows error code) [optional] [WindowsError] :
Type : __int32
Description : Optional Windows error code.

Description (description) [optional] :
Type : widestring
Description : Optional description of the message.

XMLInfo (additional info) [optional] :
Type : string
Description : Optional additional info (usually as XML string).

Text

```
SystemError (0, 0, WindowsError:  32, Description:
"description", XMLInfo:  "additional info")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSystemError(
  const PlcMessageSource aSource,
  const PlcMessageCode aMessage,
  const __int32* aWindowsError,
  const wchar_t* aDescription,
  const char* aXMLInfo);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSystemError(
  ConstHGscAction anAction,
  PlcMessageSource& aSource,
  PlcMessageCode& aMessage,
  const __int32*& aWindowsError,
  const wchar_t*& aDescription,
  const char*& aXMLInfo);
```

Delphi – Create

```
function GscAct_CreateSystemError(
  aSource: PlcMessageSource;
  aMessage: PlcMessageCode;
  aWindowsError: PInteger;
  aDescription: PWideChar;
  aXMLInfo: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSystemError(
  const anAction: HGscAction;
  out aSource: PlcMessageSource;
  out aMessage: PlcMessageCode;
  out aWindowsError: PInteger;
  out aDescription: PWideChar;
  out aXMLInfo: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.31 System info

SystemInfo (Source, Message, *Description*, *XMLInfo*)

Description : *Notify system information.*

Code : ac_SystemInfo (50)

Class : ak_System (0)

Parameters :

Source (source subsystem) :
Type : PlcMessageSource
Description : Source of the message.

Message (message code) :
Type : PlcMessageCode
Description : Kind of the message.

Description (description) [optional] :
Type : widestring
Description : Optional description of the message.

XMLInfo (additional info) [optional] :
Type : string
Description : Optional additional info (usually as XML string).

Text

```
SystemInfo (0, 0, Description:  "description", XMLInfo:
"additional info")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSystemInfo(
  const PlcMessageSource aSource,
  const PlcMessageCode aMessage,
  const wchar_t* aDescription,
  const char* aXMLInfo);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSystemInfo(
  ConstHGscAction anAction,
  PlcMessageSource& aSource,
  PlcMessageCode& aMessage,
  const wchar_t*& aDescription,
  const char*& aXMLInfo);
```

Delphi – Create

```
function GscAct_CreateSystemInfo(
  aSource: PlcMessageSource;
  aMessage: PlcMessageCode;
  aDescription: PWideChar;
  aXMLInfo: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSystemInfo(
  const anAction: HGscAction;
  out aSource: PlcMessageSource;
  out aMessage: PlcMessageCode;
  out aDescription: PWideChar;
  out aXMLInfo: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.32 System settings changed

```
SystemSettingsChanged (SetupChanged, User1, User2, Host,
TimeRangeChanged, TimeRange, LicenceChanged, Date)
```

Description : *Setup of the system and/or the current time range changed.*

Code : ac_SystemSettingsChanged (6)

Class : ak_System (0)

Parameters :

SetupChanged (setup changed) :
Type : bool
Description : System setup has changed.

User1 (first user) [optional] [UserName] :
Type : widestring
Description : Name of the user who modified the setup.

User2 (second user) [optional] [UserName] :
Type : widestring
Description : Name of the second user by four eyes authentication.

Host (remote host) [optional] :
Type : widestring
Description : Host from where the connection was done.

TimeRangeChanged (time range changed) :
Type : bool
Description : Time range has changed.

TimeRange (current time range) [optional] [TimeRange] :
Type : TResourceID
Description : Currently active time range.

LicenceChanged (licence changed) :
Type : bool
Description : Licence has changed.

Date (change time) :
Type : TGLibDateTime
Description : Time of the system settings changed.

Text

```
SystemSettingsChanged (1, User1: "first
user", User2: "second user", Host: "remote
host", 1, TimeRange: "current time range", 1,
"2013/09/05 14:59:59,999 GMT+02:00")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSystemSettingsChanged(
  const bool aSetupChanged,
  const wchar_t* aUser1,
  const wchar_t* aUser2,
  const wchar_t* aHost,
  const bool aTimeRangeChanged,
  const TPlcResourceID* aTimeRange,
  const bool aLicenceChanged,
  const TGLibDateTime& aDate);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSystemSettingsChanged(
  ConstHGscAction anAction,
  bool& aSetupChanged,
  const wchar_t*& aUser1,
  const wchar_t*& aUser2,
  const wchar_t*& aHost,
  bool& aTimeRangeChanged,
  const TPlcResourceID*& aTimeRange,
  bool& aLicenceChanged,
  TGLibDateTime& aDate);
```

Delphi – Create

```
function GscAct_CreateSystemSettingsChanged(
  aSetupChanged: Boolean;
  aUser1: PWideChar;
  aUser2: PWideChar;
  aHost: PWideChar;
  aTimeRangeChanged: Boolean;
  aTimeRange: PTPlcResourceID;
  aLicenceChanged: Boolean;
  var aDate: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSystemSettingsChanged(
  const anAction: HGscAction;
  out aSetupChanged: Boolean;
  out aUser1: PWideChar;
  out aUser2: PWideChar;
  out aHost: PWideChar;
  out aTimeRangeChanged: Boolean;
  out aTimeRange: PTPlcResourceID;
  out aLicenceChanged: Boolean;
  out aDate: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

19.33 System started

SystemStarted (Date)

Description : *This action will be fired only once at the system start-up.*

Code : ac_SystemStarted (1)

Class : ak_System (0)

Parameters :

Date (start time) :
Type : TGLibDateTime
Description : Time of the system start-up.

Text

```
SystemStarted ("2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSystemStarted(
  const TGLibDateTime& aDate);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSystemStarted(
  ConstHGscAction anAction,
  TGLibDateTime& aDate);
```

Delphi – Create

```
function GscAct_CreateSystemStarted(
  var aDate: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSystemStarted(
  const anAction: HGscAction;
  out aDate: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

19.34 System terminating

SystemTerminating (Date, WindowsShutdown)

Description : *This action will be fired when the system is going to be shut down.*

Code : ac_SystemTerminating (2)

Class : ak_System (0)

Parameters :

Date (stop time) :
Type : TGLibDateTime
Description : Time of the system shutdown.

WindowsShutdown (Windows shutdown) :
Type : bool
Description : Indicates whether the system shutdown is done due to the windows shutdown.

Text

```
SystemTerminating ("2013/09/05 14:59:59,999 GMT+02:00",
1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSystemTerminating(
  const TGLibDateTime& aDate,
  const bool aWindowsShutdown);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSystemTerminating(
  ConstHGscAction anAction,
  TGLibDateTime& aDate,
  bool& aWindowsShutdown);
```

Delphi – Create

```
function GscAct_CreateSystemTerminating(
  var aDate: TGLibDateTime;
  aWindowsShutdown: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSystemTerminating(
  const anAction: HGscAction;
  out aDate: TGLibDateTime;
  out aWindowsShutdown: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

19.35 System warning

SystemWarning (Source, Message, *WindowsError*, *Description*, *XMLInfo*)

Description : *Notify system warning.*

Code : ac_SystemWarning (51)

Class : ak_System (0)

Parameters :

Source (source subsystem) :
Type : PlcMessageSource
Description : Source of the message.

Message (message code) :
Type : PlcMessageCode
Description : Kind of the message.

WindowsError (Windows error code) [optional] [WindowsError] :
Type : __int32
Description : Optional Windows error code.

Description (description) [optional] :
Type : widestring
Description : Optional description of the message.

XMLInfo (additional info) [optional] :
Type : string
Description : Optional additional info (usually as XML string).

Text

```
SystemWarning (0, 0, WindowsError:  32, Description:
"description", XMLInfo:  "additional info")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSystemWarning(
  const PlcMessageSource aSource,
  const PlcMessageCode aMessage,
  const __int32* aWindowsError,
  const wchar_t* aDescription,
  const char* aXMLInfo);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSystemWarning(
  ConstHGscAction anAction,
  PlcMessageSource& aSource,
  PlcMessageCode& aMessage,
  const __int32*& aWindowsError,
  const wchar_t*& aDescription,
  const char*& aXMLInfo);
```

Delphi – Create

```
function GscAct_CreateSystemWarning(
  aSource: PlcMessageSource;
  aMessage: PlcMessageCode;
  aWindowsError: PInteger;
  aDescription: PWideChar;
  aXMLInfo: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSystemWarning(
  const anAction: HGscAction;
  out aSource: PlcMessageSource;
  out aMessage: PlcMessageCode;
  out aWindowsError: PInteger;
  out aDescription: PWideChar;
  out aXMLInfo: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.36 Transfer binary buffer

TransferBinaryBuffer (InternalHandle, Parameter)

Description : *Transfer binary buffer.*

Code : ac_TransferBinaryBuffer (380)

Class : ak_System (0)

Parameters :

InternalHandle (internal handle) :
Type : GUID
Description : Internal handle.

Parameter (parameter) :
Type : widestring
Description : Parameter.

Text

```
TransferBinaryBuffer ("00000000-0000-0000-0000-000000000000",
"parameter")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTransferBinaryBuffer(
  const GUID& aInternalHandle,
  const wchar_t* aParameter);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTransferBinaryBuffer(
  ConstHGscAction anAction,
  GUID& aInternalHandle,
  const wchar_t*& aParameter);
```

Delphi – Create

```
function GscAct_CreateTransferBinaryBuffer(
  var aInternalHandle: TGuid;
  aParameter: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeTransferBinaryBuffer(
  const anAction: HGscAction;
  out aInternalHandle: TGuid;
  out aParameter: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.37 Transfer binary channel buffer

TransferBinaryChannelBuffer (Channel, InternalHandle, Parameter)

Description : *Transfer binary channel buffer.*

Code : ac_TransferBinaryChannelBuffer (381)

Class : ak_System (0)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

InternalHandle (internal handle) :
Type : GUID
Description : Internal handle.

Parameter (parameter) :
Type : widestring
Description : Parameter.

Text

```
TransferBinaryChannelBuffer (32, "00000000-0000-0000-0000-000000000000",
"parameter")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateTransferBinaryChannelBuffer(
  const TPlcMediaChannelID& aChannel,
  const GUID& aInternalHandle,
  const wchar_t* aParameter);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeTransferBinaryChannelBuffer(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  GUID& aInternalHandle,
  const wchar_t*& aParameter);
```

Delphi – Create

```
function GscAct_CreateTransferBinaryChannelBuffer(
  var aChannel: TPlcMediaChannelID;
  var aInternalHandle: TGuid;
  aParameter: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeTransferBinaryChannelBuffer(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aInternalHandle: TGuid;
  out aParameter: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.38 User login

UserLogin (User1, *User2*, Host, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *This action will be fired when the user has connected to the system.*

Code : ac_UserLogin (3)

Class : ak_System (0)

Parameters :

User1 (first user) [UserName] :
Type : widestring
Description : Name of the user connected to the system.

User2 (second user) [optional] [UserName] :
Type : widestring
Description : Name of the second user by four eyes authentication.

Host (remote host) :
Type : widestring
Description : Host from where the connection is done.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
UserLogin ("first user", User2:  "second user", "remote
host", ClientHost:  "client host", ClientType:  0,
ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateUserLogin(
  const wchar_t* aUser1,
  const wchar_t* aUser2,
  const wchar_t* aHost,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeUserLogin(
  ConstHGscAction anAction,
  const wchar_t*& aUser1,
  const wchar_t*& aUser2,
  const wchar_t*& aHost,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateUserLogin(
  aUser1: PWideChar;
  aUser2: PWideChar;
  aHost: PWideChar;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeUserLogin(
  const anAction: HGscAction;
  out aUser1: PWideChar;
  out aUser2: PWideChar;
  out aHost: PWideChar;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.39 User login failed

UserLoginFailed (User1, *User2*, Host, RejectReason, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *This action will be fired when the user has tried to connect to the system but was rejected.*

Code : ac_UserLoginFailed (4)

Class : ak_System (0)

Parameters :

User1 (first user) [UserName] :
Type : widestring
Description : Name of the user who tried to connect to the system.

User2 (second user) [optional] [UserName] :
Type : widestring
Description : Name of the second user by four eyes authentication.

Host (remote host) :
Type : widestring
Description : Host from where the connection is done.

RejectReason (reject reason) :
Type : UserLoginFailureCode
Description : Reason of the rejection.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

GEUTEBRÜCK

Text

```
UserLoginFailed ("first user", User2:  "second user",
"remote host", 0, ClientHost:  "client host", ClientType:
0, ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateUserLoginFailed(
  const wchar_t* aUser1,
  const wchar_t* aUser2,
  const wchar_t* aHost,
  const UserLoginFailureCode aRejectReason,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeUserLoginFailed(
  ConstHGscAction anAction,
  const wchar_t*& aUser1,
  const wchar_t*& aUser2,
  const wchar_t*& aHost,
  UserLoginFailureCode& aRejectReason,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateUserLoginFailed(
  aUser1: PWideChar;
  aUser2: PWideChar;
  aHost: PWideChar;
  aRejectReason: UserLoginFailureCode;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeUserLoginFailed(
  const anAction: HGscAction;
  out aUser1: PWideChar;
  out aUser2: PWideChar;
  out aHost: PWideChar;
  out aRejectReason: UserLoginFailureCode;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

19.40 User logout

UserLogout (User1, *User2*, Host, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *This action will be fired when the user has disconnected from the system.*

Code : ac_UserLogout (5)

Class : ak_System (0)

Parameters :

User1 (first user) [UserName] :
Type : widestring
Description : Name of the user disconnected from the system.

User2 (second user) [optional] [UserName] :
Type : widestring
Description : Name of the second user by four eyes authentication.

Host (remote host) :
Type : widestring
Description : Host from where the connection was done.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
UserLogout ("first user", User2:  "second user", "remote
host", ClientHost:  "client host", ClientType:  0,
ClientAccount:  "client account")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateUserLogout(
  const wchar_t* aUser1,
  const wchar_t* aUser2,
  const wchar_t* aHost,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeUserLogout(
  ConstHGscAction anAction,
  const wchar_t*& aUser1,
  const wchar_t*& aUser2,
  const wchar_t*& aHost,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateUserLogout(
  aUser1: PWideChar;
  aUser2: PWideChar;
  aHost: PWideChar;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeUserLogout(
  const anAction: HGscAction;
  out aUser1: PWideChar;
  out aUser2: PWideChar;
  out aHost: PWideChar;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20 Video control

All actions to control the video streams, also all notifications about the state change of the video streams.

20.1 Activate external process

ActivateExternalProcess (Channel, TimeStamp, ExternalSystem)

Description : *Activate external process.*

Code : ac_ActivateExternalProcess (370)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ExternalSystem (external system) :
Type : VideoSensorKind
Description : External system to activate.

Text

```
ActivateExternalProcess (32, "2013/09/05 14:59:59,999 GMT+02:00",
0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateActivateExternalProcess(
  const TPlcMediaChannelID& aChannel,
  const TGLibDateTime& aTimeStamp,
  const VideoSensorKind aExternalSystem);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeActivateExternalProcess(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TGLibDateTime& aTimeStamp,
  VideoSensorKind& aExternalSystem);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateActivateExternalProcess(
  var aChannel: TPlcMediaChannelID;
  var aTimeStamp: TGLibDateTime;
  aExternalSystem: VideoSensorKind)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeActivateExternalProcess(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aTimeStamp: TGLibDateTime;
  out aExternalSystem: VideoSensorKind)
  : Boolean; stdcall; external GscActionsDll;
```

20.2 CPA measurement

CPAMeasurement (Channel, Correlation)

Description : *CPA measurement.*

Code : ac_CPAMeasurement (70)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Correlation (correlation) :
Type : __int32
Description : Correlation factor.

Text

```
CPAMeasurement (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateCPAMeasurement(
  const TPlcMediaChannelID& aChannel,
  const __int32 aCorrelation);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeCPAMeasurement(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  __int32& aCorrelation);
```

Delphi – Create

```
function GscAct_CreateCPAMeasurement(
  var aChannel: TPlcMediaChannelID;
  aCorrelation: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeCPAMeasurement(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aCorrelation: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

20.3 Change AD parameter set

ChangeADParameterSet (Channel, ParameterSet)

Description : *This action changes the current AD parameter set of the video channel.*

Code : ac_ChangeADParameterSet (42)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

ParameterSet (AD parameter set) [ADParameterSet] :
Type : TResourceID
Description : The name of the new AD parameter set.

Text

```
ChangeADParameterSet (32, "AD parameter set")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChangeADParameterSet(
  const TPlcMediaChannelID& aChannel,
  const TPlcResourceID& aParameterSet);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChangeADParameterSet(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TPlcResourceID& aParameterSet);
```

Delphi – Create

```
function GscAct_CreateChangeADParameterSet(
  var aChannel: TPlcMediaChannelID;
  var aParameterSet: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChangeADParameterSet(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aParameterSet: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

20.4 Change CPA parameter set

ChangeCPAParameterSet (Channel, ParameterSet)

Description : *This action changes the current CPA parameter set of the video channel.*

Code : ac_ChangeCPAParameterSet (44)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

ParameterSet (CPA parameter set) [CPAParameterSet] :
Type : TResourceID
Description : The name of the new CPA parameter set.

Text

```
ChangeCPAParameterSet (32, "CPA parameter set")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChangeCPAParameterSet(
  const TPlcMediaChannelID& aChannel,
  const TPlcResourceID& aParameterSet);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChangeCPAParameterSet(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TPlcResourceID& aParameterSet);
```

Delphi – Create

```
function GscAct_CreateChangeCPAParameterSet(
  var aChannel: TPlcMediaChannelID;
  var aParameterSet: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChangeCPAParameterSet(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aParameterSet: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

20.5 Change GTectVMX parameter set

ChangeGTectParameterSet (Channel, ParameterSet)

Description : *This action changes the current GTectVMX parameter set of the video channel.*

Code : ac_ChangeGTectParameterSet (534)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

ParameterSet (GTect parameter set) [GTectParameterSet] :
Type : TResourceID
Description : The name of the new GTect parameter set.

Text

```
ChangeGTectParameterSet (32, "GTect parameter set")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChangeGTectParameterSet(
  const TPlcMediaChannelID& aChannel,
  const TPlcResourceID& aParameterSet);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChangeGTectParameterSet(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TPlcResourceID& aParameterSet);
```

Delphi – Create

```
function GscAct_CreateChangeGTectParameterSet(
  var aChannel: TPlcMediaChannelID;
  var aParameterSet: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChangeGTectParameterSet(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aParameterSet: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

20.6 Change OBTRACK parameter set

```
ChangeObtrackParameterSet (Channel, ParameterSet)
```

Description : *This action changes the current OBTRACK parameter set of the video channel.*

Code : ac_ChangeObtrackParameterSet (90)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

ParameterSet (OBTRACK parameter set) [ObtrackParameterSet] :
Type : TResourceID
Description : The name of the new OBTRACK parameter set.

Text

```
ChangeObtrackParameterSet (32, "OBTRACK parameter set")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChangeObtrackParameterSet(
  const TPlcMediaChannelID& aChannel,
  const TPlcResourceID& aParameterSet);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChangeObtrackParameterSet(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TPlcResourceID& aParameterSet);
```

Delphi – Create

```
function GscAct_CreateChangeObtrackParameterSet(
  var aChannel: TPlcMediaChannelID;
  var aParameterSet: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChangeObtrackParameterSet(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aParameterSet: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

20.7 Change VMD parameter set

```
ChangeVMDParameterSet (Channel, ParameterSet)
```

GEUTEBRÜCK

Description : *This action changes the current VMD parameter set of the video channel.*

Code : `ac_ChangeVMDParameterSet` (43)

Class : `ak_Video` (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

ParameterSet (VMD parameter set) [VMDParameterSet] :
Type : TResourceID
Description : The name of the new VMD parameter set.

Text

```
ChangeVMDParameterSet (32, "VMD parameter set")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChangeVMDParameterSet(
  const TPlcMediaChannelID& aChannel,
  const TPlcResourceID& aParameterSet);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChangeVMDParameterSet(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TPlcResourceID& aParameterSet);
```

Delphi – Create

```
function GscAct_CreateChangeVMDParameterSet(
  var aChannel: TPlcMediaChannelID;
  var aParameterSet: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChangeVMDParameterSet(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aParameterSet: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

20.8 Change camera profile

```
ChangeCameraProfile (HardwareModule, CameraProfile)
```

Description : *This action changes the current camera profile of the hardware module.*

Code : `ac_ChangeCameraProfile` (330)

Class : `ak_Video` (1)

Parameters :

HardwareModule (hardware) [HardwareModule] :
Type : TResourceID
Description : Hardware module.

CameraProfile (profile) [CameraProfile] :
Type : TResourceID
Description : The name of the camera profile.

Text

```
ChangeCameraProfile ("hardware", "profile")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChangeCameraProfile(
  const TPlcResourceID& aHardwareModule,
  const TPlcResourceID& aCameraProfile);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChangeCameraProfile(
  ConstHGscAction anAction,
  TPlcResourceID& aHardwareModule,
  TPlcResourceID& aCameraProfile);
```

Delphi – Create

```
function GscAct_CreateChangeCameraProfile(
  var aHardwareModule: TPlcResourceID;
  var aCameraProfile: TPlcResourceID)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeChangeCameraProfile(
  const anAction: HGscAction;
  out aHardwareModule: TPlcResourceID;
  out aCameraProfile: TPlcResourceID)
  : Boolean; stdcall; external GscActionsDll;
```

20.9 Channel error

ChannelError (Channel, *SensorType*, Source, Message, *WindowsError*, *Description*, *XMLInfo*)

Description : *Notify channel error.*

Code : ac_ChannelError (182)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) [optional] :
Type : VideoSensorKind
Description : Sensor type.

Source (source subsystem) :
Type : PlcMessageSource
Description : Source of the message.

Message (message code) :
Type : PlcMessageCode
Description : Kind of the message.

WindowsError (Windows error code) [optional] [WindowsError] :
Type : __int32
Description : Optional Windows error code.

Description (description) [optional] :
Type : widestring
Description : Optional description of the message.

XMLInfo (additional info) [optional] :
Type : string
Description : Optional additional info (usually as XML string).

Text

```
ChannelError (32, SensorType:  0, 0, 0, WindowsError:
32, Description:  "description", XMLInfo:  "additional
info")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChannelError(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind* aSensorType,
  const PlcMessageSource aSource,
  const PlcMessageCode aMessage,
  const __int32* aWindowsError,
  const wchar_t* aDescription,
  const char* aXMLInfo);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChannelError(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const VideoSensorKind*& aSensorType,
  PlcMessageSource& aSource,
  PlcMessageCode& aMessage,
  const __int32*& aWindowsError,
  const wchar_t*& aDescription,
  const char*& aXMLInfo);
```

Delphi – Create

```
function GscAct_CreateChannelError(
  var aChannel: TPlcMediaChannelID;
  aSensorType: PVideoSensorKind;
  aSource: PlcMessageSource;
  aMessage: PlcMessageCode;
  aWindowsError: PInteger;
  aDescription: PWideChar;
  aXMLInfo: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChannelError(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: PVideoSensorKind;
  out aSource: PlcMessageSource;
  out aMessage: PlcMessageCode;
  out aWindowsError: PInteger;
  out aDescription: PWideChar;
  out aXMLInfo: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

20.10 Channel info

ChannelInfo (Channel, *SensorType*, Source, Message, *Description*, *XMLInfo*)

Description : *Notify channel information.*

Code : ac_ChannelInfo (180)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) [optional] :
Type : VideoSensorKind
Description : Sensor type.

Source (source subsystem) :
Type : PlcMessageSource
Description : Source of the message.

Message (message code) :
Type : PlcMessageCode
Description : Kind of the message.

Description (description) [optional] :
Type : widestring
Description : Optional description of the message.

XMLInfo (additional info) [optional] :
Type : string
Description : Optional additional info (usually as XML string).

Text

```
ChannelInfo (32, SensorType:  0, 0, 0, Description:
"description", XMLInfo:  "additional info")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChannelInfo(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind* aSensorType,
  const PlcMessageSource aSource,
  const PlcMessageCode aMessage,
  const wchar_t* aDescription,
  const char* aXMLInfo);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChannelInfo(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const VideoSensorKind*& aSensorType,
  PlcMessageSource& aSource,
  PlcMessageCode& aMessage,
  const wchar_t*& aDescription,
  const char*& aXMLInfo);
```

Delphi – Create

```
function GscAct_CreateChannelInfo(
  var aChannel: TPlcMediaChannelID;
  aSensorType: PVideoSensorKind;
  aSource: PlcMessageSource;
  aMessage: PlcMessageCode;
  aDescription: PWideChar;
  aXMLInfo: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChannelInfo(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: PVideoSensorKind;
  out aSource: PlcMessageSource;
  out aMessage: PlcMessageCode;
  out aDescription: PWideChar;
  out aXMLInfo: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.11 Channel live check

ChannelLiveCheck (Channel, SensorType, TimeStamp)

Description : *This action notifies that the channel is alive.*

Code : ac_ChannelLiveCheck (74)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Text

```
ChannelLiveCheck (32, 0, "2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChannelLiveCheck(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const TGLibDateTime& aTimeStamp);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChannelLiveCheck(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  TGLibDateTime& aTimeStamp);
```

Delphi – Create

```
function GscAct_CreateChannelLiveCheck(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  var aTimeStamp: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChannelLiveCheck(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aTimeStamp: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

20.12 Channel warning

ChannelWarning (Channel, *SensorType*, Source, Message, *WindowsError*, *Description*, *XMLInfo*)

Description : *Notify channel warning.*

Code : ac_ChannelWarning (181)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) [optional] :
Type : VideoSensorKind
Description : Sensor type.

Source (source subsystem) :
Type : PlcMessageSource
Description : Source of the message.

Message (message code) :
Type : PlcMessageCode
Description : Kind of the message.

WindowsError (Windows error code) [optional] [WindowsError] :
Type : __int32
Description : Optional Windows error code.

Description (description) [optional] :
Type : widestring
Description : Optional description of the message.

XMLInfo (additional info) [optional] :
Type : string
Description : Optional additional info (usually as XML string).

Text

```
ChannelWarning (32, SensorType:  0, 0, 0, WindowsError:
32, Description:  "description", XMLInfo:  "additional
info")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateChannelWarning(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind* aSensorType,
  const PlcMessageSource aSource,
  const PlcMessageCode aMessage,
  const __int32* aWindowsError,
  const wchar_t* aDescription,
  const char* aXMLInfo);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeChannelWarning(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const VideoSensorKind*& aSensorType,
  PlcMessageSource& aSource,
  PlcMessageCode& aMessage,
  const __int32*& aWindowsError,
  const wchar_t*& aDescription,
  const char*& aXMLInfo);
```

Delphi – Create

```
function GscAct_CreateChannelWarning(
  var aChannel: TPlcMediaChannelID;
  aSensorType: PVideoSensorKind;
  aSource: PlcMessageSource;
  aMessage: PlcMessageCode;
  aWindowsError: PInteger;
  aDescription: PWideChar;
  aXMLInfo: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeChannelWarning(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: PVideoSensorKind;
  out aSource: PlcMessageSource;
  out aMessage: PlcMessageCode;
  out aWindowsError: PInteger;
  out aDescription: PWideChar;
  out aXMLInfo: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.13 Enable client VCA

SetClientVCA (Channel, ClientVCAType, Enabled)

Description : *Enable client VCA*

Code : ac_SetClientVCA (536)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

ClientVCAType (VCA Type) :
Type : GTectClientVCAType
Description : VCA Type

Enabled (enabled) :
Type : bool
Description : Enabled

Text

```
SetClientVCA (32, 0, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetClientVCA(
  const TPlcMediaChannelID& aChannel,
  const GTectClientVCAType aClientVCAType,
  const bool aEnabled);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetClientVCA(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  GTectClientVCAType& aClientVCAType,
  bool& aEnabled);
```

Delphi – Create

```
function GscAct_CreateSetClientVCA(
  var aChannel: TPlcMediaChannelID;
  aClientVCAType: GTectClientVCAType;
  aEnabled: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetClientVCA(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aClientVCAType: GTectClientVCAType;
  out aEnabled: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

20.14 G-Tect analytics live check

```
GTectAnalyticsLiveCheck ()
```

Description : *This action will be fired periodically to ensure that analytics client is still running.*

Code : `ac_GTectAnalyticsLiveCheck` (532)

Class : `ak_Video` (1)

This action has no parameters.

Text

```
GTectAnalyticsLiveCheck ()
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGTectAnalyticsLiveCheck();
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGTectAnalyticsLiveCheck(
  ConstHGscAction anAction);
```

Delphi – Create

```
function GscAct_CreateGTectAnalyticsLiveCheck()
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGTectAnalyticsLiveCheck(
  const anAction: HGscAction)
  : Boolean; stdcall; external GscActionsDll;
```

20.15 G-Tect scene alarm

```
GTectSceneAlarm (Channel, AlarmReason)
```

Description : *This action will be fired when G-Tect Video content analysis detects a scene alarm.*

Code : `ac_GTectSceneAlarm` (528)

Class : `ak_Video` (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

AlarmReason (alarm reason) :
Type : GTectSceneAlarmReason
Description : Alarm reason.

Text

```
GTectSceneAlarm (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGTectSceneAlarm(
  const TPlcMediaChannelID& aChannel,
  const GTectSceneAlarmReason aAlarmReason);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGTectSceneAlarm(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  GTectSceneAlarmReason& aAlarmReason);
```

Delphi – Create

```
function GscAct_CreateGTectSceneAlarm(
  var aChannel: TPlcMediaChannelID;
  aAlarmReason: GTectSceneAlarmReason)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGTectSceneAlarm(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aAlarmReason: GTectSceneAlarmReason)
  : Boolean; stdcall; external GscActionsDll;
```

20.16 G-Tect scene alarm finished

```
GTectSceneAlarmFinished (Channel, AlarmReason)
```

Description : *This action will be fired when G-Tect Video content analysis scene alarm is finished.*

GEUTEBRÜCK

Code : ac_GTectSceneAlarmFinished (530)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

AlarmReason (alarm reason) :
Type : GTectSceneAlarmReason
Description : Alarm reason.

Text

```
GTectSceneAlarmFinished (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGTectSceneAlarmFinished(
  const TPlcMediaChannelID& aChannel,
  const GTectSceneAlarmReason aAlarmReason);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGTectSceneAlarmFinished(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  GTectSceneAlarmReason& aAlarmReason);
```

Delphi – Create

```
function GscAct_CreateGTectSceneAlarmFinished(
  var aChannel: TPlcMediaChannelID;
  aAlarmReason: GTectSceneAlarmReason)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGTectSceneAlarmFinished(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aAlarmReason: GTectSceneAlarmReason)
  : Boolean; stdcall; external GscActionsDll;
```

20.17 G-Tect/Dual sensor alarm

```
GTectDualSensorAlarm (Channel, SterileZoneNumber,
VirtualFenceNumber, VMDGroup, VMDZone, VMDCycle, AlarmArea)
```

Description : *This action will be fired when G-Tect/Dual sensor detects an alarm.*

Code : `ac_GTectDualSensorAlarm` (524)

Class : `ak_Video` (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SterileZoneNumber (sterile zone number) [optional] :
Type : __int32
Description : Sterile Zone Number

VirtualFenceNumber (virtual fence number) [optional] :
Type : __int32
Description : Virtual Fence Number.

VMDGroup (VMD alarm group) [optional] :
Type : VMDGroup
Description : VMD alarm group.

VMDZone (VMD zone) [optional] :
Type : __int32
Description : VMD zone nr.

VMDCycle (VMD cycle) [optional] :
Type : VMDCycle
Description : VMD measure cycle.

AlarmArea (alarm area) [optional] :
Type : TPlcRect
Description : Alarm area.

Text

```
GTectDualSensorAlarm (32, SterileZoneNumber: 32,
VirtualFenceNumber: 32, VMDGroup: 0, VMDZone: 32,
VMDCycle: 0, AlarmArea: { 0, 0, 0, 0 })
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGTectDualSensorAlarm(
  const TPlcMediaChannelID& aChannel,
  const __int32* aSterileZoneNumber,
  const __int32* aVirtualFenceNumber,
  const VMDGroup* aVMDGroup,
  const __int32* aVMDZone,
  const VMDCycle* aVMDCycle,
  const TPlcRect* aAlarmArea);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGTectDualSensorAlarm(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const __int32*& aSterileZoneNumber,
  const __int32*& aVirtualFenceNumber,
  const VMDGroup*& aVMDGroup,
  const __int32*& aVMDZone,
  const VMDCycle*& aVMDCycle,
  const TPlcRect*& aAlarmArea);
```

Delphi – Create

```
function GscAct_CreateGTectDualSensorAlarm(
  var aChannel: TPlcMediaChannelID;
  aSterileZoneNumber: PInteger;
  aVirtualFenceNumber: PInteger;
  aVMDGroup: PVMDGroup;
  aVMDZone: PInteger;
  aVMDCycle: PVMDCycle;
  aAlarmArea: PTPlcRect)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGTectDualSensorAlarm(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSterileZoneNumber: PInteger;
  out aVirtualFenceNumber: PInteger;
  out aVMDGroup: PVMDGroup;
  out aVMDZone: PInteger;
  out aVMDCycle: PVMDCycle;
  out aAlarmArea: PTPlcRect)
  : Boolean; stdcall; external GscActionsDll;
```

20.18 G-Tect/Dual sensor alarm finished

GTectDualSensorAlarmFinished (Channel)

Description : *This action will be fired when the alarm is finished.*

Code : ac_GTectDualSensorAlarmFinished (526)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
GTectDualSensorAlarmFinished (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGTectDualSensorAlarmFinished(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGTectDualSensorAlarmFinished(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateGTectDualSensorAlarmFinished(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGTectDualSensorAlarmFinished(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

20.19 G-Tect/VMX alarm

GTectVMXAlarm (Channel, *SterileZoneNumber*, *VirtualFenceNumber*, *AlarmGroup*, *TrackID*, *AlarmArea*)

GEUTEBRÜCK

Description : *This action will be fired when G-Tect/VMX Video content analysis detects an alarm.*

Code : `ac_GTectVMXAlarm` (520)

Class : `ak_Video` (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SterileZoneNumber (sterile zone number) [optional] :
Type : __int32
Description : Sterile Zone Number

VirtualFenceNumber (virtual fence number) [optional] :
Type : __int32
Description : Virtual Fence Number.

AlarmGroup (group zones or fences) [optional] :
Type : __int32
Description : Used to group different Zones or fences to the same level.

TrackID (internal track ID) [optional] :
Type : __int32
Description : Internal Track ID.

AlarmArea (alarm area) [optional] :
Type : TPlcRect
Description : Alarm area.

Text

```
GTectVMXAlarm (32, SterileZoneNumber:  32,
VirtualFenceNumber:  32, AlarmGroup:  32, TrackID: 32,
AlarmArea:  { 0, 0, 0, 0 })
```

C++ - Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGTectVMXAlarm(
  const TPlcMediaChannelID& aChannel,
  const __int32* aSterileZoneNumber,
  const __int32* aVirtualFenceNumber,
  const __int32* aAlarmGroup,
  const __int32* aTrackID,
  const TPlcRect* aAlarmArea);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGTectVMXAlarm(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  const __int32*& aSterileZoneNumber,
  const __int32*& aVirtualFenceNumber,
  const __int32*& aAlarmGroup,
  const __int32*& aTrackID,
  const TPlcRect*& aAlarmArea);
```

Delphi – Create

```
function GscAct_CreateGTectVMXAlarm(
  var aChannel: TPlcMediaChannelID;
  aSterileZoneNumber: PInteger;
  aVirtualFenceNumber: PInteger;
  aAlarmGroup: PInteger;
  aTrackID: PInteger;
  aAlarmArea: PTPlcRect)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGTectVMXAlarm(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSterileZoneNumber: PInteger;
  out aVirtualFenceNumber: PInteger;
  out aAlarmGroup: PInteger;
  out aTrackID: PInteger;
  out aAlarmArea: PTPlcRect)
  : Boolean; stdcall; external GscActionsDll;
```

20.20 G-Tect/VMX alarm finished

GTectVMXAlarmFinished (Channel)

Description : *This action will be fired when the alarm is finished.*

Code : ac_GTectVMXAlarmFinished (522)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

GEUTEBRÜCK

Text

```
GTectVMXAlarmFinished (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateGTectVMXAlarmFinished(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeGTectVMXAlarmFinished(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateGTectVMXAlarmFinished(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeGTectVMXAlarmFinished(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

20.21 IAS settings changed

IASSettingsChanged (Channel, SensorType)

Description : *IAS settings changed.*

Code : ac_IASSettingsChanged (49)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Text

```
IASSettingsChanged (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIASSettingsChanged(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIASSettingsChanged(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType);
```

Delphi – Create

```
function GscAct_CreateIASSettingsChanged(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIASSettingsChanged(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind)
  : Boolean; stdcall; external GscActionsDll;
```

20.22 IP camera failover notification

IPCamFailoverNotification (PrimaryServer, SecondaryServer, PrimaryChannel, SecondaryChannel, *CameraParameters*)

Description : *IP camera failover notification.*

Code : ac_IPCamFailoverNotification (540)

Class : ak_Video (1)

Parameters :

PrimaryServer (primary server) :
Type : widestring
Description : The name of the primary server.

SecondaryServer (secondary server) :
Type : widestring
Description : The name of the secondary server.

PrimaryChannel (primary channel) :
Type : TMediaChannelID
Description : Primary channel.

SecondaryChannel (secondary channel) [VideoInput] :
Type : TMediaChannelID
Description : Secondary channel.

CameraParameters (camera parameters) [optional] :
Type : widestring
Description : Optional camera parameters.

Text

```
IPCamFailoverNotification ("primary server", "secondary
server", 32, 32, CameraParameters:  "camera parameters")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIPCamFailoverNotification(
  const wchar_t* aPrimaryServer,
  const wchar_t* aSecondaryServer,
  const TPlcMediaChannelID& aPrimaryChannel,
  const TPlcMediaChannelID& aSecondaryChannel,
  const wchar_t* aCameraParameters);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIPCamFailoverNotification(
  ConstHGscAction anAction,
  const wchar_t*& aPrimaryServer,
  const wchar_t*& aSecondaryServer,
  TPlcMediaChannelID& aPrimaryChannel,
  TPlcMediaChannelID& aSecondaryChannel,
  const wchar_t*& aCameraParameters);
```

Delphi – Create

```
function GscAct_CreateIPCamFailoverNotification(
  aPrimaryServer: PWideChar;
  aSecondaryServer: PWideChar;
  var aPrimaryChannel: TPlcMediaChannelID;
  var aSecondaryChannel: TPlcMediaChannelID;
  aCameraParameters: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIPCamFailoverNotification(
  const anAction: HGscAction;
  out aPrimaryServer: PWideChar;
  out aSecondaryServer: PWideChar;
  out aPrimaryChannel: TPlcMediaChannelID;
  out aSecondaryChannel: TPlcMediaChannelID;
  out aCameraParameters: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.23 IP camera failover restore

IPCamFailoverRestore (PrimaryServer, SecondaryServer, PrimaryChannel, SecondaryChannel, *CameraParameters*)

Description : *IP camera failover restore.*

Code : ac_IPCamFailoverRestore (541)

Class : ak_Video (1)

Parameters :

PrimaryServer (primary server) :
Type : widestring
Description : The name of the primary server.

SecondaryServer (secondary server) :
Type : widestring
Description : The name of the secondary server.

PrimaryChannel (primary channel) :
Type : TMediaChannelID
Description : Primary channel.

SecondaryChannel (secondary channel) [VideoInput] :
Type : TMediaChannelID
Description : Secondary channel.

CameraParameters (camera parameters) [optional] :
Type : widestring
Description : Optional camera parameters.

Text

```
IPCamFailoverRestore ("primary server", "secondary
server", 32, 32, CameraParameters:  "camera parameters")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIPCamFailoverRestore(
  const wchar_t* aPrimaryServer,
  const wchar_t* aSecondaryServer,
  const TPlcMediaChannelID& aPrimaryChannel,
  const TPlcMediaChannelID& aSecondaryChannel,
  const wchar_t* aCameraParameters);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIPCamFailoverRestore(
  ConstHGscAction anAction,
  const wchar_t*& aPrimaryServer,
  const wchar_t*& aSecondaryServer,
  TPlcMediaChannelID& aPrimaryChannel,
  TPlcMediaChannelID& aSecondaryChannel,
  const wchar_t*& aCameraParameters);
```

Delphi – Create

```
function GscAct_CreateIPCamFailoverRestore(
  aPrimaryServer: PWideChar;
  aSecondaryServer: PWideChar;
  var aPrimaryChannel: TPlcMediaChannelID;
  var aSecondaryChannel: TPlcMediaChannelID;
  aCameraParameters: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeIPCamFailoverRestore(
  const anAction: HGscAction;
  out aPrimaryServer: PWideChar;
  out aSecondaryServer: PWideChar;
  out aPrimaryChannel: TPlcMediaChannelID;
  out aSecondaryChannel: TPlcMediaChannelID;
  out aCameraParameters: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.24 IP camera raw command

IPCameraRawCommand (URL, *User*, *Password*, *POST*)

Description : *This action sends a special command to the IP camera.*

Code : ac_IPCameraRawCommand (75)

Class : ak_Video (1)

Parameters :

URL (url) :
Type : widestring
Description : Complete command URL (like http://192.168.0.165:80/set?daynight=night).

User (user) [optional] :
Type : widestring
Description : User name to authenticate by the camera (optional).

Password (password) [optional] :
Type : widestring
Description : Password to authenticate by the camera (optional).

POST (post) [optional] :
Type : widestring
Description : POST parameters (optional, separate lines with
r
n).

Text

```
IPCameraRawCommand ("url", User:  "user", Password:
"password", POST: "post")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateIPCameraRawCommand(
  const wchar_t* aURL,
  const wchar_t* aUser,
  const wchar_t* aPassword,
  const wchar_t* aPOST);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeIPCameraRawCommand(
  ConstHGscAction anAction,
  const wchar_t*& aURL,
  const wchar_t*& aUser,
  const wchar_t*& aPassword,
  const wchar_t*& aPOST);
```

Delphi – Create

```
function GscAct_CreateIPCameraRawCommand(
  aURL: PWideChar;
  aUser: PWideChar;
  aPassword: PWideChar;
  aPOST: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeIPCameraRawCommand(
  const anAction: HGscAction;
  out aURL: PWideChar;
  out aUser: PWideChar;
  out aPassword: PWideChar;
  out aPOST: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.25 Make CPA reference image

MakeCPAReferenceImage (Channel)

Description : *Make CPA reference image.*

Code : ac_MakeCPAReferenceImage (48)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
MakeCPAReferenceImage (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateMakeCPAReferenceImage(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeMakeCPAReferenceImage(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateMakeCPAReferenceImage(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeMakeCPAReferenceImage(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

20.26 Media channel setup

MediaChannelSetupInfo (Channel, TimeStamp, Parameter)

Description : *Media channel setup info.*

Code : ac_MediaChannelSetupInfo (395)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Parameter (parameter) :
Type : widestring
Description : Parameter.

Text

```
MediaChannelSetupInfo (32, "2013/09/05 14:59:59,999 GMT+02:00",
"parameter")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateMediaChannelSetupInfo(
  const TPlcMediaChannelID& aChannel,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aParameter);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeMediaChannelSetupInfo(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aParameter);
```

Delphi – Create

```
function GscAct_CreateMediaChannelSetupInfo(
  var aChannel: TPlcMediaChannelID;
  var aTimeStamp: TGLibDateTime;
  aParameter: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeMediaChannelSetupInfo(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aTimeStamp: TGLibDateTime;
  out aParameter: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.27 NPR raw data

NPRRawData (PlateNo, Country, Channel, TimeStamp, ZoneRect, Weight, ZoneState, ZonePlace, Speed, Direction, ZoneIndex, CurBest, PlateWidth, PlateHeight, PlateAngle, SymHeight, Type)

Description : *NPR raw data.*

Code : ac_NPRRawData (371)

Class : ak_Video (1)

Parameters :

PlateNo (plate no.) :
Type : widestring
Description : Recognized plate no.

Country (country) :
Type : widestring
Description : Country.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ZoneRect (zone rect) :
Type : TPlcRect
Description : Zone rectangle.

Weight (weight) :
Type : __int32
Description : Weight of recognition.

ZoneState (zone state) :
Type : __int32
Description : Zone state.

ZonePlace (zone status) :
Type : __int32
Description : Zone status.

Speed (speed) :
Type : __int32
Description : Speed in km/h

Direction (direction) :
Type : __int32
Description : Direction of the motion.

ZoneIndex (zone index) :
Type : __int32
Description : Zone index.

CurBest (best) :
Type : bool
Description : Current recognition is best.

PlateWidth (plate width) :
Type : __int32
Description : Plate width.

PlateHeight (plate height) :
Type : __int32
Description : Plate height.

PlateAngle (plate angle) :
Type : __int32
Description : Plate angle.

SymHeight (symbol height) :
Type : __int32
Description : Symbol height.

Type (type) :
Type : widestring
Description : Number type.

Text

```
NPRRawData ("plate no.", "country", 32,
"2013/09/05 14:59:59,999 GMT+02:00", { 0, 0, 0, 0 }, 32,
32, 32, 32, 32, 32, 1, 32, 32, 32, 32, "type")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateNPRRawData(
  const wchar_t* aPlateNo,
  const wchar_t* aCountry,
  const TPlcMediaChannelID& aChannel,
  const TGLibDateTime& aTimeStamp,
  const TPlcRect& aZoneRect,
  const __int32 aWeight,
  const __int32 aZoneState,
  const __int32 aZonePlace,
  const __int32 aSpeed,
  const __int32 aDirection,
  const __int32 aZoneIndex,
  const bool aCurBest,
  const __int32 aPlateWidth,
  const __int32 aPlateHeight,
  const __int32 aPlateAngle,
  const __int32 aSymHeight,
  const wchar_t* aType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeNPRRawData(
  ConstHGscAction anAction,
  const wchar_t*& aPlateNo,
  const wchar_t*& aCountry,
  TPlcMediaChannelID& aChannel,
  TGLibDateTime& aTimeStamp,
  TPlcRect& aZoneRect,
  __int32& aWeight,
  __int32& aZoneState,
  __int32& aZonePlace,
  __int32& aSpeed,
  __int32& aDirection,
  __int32& aZoneIndex,
  bool& aCurBest,
  __int32& aPlateWidth,
  __int32& aPlateHeight,
  __int32& aPlateAngle,
  __int32& aSymHeight,
  const wchar_t*& aType);
```

Delphi – Create

```
function GscAct_CreateNPRRawData(
  aPlateNo: PWideChar;
  aCountry: PWideChar;
  var aChannel: TPlcMediaChannelID;
  var aTimeStamp: TGLibDateTime;
  var aZoneRect: TPlcRect;
  aWeight: Integer;
  aZoneState: Integer;
  aZonePlace: Integer;
  aSpeed: Integer;
  aDirection: Integer;
  aZoneIndex: Integer;
  aCurBest: Boolean;
  aPlateWidth: Integer;
  aPlateHeight: Integer;
  aPlateAngle: Integer;
  aSymHeight: Integer;
  aType: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeNPRRawData(
  const anAction: HGscAction;
  out aPlateNo: PWideChar;
  out aCountry: PWideChar;
  out aChannel: TPlcMediaChannelID;
  out aTimeStamp: TGLibDateTime;
  out aZoneRect: TPlcRect;
  out aWeight: Integer;
  out aZoneState: Integer;
  out aZonePlace: Integer;
  out aSpeed: Integer;
  out aDirection: Integer;
  out aZoneIndex: Integer;
  out aCurBest: Boolean;
  out aPlateWidth: Integer;
  out aPlateHeight: Integer;
  out aPlateAngle: Integer;
  out aSymHeight: Integer;
  out aType: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.28 NPR recognition

NPRRecognition (PlateNo, Country, Channel, TimeStamp, ZoneRect, Restriction, Category, Speed, Direction, ZoneIndex, Type, Weight)

Description : *NPR recognition.*

Code : `ac_NPRRecognition` (375)

GEUTEBRÜCK

Class : `ak_Video` (1)

Parameters :

PlateNo (plate no.) :
Type : widestring
Description : Recognized plate no.

Country (country) :
Type : widestring
Description : Country.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ZoneRect (zone rect) :
Type : TPlcRect
Description : Zone rectangle.

Restriction (restriction) :
Type : PlcNPRRestriction
Description : Restriction of recognized number.

Category (category) :
Type : widestring
Description : Category of recognized number.

Speed (speed) :
Type : __int32
Description : Speed in km/h

Direction (direction) :
Type : __int32
Description : Direction of the motion.

ZoneIndex (zone index) :
Type : __int32
Description : Zone index.

Type (type) :
Type : widestring
Description : Number type.

Weight (weight) :
Type : __int32
Description : Weight of recognition.

Text

```
NPRRecognition ("plate no.", "country", 32,
"2013/09/05 14:59:59,999 GMT+02:00", { 0, 0, 0, 0 }, 0,
"category", 32, 32, 32, "type", 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateNPRRecognition(
  const wchar_t* aPlateNo,
  const wchar_t* aCountry,
  const TPlcMediaChannelID& aChannel,
  const TGLibDateTime& aTimeStamp,
  const TPlcRect& aZoneRect,
  const PlcNPRRestriction aRestriction,
  const wchar_t* aCategory,
  const __int32 aSpeed,
  const __int32 aDirection,
  const __int32 aZoneIndex,
  const wchar_t* aType,
  const __int32 aWeight);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeNPRRecognition(
  ConstHGscAction anAction,
  const wchar_t*& aPlateNo,
  const wchar_t*& aCountry,
  TPlcMediaChannelID& aChannel,
  TGLibDateTime& aTimeStamp,
  TPlcRect& aZoneRect,
  PlcNPRRestriction& aRestriction,
  const wchar_t*& aCategory,
  __int32& aSpeed,
  __int32& aDirection,
  __int32& aZoneIndex,
  const wchar_t*& aType,
  __int32& aWeight);
```

Delphi – Create

```
function GscAct_CreateNPRRecognition(
  aPlateNo: PWideChar;
  aCountry: PWideChar;
  var aChannel: TPlcMediaChannelID;
  var aTimeStamp: TGLibDateTime;
  var aZoneRect: TPlcRect;
  aRestriction: PlcNPRRestriction;
  aCategory: PWideChar;
  aSpeed: Integer;
  aDirection: Integer;
  aZoneIndex: Integer;
  aType: PWideChar;
  aWeight: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeNPRRecognition(
  const anAction: HGscAction;
  out aPlateNo: PWideChar;
  out aCountry: PWideChar;
  out aChannel: TPlcMediaChannelID;
  out aTimeStamp: TGLibDateTime;
  out aZoneRect: TPlcRect;
  out aRestriction: PlcNPRRestriction;
  out aCategory: PWideChar;
  out aSpeed: Integer;
  out aDirection: Integer;
  out aZoneIndex: Integer;
  out aType: PWideChar;
  out aWeight: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

20.29 OBTRACK channel counter

ObtrackChannelCounter (Channel, CounterType, CounterValue, ObjectDirection, TimeStamp, *ResetTimeStamp*)

Description : *OBTRACK channel counter.*

Code : ac_ObtrackChannelCounter (94)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

CounterType (counter type) :
Type : PlcObtrackCounterType
Description : Counter type.

CounterValue (counter value) :
Type : __int64
Description : Counter value.

ObjectDirection (object direction) :
Type : __int32
Description : Object direction.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ResetTimeStamp (reset time stamp) [optional] :
Type : TGLibDateTime
Description : Reset time stamp.

Text

```
ObtrackChannelCounter (32, 0, 64, 32,
"2013/09/05 14:59:59,999 GMT+02:00", ResetTimeStamp:
"2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackChannelCounter(
  const TPlcMediaChannelID& aChannel,
  const PlcObtrackCounterType aCounterType,
  const __int64& aCounterValue,
  const __int32 aObjectDirection,
  const TGLibDateTime& aTimeStamp,
  const TGLibDateTime* aResetTimeStamp);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackChannelCounter(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  PlcObtrackCounterType& aCounterType,
  __int64& aCounterValue,
  __int32& aObjectDirection,
  TGLibDateTime& aTimeStamp,
  const TGLibDateTime*& aResetTimeStamp);
```

Delphi – Create

```
function GscAct_CreateObtrackChannelCounter(
  var aChannel: TPlcMediaChannelID;
  aCounterType: PlcObtrackCounterType;
  var aCounterValue: Int64;
  aObjectDirection: Integer;
  var aTimeStamp: TGLibDateTime;
  aResetTimeStamp: PTGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackChannelCounter(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aCounterType: PlcObtrackCounterType;
  out aCounterValue: Int64;
  out aObjectDirection: Integer;
  out aTimeStamp: TGLibDateTime;
  out aResetTimeStamp: PTGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

20.30 OBTRACK channel counter threshold

ObtrackChannelCounterThreshold (Channel, CounterType, CounterValue, ExceedingDirection, TimeStamp)

Description : *OBTRACK channel counter threshold.*

Code : ac_ObtrackChannelCounterThreshold (96)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

CounterType (counter type) :
Type : PlcObtrackCounterType
Description : Counter type.

CounterValue (counter value) :
Type : __int64
Description : Counter value.

ExceedingDirection (exceeding direction) :
Type : PlcObtrackExceedingDirection
Description : Exceeding direction.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Text

```
ObtrackChannelCounterThreshold (32, 0, 64, 0,
"2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackChannelCounterThreshold(
  const TPlcMediaChannelID& aChannel,
  const PlcObtrackCounterType aCounterType,
  const __int64& aCounterValue,
  const PlcObtrackExceedingDirection aExceedingDirection,
  const TGLibDateTime& aTimeStamp);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackChannelCounterThreshold(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  PlcObtrackCounterType& aCounterType,
  __int64& aCounterValue,
  PlcObtrackExceedingDirection& aExceedingDirection,
  TGLibDateTime& aTimeStamp);
```

Delphi – Create

```
function GscAct_CreateObtrackChannelCounterThreshold(
  var aChannel: TPlcMediaChannelID;
  aCounterType: PlcObtrackCounterType;
  var aCounterValue: Int64;
  aExceedingDirection: PlcObtrackExceedingDirection;
  var aTimeStamp: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackChannelCounterThreshold(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aCounterType: PlcObtrackCounterType;
  out aCounterValue: Int64;
  out aExceedingDirection: PlcObtrackExceedingDirection;
  out aTimeStamp: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

20.31 OBTRACK channel set counter

ObtrackChannelSetCounter (Channel, CounterType, CounterValue, TimeStamp)

Description : *OBTRACK channel set counter.*

Code : ac_ObtrackChannelSetCounter (98)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

CounterType (counter type) :
Type : PlcObtrackCounterType
Description : Counter type.

CounterValue (counter value) :
Type : __int64
Description : Counter value.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Text

```
ObtrackChannelSetCounter (32, 0, 64,
"2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackChannelSetCounter(
  const TPlcMediaChannelID& aChannel,
  const PlcObtrackCounterType aCounterType,
  const __int64& aCounterValue,
  const TGLibDateTime& aTimeStamp);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackChannelSetCounter(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  PlcObtrackCounterType& aCounterType,
  __int64& aCounterValue,
  TGLibDateTime& aTimeStamp);
```

Delphi – Create

```
function GscAct_CreateObtrackChannelSetCounter(
  var aChannel: TPlcMediaChannelID;
  aCounterType: PlcObtrackCounterType;
  var aCounterValue: Int64;
  var aTimeStamp: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackChannelSetCounter(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aCounterType: PlcObtrackCounterType;
  out aCounterValue: Int64;
  out aTimeStamp: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

20.32 OBTRACK frame raw data

ObtrackFrameRawData (TimeStamp, Channel, Brightness, Contrast)

Description : *OBTRACK frame raw data.*

Code : ac_ObtrackFrameRawData (91)

Class : ak_Video (1)

Parameters :

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Brightness (brightness) :
Type : __int32
Description : Brightness.

Contrast (contrast) :
Type : __int32
Description : Contrast.

Text

```
ObtrackFrameRawData ("2013/09/05 14:59:59,999 GMT+02:00",
32, 32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateObtrackFrameRawData(
  const TGLibDateTime& aTimeStamp,
  const TPlcMediaChannelID& aChannel,
  const __int32 aBrightness,
  const __int32 aContrast);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackFrameRawData(
  ConstHGscAction anAction,
  TGLibDateTime& aTimeStamp,
  TPlcMediaChannelID& aChannel,
  __int32& aBrightness,
  __int32& aContrast);
```

Delphi – Create

```
function GscAct_CreateObtrackFrameRawData(
  var aTimeStamp: TGLibDateTime;
  var aChannel: TPlcMediaChannelID;
  aBrightness: Integer;
  aContrast: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackFrameRawData(
  const anAction: HGscAction;
  out aTimeStamp: TGLibDateTime;
  out aChannel: TPlcMediaChannelID;
  out aBrightness: Integer;
  out aContrast: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

20.33 OBTRACK group counter

ObtrackGroupCounter (GroupId, CounterType, CounterValue, ObjectDirection, TimeStamp, *ResetTimeStamp*, *GroupName*)

Description : *OBTRACK group counter.*

Code : ac_ObtrackGroupCounter (95)

Class : ak_Video (1)

Parameters :

GroupId (group ID) :
Type : __int32
Description : Group ID.

CounterType (counter type) :
Type : PlcObtrackCounterType
Description : Counter type.

CounterValue (counter value) :
Type : __int64
Description : Counter value.

ObjectDirection (object direction) :
Type : __int32
Description : Object direction.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ResetTimeStamp (reset time stamp) [optional] :
Type : TGLibDateTime
Description : Reset time stamp.

GroupName (group name) [optional] :
Type : widestring
Description : Group name.

Text

```
ObtrackGroupCounter (32, 0, 64, 32, "2013/09/05 14:59:59,999 GMT+02:00",
ResetTimeStamp:  "2013/09/05 14:59:59,999 GMT+02:00",
GroupName:  "group name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackGroupCounter(
  const __int32 aGroupId,
  const PlcObtrackCounterType aCounterType,
  const __int64& aCounterValue,
  const __int32 aObjectDirection,
  const TGLibDateTime& aTimeStamp,
  const TGLibDateTime* aResetTimeStamp,
  const wchar_t* aGroupName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackGroupCounter(
  ConstHGscAction anAction,
  __int32& aGroupId,
  PlcObtrackCounterType& aCounterType,
  __int64& aCounterValue,
  __int32& aObjectDirection,
  TGLibDateTime& aTimeStamp,
  const TGLibDateTime*& aResetTimeStamp,
  const wchar_t*& aGroupName);
```

Delphi – Create

```
function GscAct_CreateObtrackGroupCounter(
  aGroupId: Integer;
  aCounterType: PlcObtrackCounterType;
  var aCounterValue: Int64;
  aObjectDirection: Integer;
  var aTimeStamp: TGLibDateTime;
  aResetTimeStamp: PTGLibDateTime;
  aGroupName: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeObtrackGroupCounter(
  const anAction: HGscAction;
  out aGroupId: Integer;
  out aCounterType: PlcObtrackCounterType;
  out aCounterValue: Int64;
  out aObjectDirection: Integer;
  out aTimeStamp: TGLibDateTime;
  out aResetTimeStamp: PTGLibDateTime;
  out aGroupName: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.34 OBTRACK group counter threshold

ObtrackGroupCounterThreshold (GroupId, CounterType, CounterValue, ExceedingDirection, TimeStamp, *GroupName*)

Description : *OBTRACK group counter threshold.*

Code : ac_ObtrackGroupCounterThreshold (97)

Class : ak_Video (1)

Parameters :

GroupId (group ID) :
Type : __int32
Description : Group ID.

CounterType (counter type) :
Type : PlcObtrackCounterType
Description : Counter type.

CounterValue (counter value) :
Type : __int64
Description : Counter value.

ExceedingDirection (exceeding direction) :
Type : PlcObtrackExceedingDirection
Description : Exceeding direction.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

GroupName (group name) [optional] :
Type : widestring
Description : Group name.

Text

```
ObtrackGroupCounterThreshold (32, 0, 64, 0,
"2013/09/05 14:59:59,999 GMT+02:00", GroupName:  "group
name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackGroupCounterThreshold(
  const __int32 aGroupId,
  const PlcObtrackCounterType aCounterType,
  const __int64& aCounterValue,
  const PlcObtrackExceedingDirection aExceedingDirection,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aGroupName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackGroupCounterThreshold(
  ConstHGscAction anAction,
  __int32& aGroupId,
  PlcObtrackCounterType& aCounterType,
  __int64& aCounterValue,
  PlcObtrackExceedingDirection& aExceedingDirection,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aGroupName);
```

Delphi – Create

```
function GscAct_CreateObtrackGroupCounterThreshold(
  aGroupId: Integer;
  aCounterType: PlcObtrackCounterType;
  var aCounterValue: Int64;
  aExceedingDirection: PlcObtrackExceedingDirection;
  var aTimeStamp: TGLibDateTime;
  aGroupName: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackGroupCounterThreshold(
  const anAction: HGscAction;
  out aGroupId: Integer;
  out aCounterType: PlcObtrackCounterType;
  out aCounterValue: Int64;
  out aExceedingDirection: PlcObtrackExceedingDirection;
  out aTimeStamp: TGLibDateTime;
  out aGroupName: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.35 OBTRACK group set counter

ObtrackGroupSetCounter (GroupId, CounterType, CounterValue, TimeStamp, *GroupName*)

GEUTEBRÜCK

Description : *OBTRACK group set counter.*

Code : `ac_ObtrackGroupSetCounter` (99)

Class : `ak_Video` (1)

Parameters :

GroupId (group ID) :
Type : __int32
Description : Group ID.

CounterType (counter type) :
Type : PlcObtrackCounterType
Description : Counter type.

CounterValue (counter value) :
Type : __int64
Description : Counter value.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

GroupName (group name) [optional] :
Type : widestring
Description : Group name.

Text

```
ObtrackGroupSetCounter (32, 0, 64, "2013/09/05 14:59:59,999 GMT+02:00",
GroupName:  "group name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackGroupSetCounter(
  const __int32 aGroupId,
  const PlcObtrackCounterType aCounterType,
  const __int64& aCounterValue,
  const TGLibDateTime& aTimeStamp,
  const wchar_t* aGroupName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackGroupSetCounter(
  ConstHGscAction anAction,
  __int32& aGroupId,
  PlcObtrackCounterType& aCounterType,
  __int64& aCounterValue,
  TGLibDateTime& aTimeStamp,
  const wchar_t*& aGroupName);
```

Delphi – Create

```
function GscAct_CreateObtrackGroupSetCounter(
  aGroupId: Integer;
  aCounterType: PlcObtrackCounterType;
  var aCounterValue: Int64;
  var aTimeStamp: TGLibDateTime;
  aGroupName: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackGroupSetCounter(
  const anAction: HGscAction;
  out aGroupId: Integer;
  out aCounterType: PlcObtrackCounterType;
  out aCounterValue: Int64;
  out aTimeStamp: TGLibDateTime;
  out aGroupName: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.36 OBTRACK object raw data

ObtrackObjectRawData (TimeStamp, Channel, Area, ObjectID, ObjectStatus, ObjectClass, *Confidence*, *Position*, *Speed*, *Duration*, *Direction*, *Size*, *ObjectWidth*, *ObjectHeight*, *ProcessSize*, *GscNetName*)

Description : *OBTRACK object raw data.*

Code : ac_ObtrackObjectRawData (92)

Class : ak_Video (1)

Parameters :

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Area (area no) :
Type : __int32
Description : Area no.

ObjectID (object ID) :
Type : __int32
Description : Object ID.

ObjectStatus (object status) :
Type : PlcObtrackObjectStatus
Description : Object status.

GEUTEBRÜCK

ObjectClass (object class) :
Type : PlcObtrackObjectClass
Description : Object class.

Confidence (confidence) [optional] :
Type : __int32
Description : Confidence.

Position (position) [optional] :
Type : TPlcRect
Description : Position.

Speed (speed) [optional] :
Type : double
Description : Speed.

Duration (duration) [optional] :
Type : __int32
Description : Duration.

Direction (direction) [optional] :
Type : __int32
Description : Direction.

Size (object size) [optional] :
Type : double
Description : Object size.

ObjectWidth (object width) [optional] :
Type : double
Description : Object width in meters.

ObjectHeight (object height) [optional] :
Type : double
Description : Object height in meters.

ProcessSize (process size) [optional] :
Type : PlcObtrackProcessSize
Description : Process size.

GscNetName (GSC net name) [optional] :
Type : string
Description : GeViScope network name.

Text

```
ObtrackObjectRawData ("2013/09/05 14:59:59,999 GMT+02:00",
32, 32, 32, 0, 0, Confidence: 32, Position: { 0, 0, 0,
0 }, Speed: 0.0, Duration: 32, Direction: 32, Size:
0.0, ObjectWidth: 0.0, ObjectHeight: 0.0, ProcessSize:
0, GscNetName: "GSC net name")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackObjectRawData(
  const TGLibDateTime& aTimeStamp,
  const TPlcMediaChannelID& aChannel,
  const __int32 aArea,
  const __int32 aObjectID,
  const PlcObtrackObjectStatus aObjectStatus,
  const PlcObtrackObjectClass aObjectClass,
  const __int32* aConfidence,
  const TPlcRect* aPosition,
  const double* aSpeed,
  const __int32* aDuration,
  const __int32* aDirection,
  const double* aSize,
  const double* aObjectWidth,
  const double* aObjectHeight,
  const PlcObtrackProcessSize* aProcessSize,
  const char* aGscNetName);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackObjectRawData(
  ConstHGscAction anAction,
  TGLibDateTime& aTimeStamp,
  TPlcMediaChannelID& aChannel,
  __int32& aArea,
  __int32& aObjectID,
  PlcObtrackObjectStatus& aObjectStatus,
  PlcObtrackObjectClass& aObjectClass,
  const __int32*& aConfidence,
  const TPlcRect*& aPosition,
  const double*& aSpeed,
  const __int32*& aDuration,
  const __int32*& aDirection,
  const double*& aSize,
  const double*& aObjectWidth,
  const double*& aObjectHeight,
  const PlcObtrackProcessSize*& aProcessSize,
  const char*& aGscNetName);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateObtrackObjectRawData(
  var aTimeStamp: TGLibDateTime;
  var aChannel: TPlcMediaChannelID;
  aArea: Integer;
  aObjectID: Integer;
  aObjectStatus: PlcObtrackObjectStatus;
  aObjectClass: PlcObtrackObjectClass;
  aConfidence: PInteger;
  aPosition: PTPlcRect;
  aSpeed: PDouble;
  aDuration: PInteger;
  aDirection: PInteger;
  aSize: PDouble;
  aObjectWidth: PDouble;
  aObjectHeight: PDouble;
  aProcessSize: PPlcObtrackProcessSize;
  aGscNetName: PAnsiChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackObjectRawData(
  const anAction: HGscAction;
  out aTimeStamp: TGLibDateTime;
  out aChannel: TPlcMediaChannelID;
  out aArea: Integer;
  out aObjectID: Integer;
  out aObjectStatus: PlcObtrackObjectStatus;
  out aObjectClass: PlcObtrackObjectClass;
  out aConfidence: PInteger;
  out aPosition: PTPlcRect;
  out aSpeed: PDouble;
  out aDuration: PInteger;
  out aDirection: PInteger;
  out aSize: PDouble;
  out aObjectWidth: PDouble;
  out aObjectHeight: PDouble;
  out aProcessSize: PPlcObtrackProcessSize;
  out aGscNetName: PAnsiChar)
  : Boolean; stdcall; external GscActionsDll;
```

20.37 OBTRACK tunnel alarm

ObtrackTunnelAlarm (Channel, TimeStamp, AlarmReason, ObjectID, AlarmAreaID, *ObjectArea*)

Description : *OBTRACK tunnel alarm notification.*

Code : ac_ObtrackTunnelAlarm (301)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

AlarmReason (alarm reason) :
Type : PlcTunnelAlarmReason
Description : Alarm reason.

ObjectID (object ID) :
Type : __int32
Description : Object ID.

AlarmAreaID (alarm area ID) :
Type : __int32
Description : Alarm area ID.

ObjectArea (object area) [optional] :
Type : TPlcRect
Description : Object area.

Text

```
ObtrackTunnelAlarm (32, "2013/09/05 14:59:59,999 GMT+02:00",
0, 32, 32, ObjectArea:  { 0, 0, 0, 0 })
```

C++ - Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateObtrackTunnelAlarm(
  const TPlcMediaChannelID& aChannel,
  const TGLibDateTime& aTimeStamp,
  const PlcTunnelAlarmReason aAlarmReason,
  const __int32 aObjectID,
  const __int32 aAlarmAreaID,
  const TPlcRect* aObjectArea);
```

C++ - Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeObtrackTunnelAlarm(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  TGLibDateTime& aTimeStamp,
  PlcTunnelAlarmReason& aAlarmReason,
  __int32& aObjectID,
  __int32& aAlarmAreaID,
  const TPlcRect*& aObjectArea);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateObtrackTunnelAlarm(
  var aChannel: TPlcMediaChannelID;
  var aTimeStamp: TGLibDateTime;
  aAlarmReason: PlcTunnelAlarmReason;
  aObjectID: Integer;
  aAlarmAreaID: Integer;
  aObjectArea: PTPlcRect)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeObtrackTunnelAlarm(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aTimeStamp: TGLibDateTime;
  out aAlarmReason: PlcTunnelAlarmReason;
  out aObjectID: Integer;
  out aAlarmAreaID: Integer;
  out aObjectArea: PTPlcRect)
  : Boolean; stdcall; external GscActionsDll;
```

20.38 Sensor alarm finished

SensorAlarmFinished (Channel, SensorType)

Description : *This action will be fired when the alarm is finished.*

Code : ac_SensorAlarmFinished (84)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Text

```
SensorAlarmFinished (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSensorAlarmFinished(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSensorAlarmFinished(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType);
```

Delphi – Create

```
function GscAct_CreateSensorAlarmFinished(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSensorAlarmFinished(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind)
  : Boolean; stdcall; external GscActionsDll;
```

20.39 Sensor inhibit alarm finished

SensorInhibitAlarmFinished (Channel, SensorType)

Description : *This action will be fired when the inhibit alarm is finished.*

Code : ac_SensorInhibitAlarmFinished (85)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Text

```
SensorInhibitAlarmFinished (32, 0)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSensorInhibitAlarmFinished(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSensorInhibitAlarmFinished(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType);
```

Delphi – Create

```
function GscAct_CreateSensorInhibitAlarmFinished(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSensorInhibitAlarmFinished(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind)
  : Boolean; stdcall; external GscActionsDll;
```

20.40 Sensor inhibit video alarm

SensorInhibitVideoAlarm (Channel, SensorType, *ADArea*, *ADCell*, *VMDGroup*, *VMDZone*, *VMDCycle*, *AlarmArea*, *ObjectClass*)

Description : *This action will be fired when the motion in inhibit area is detected.*

Attention : *This action has extended create and decode functions!*

Code : ac_SensorInhibitVideoAlarm (81)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

ADArea (AD alarm area) [optional] :
Type : ADArea
Description : AD alarm area.

ADCell (AD cell) [optional] :
Type : __int32
Description : AD cell nr.

VMDGroup (VMD alarm group) [optional] :
Type : VMDGroup
Description : VMD alarm group.

VMDZone (VMD zone) [optional] :
Type : __int32
Description : VMD zone nr.

VMDCycle (VMD cycle) [optional] :
Type : VMDCycle
Description : VMD measure cycle.

AlarmArea (alarm area) [optional] :
Type : TPlcRect
Description : Alarm area.

ObjectClass (object class) [optional] :
Type : PlcObtrackObjectClass
Description : OBTRACK object class.

Text

```
SensorInhibitVideoAlarm (32, 0, ADArea: 0, ADCell: 32,
VMDGroup: 0, VMDZone: 32, VMDCycle: 0, AlarmArea: {
0, 0, 0, 0 }, ObjectClass: 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction __stdcall
GscAct_CreateSensorInhibitVideoAlarmEx(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const ADArea* aADArea,
  const __int32* aADCell,
  const VMDGroup* aVMDGroup,
  const __int32* aVMDZone,
  const VMDCycle* aVMDCycle,
  const TPlcRect* aAlarmArea,
  const PlcObtrackObjectClass* aObjectClass);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSensorInhibitVideoAlarmEx(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  const ADArea*& aADArea,
  const __int32*& aADCell,
  const VMDGroup*& aVMDGroup,
  const __int32*& aVMDZone,
  const VMDCycle*& aVMDCycle,
  const TPlcRect*& aAlarmArea,
  const PlcObtrackObjectClass*& aObjectClass);
```

Delphi – Create

```
function GscAct_CreateSensorInhibitVideoAlarmEx(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  aADArea: PADArea;
  aADCell: PInteger;
  aVMDGroup: PVMDGroup;
  aVMDZone: PInteger;
  aVMDCycle: PVMDCycle;
  aAlarmArea: PTPlcRect;
  aObjectClass: PPlcObtrackObjectClass)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSensorInhibitVideoAlarmEx(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aADArea: PADArea;
  out aADCell: PInteger;
  out aVMDGroup: PVMDGroup;
  out aVMDZone: PInteger;
  out aVMDCycle: PVMDCycle;
  out aAlarmArea: PTPlcRect;
  out aObjectClass: PPlcObtrackObjectClass)
  : Boolean; stdcall; external GscActionsDll;
```

20.41 Sensor video alarm

SensorVideoAlarm (Channel, SensorType, *ADArea*, *ADCell*, *VMDGroup*, *VMDZone*, *VMDCycle*, *AlarmArea*, *ObjectClass*)

Description : *This action will be fired when video alarm is detected.*

Attention : *This action has extended create and decode functions!*

Code : ac_SensorVideoAlarm (80)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

ADArea (AD alarm area) [optional] :
Type : ADArea
Description : AD alarm area.

ADCell (AD cell) [optional] :
Type : __int32
Description : AD cell nr.

VMDGroup (VMD alarm group) [optional] :
Type : VMDGroup
Description : VMD alarm group.

VMDZone (VMD zone) [optional] :
Type : __int32
Description : VMD zone nr.

VMDCycle (VMD cycle) [optional] :
Type : VMDCycle
Description : VMD measure cycle.

AlarmArea (alarm area) [optional] :
Type : TPlcRect
Description : Alarm area.

ObjectClass (object class) [optional] :
Type : PlcObtrackObjectClass
Description : OBTRACK object class.

Text

```
SensorVideoAlarm (32, 0, ADArea: 0, ADCell: 32,
VMDGroup: 0, VMDZone: 32, VMDCycle: 0, AlarmArea: {
0, 0, 0, 0 }, ObjectClass: 0)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSensorVideoAlarmEx(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const ADArea* aADArea,
  const __int32* aADCell,
  const VMDGroup* aVMDGroup,
  const __int32* aVMDZone,
  const VMDCycle* aVMDCycle,
  const TPlcRect* aAlarmArea,
  const PlcObtrackObjectClass* aObjectClass);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSensorVideoAlarmEx(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  const ADArea*& aADArea,
  const __int32*& aADCell,
  const VMDGroup*& aVMDGroup,
  const __int32*& aVMDZone,
  const VMDCycle*& aVMDCycle,
  const TPlcRect*& aAlarmArea,
  const PlcObtrackObjectClass*& aObjectClass);
```

Delphi – Create

```
function GscAct_CreateSensorVideoAlarmEx(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  aADArea: PADArea;
  aADCell: PInteger;
  aVMDGroup: PVMDGroup;
  aVMDZone: PInteger;
  aVMDCycle: PVMDCycle;
  aAlarmArea: PTPlcRect;
  aObjectClass: PPlcObtrackObjectClass)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSensorVideoAlarmEx(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aADArea: PADArea;
  out aADCell: PInteger;
  out aVMDGroup: PVMDGroup;
  out aVMDZone: PInteger;
  out aVMDCycle: PVMDCycle;
  out aAlarmArea: PTPlcRect;
  out aObjectClass: PPlcObtrackObjectClass)
  : Boolean; stdcall; external GscActionsDll;
```

20.42 Set system time

SetSystemTime (TimeStamp)

Description : *Set system time.*

Code : ac_SetSystemTime (385)

Class : ak_Video (1)

Parameters :

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

Text

```
SetSystemTime ("2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetSystemTime(
  const TGLibDateTime& aTimeStamp);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetSystemTime(
  ConstHGscAction anAction,
  TGLibDateTime& aTimeStamp);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateSetSystemTime(
  var aTimeStamp: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetSystemTime(
  const anAction: HGscAction;
  out aTimeStamp: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

20.43 Set test picture mode

SetTestPictureMode (Channel, Mode)

Description : *Enable or disable test picture generator.*

Code : ac_SetTestPictureMode (72)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Mode (enable) :
Type : bool
Description : Enable or disable test picture generator.

Text

```
SetTestPictureMode (32, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSetTestPictureMode(
  const TPlcMediaChannelID& aChannel,
  const bool aMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSetTestPictureMode(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  bool& aMode);
```

Delphi – Create

```
function GscAct_CreateSetTestPictureMode(
  var aChannel: TPlcMediaChannelID;
  aMode: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSetTestPictureMode(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aMode: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

20.44 VCA set armed

```
VCASetArmed (Channel, SensorType, Armed)
```

Description : *Send this action to notify VCA to be armed or disarmed.*

Code : ac_VCASetArmed (537)

Class : ak_Video (1)

Parameters :

Channel (Channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (Sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Armed (Armed (true) or disarmed (false)) :
Type : bool
Description : Armed (true) or disarmed (false)

Text

```
VCASetArmed (32, 0, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCASetArmed(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const bool aArmed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCASetArmed(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  bool& aArmed);
```

Delphi – Create

```
function GscAct_CreateVCASetArmed(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  aArmed: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCASetArmed(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aArmed: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

20.45 VCA status answer

VCAStatusAnswer (Channel, SensorType, Armed)

Description : *This action will be sent in response to VCA status request.*

Code : ac_VCAStatusAnswer (539)

Class : ak_Video (1)

Parameters :

Channel (Channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (Sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Armed (Armed (true) or disarmed (false)) :
Type : bool
Description : Armed (true) or disarmed (false)

Text

```
VCAStatusAnswer (32, 0, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAStatusAnswer(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const bool aArmed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAStatusAnswer(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  bool& aArmed);
```

Delphi – Create

```
function GscAct_CreateVCAStatusAnswer(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  aArmed: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAStatusAnswer(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aArmed: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

20.46 VCA status request

VCAStatusRequest (Channel, SensorType)

Description : *Send this action to request status of VCA. As response VCA will send VCA status answer.*

Code : ac_VCAStatusRequest (538)

Class : ak_Video (1)

Parameters :

Channel (Channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (Sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Text

```
VCAStatusRequest (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAStatusRequest(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAStatusRequest(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType);
```

Delphi – Create

```
function GscAct_CreateVCAStatusRequest(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAStatusRequest(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind)
  : Boolean; stdcall; external GscActionsDll;
```

20.47 Video contrast detected

VideoContrastDetected (Channel)

Description : *This action will be fired when the contrast is detected in the video signal.*

Code : ac_VideoContrastDetected (46)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
VideoContrastDetected (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoContrastDetected(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoContrastDetected(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateVideoContrastDetected(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoContrastDetected(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

20.48 Video contrast failed

VideoContrastFailed (Channel)

Description : *This action will be fired when the contrast is lost in the video signal.*

Code : ac_VideoContrastFailed (47)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
VideoContrastFailed (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoContrastFailed(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoContrastFailed(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateVideoContrastFailed(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoContrastFailed(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

20.49 Video set image brightness

`VideoSetImageBrightness (Channel, SensorType, Brightness)`

Description : *Video set image brightness.*

Code : `ac_VideoSetImageBrightness` (391)

Class : `ak_Video` (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Brightness (brightness) :
Type : __int32
Description : Brightness.

Text

```
VideoSetImageBrightness (32, 0, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoSetImageBrightness(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const __int32 aBrightness);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoSetImageBrightness(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  __int32& aBrightness);
```

Delphi – Create

```
function GscAct_CreateVideoSetImageBrightness(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  aBrightness: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoSetImageBrightness(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aBrightness: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

20.50 Video set image contrast

```
VideoSetImageContrast (Channel, SensorType, Contrast)
```

GEUTEBRÜCK

Description : *Video set image contrast.*

Code : `ac_VideoSetImageContrast` (390)

Class : `ak_Video` (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Contrast (contrast) :
Type : __int32
Description : Contrast.

Text

```
VideoSetImageContrast (32, 0, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoSetImageContrast(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const __int32 aContrast);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoSetImageContrast(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  __int32& aContrast);
```

Delphi – Create

```
function GscAct_CreateVideoSetImageContrast(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  aContrast: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoSetImageContrast(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aContrast: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

20.51 Video set image saturation

VideoSetImageSaturation (Channel, SensorType, Saturation)

Description : *Video set image saturation.*

Code : ac_VideoSetImageSaturation (392)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SensorType (sensor type) :
Type : VideoSensorKind
Description : Sensor type.

Saturation (saturation) :
Type : __int32
Description : Saturation.

Text

```
VideoSetImageSaturation (32, 0, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoSetImageSaturation(
  const TPlcMediaChannelID& aChannel,
  const VideoSensorKind aSensorType,
  const __int32 aSaturation);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoSetImageSaturation(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSensorKind& aSensorType,
  __int32& aSaturation);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateVideoSetImageSaturation(
  var aChannel: TPlcMediaChannelID;
  aSensorType: VideoSensorKind;
  aSaturation: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoSetImageSaturation(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSensorType: VideoSensorKind;
  out aSaturation: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

20.52 Video source has changed

VideoSourceChanged (Channel, SignalNorm, SignalType, InterlaceType)

Description : *This action indicates the changes on the video input source.*

Code : ac_VideoSourceChanged (45)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

SignalNorm (signal norm) :
Type : VideoSignalNorm
Description : New signal norm.

SignalType (signal type) :
Type : VideoSignalType
Description : New signal type.

InterlaceType (interlace type) :
Type : VideoInterlaceType
Description : New interlace type.

Text

```
VideoSourceChanged (32, 0, 0, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoSourceChanged(
  const TPlcMediaChannelID& aChannel,
  const VideoSignalNorm aSignalNorm,
  const VideoSignalType aSignalType,
  const VideoInterlaceType aInterlaceType);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoSourceChanged(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel,
  VideoSignalNorm& aSignalNorm,
  VideoSignalType& aSignalType,
  VideoInterlaceType& aInterlaceType);
```

Delphi – Create

```
function GscAct_CreateVideoSourceChanged(
  var aChannel: TPlcMediaChannelID;
  aSignalNorm: VideoSignalNorm;
  aSignalType: VideoSignalType;
  aInterlaceType: VideoInterlaceType)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoSourceChanged(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID;
  out aSignalNorm: VideoSignalNorm;
  out aSignalType: VideoSignalType;
  out aInterlaceType: VideoInterlaceType)
  : Boolean; stdcall; external GscActionsDll;
```

20.53 Video sync detected

VideoSyncDetected (Channel)

Description : *This action will be fired when the sync is detected in the video signal.*

Code : ac_VideoSyncDetected (40)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
VideoSyncDetected (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoSyncDetected(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoSyncDetected(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateVideoSyncDetected(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoSyncDetected(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

20.54 Video sync failed

VideoSyncFailed (Channel)

Description : *This action will be fired when the sync is lost in the video signal.*

Code : ac_VideoSyncFailed (41)

Class : ak_Video (1)

Parameters :

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
VideoSyncFailed (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVideoSyncFailed(
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVideoSyncFailed(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateVideoSyncFailed(
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVideoSyncFailed(
  const anAction: HGscAction;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

21 Viewer actions

All actions for viewer customizing.

21.1 VC alarm queue confirm

```
VCAlarmQueueConfirm (Viewer, SelectionMode)
```

Description : *Confirm alarm from the queue on the viewer client.*

Code : ac_VCAlarmQueueConfirm (252)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

SelectionMode (selection mode) :
Type : PlcViewerAlarmQueueSelection
Description : Selection mode.

Text

```
VCAlarmQueueConfirm (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueConfirm(
  const TPlcMediaChannelID& aViewer,
  const PlcViewerAlarmQueueSelection aSelectionMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueConfirm(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  PlcViewerAlarmQueueSelection& aSelectionMode);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueConfirm(
  var aViewer: TPlcMediaChannelID;
  aSelectionMode: PlcViewerAlarmQueueSelection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueConfirm(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aSelectionMode: PlcViewerAlarmQueueSelection)
  : Boolean; stdcall; external GscActionsDll;
```

21.2 VC alarm queue confirm by instance

VCAlarmQueueConfirmByInstance (Viewer, AlarmID)

Description : *Confirm alarm from the queue on the viewer client.*

Code : ac_VCAlarmQueueConfirmByInstance (257)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

AlarmID (instance ID) :
Type : __int64
Description : Instance ID of the alarm.

Text

```
VCAlarmQueueConfirmByInstance (32, 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueConfirmByInstance(
  const TPlcMediaChannelID& aViewer,
  const __int64& aAlarmID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueConfirmByInstance(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  __int64& aAlarmID);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueConfirmByInstance(
  var aViewer: TPlcMediaChannelID;
  var aAlarmID: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueConfirmByInstance(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aAlarmID: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

21.3 VC alarm queue confirm by type

VCAlarmQueueConfirmByType (Viewer, TypeID, SelectionMode)

Description : *Confirm alarm of specific type from the queue on the viewer client.*

Code : ac_VCAlarmQueueConfirmByType (253)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

SelectionMode (selection mode) :
Type : PlcViewerAlarmQueueSelection
Description : Selection mode.

Text

```
VCAlarmQueueConfirmByType (32, "event type", 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueConfirmByType(
  const TPlcMediaChannelID& aViewer,
  const TPlcEventTypeID& aTypeID,
  const PlcViewerAlarmQueueSelection aSelectionMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueConfirmByType(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcEventTypeID& aTypeID,
  PlcViewerAlarmQueueSelection& aSelectionMode);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueConfirmByType(
  var aViewer: TPlcMediaChannelID;
  var aTypeID: TPlcEventTypeID;
  aSelectionMode: PlcViewerAlarmQueueSelection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueConfirmByType(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aTypeID: TPlcEventTypeID;
  out aSelectionMode: PlcViewerAlarmQueueSelection)
  : Boolean; stdcall; external GscActionsDll;
```

21.4 VC alarm queue remove

VCAlarmQueueRemove (Viewer, SelectionMode)

Description : *Remove alarm from the queue on the viewer client.*

Code : ac_VCAlarmQueueRemove (254)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

SelectionMode (selection mode) :
Type : PlcViewerAlarmQueueSelection
Description : Selection mode.

Text

```
VCAlarmQueueRemove (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueRemove(
  const TPlcMediaChannelID& aViewer,
  const PlcViewerAlarmQueueSelection aSelectionMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueRemove(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  PlcViewerAlarmQueueSelection& aSelectionMode);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueRemove(
  var aViewer: TPlcMediaChannelID;
  aSelectionMode: PlcViewerAlarmQueueSelection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueRemove(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aSelectionMode: PlcViewerAlarmQueueSelection)
  : Boolean; stdcall; external GscActionsDll;
```

21.5 VC alarm queue remove by instance

VCAlarmQueueRemoveByInstance (Viewer, AlarmID)

Description : *Remove alarm from the queue on the viewer client.*

Code : ac_VCAlarmQueueRemoveByInstance (258)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

AlarmID (instance ID) :
Type : __int64
Description : Instance ID of the alarm.

Text

```
VCAlarmQueueRemoveByInstance (32, 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueRemoveByInstance(
  const TPlcMediaChannelID& aViewer,
  const __int64& aAlarmID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueRemoveByInstance(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  __int64& aAlarmID);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueRemoveByInstance(
  var aViewer: TPlcMediaChannelID;
  var aAlarmID: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueRemoveByInstance(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aAlarmID: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

21.6 VC alarm queue remove by type

VCAlarmQueueRemoveByType (Viewer, TypeID, SelectionMode)

Description : *Remove alarm of specific type from the queue on the viewer client.*

Code : ac_VCAlarmQueueRemoveByType (255)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

SelectionMode (selection mode) :
Type : PlcViewerAlarmQueueSelection
Description : Selection mode.

Text

```
VCAlarmQueueRemoveByType (32, "event type", 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueRemoveByType(
  const TPlcMediaChannelID& aViewer,
  const TPlcEventTypeID& aTypeID,
  const PlcViewerAlarmQueueSelection aSelectionMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueRemoveByType(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcEventTypeID& aTypeID,
  PlcViewerAlarmQueueSelection& aSelectionMode);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueRemoveByType(
  var aViewer: TPlcMediaChannelID;
  var aTypeID: TPlcEventTypeID;
  aSelectionMode: PlcViewerAlarmQueueSelection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueRemoveByType(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aTypeID: TPlcEventTypeID;
  out aSelectionMode: PlcViewerAlarmQueueSelection)
  : Boolean; stdcall; external GscActionsDll;
```

21.7 VC alarm queue select

```
VCAlarmQueueSelect (Viewer, SelectionMode)
```

Description : *Select alarm from the queue on the viewer client.*

Code : `ac_VCAlarmQueueSelect` (250)

Class : `ak_Viewer` (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

SelectionMode (selection mode) :
Type : PlcViewerAlarmQueueSelection
Description : Selection mode.

Text

```
VCAlarmQueueSelect (32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueSelect(
  const TPlcMediaChannelID& aViewer,
  const PlcViewerAlarmQueueSelection aSelectionMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueSelect(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  PlcViewerAlarmQueueSelection& aSelectionMode);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueSelect(
  var aViewer: TPlcMediaChannelID;
  aSelectionMode: PlcViewerAlarmQueueSelection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueSelect(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aSelectionMode: PlcViewerAlarmQueueSelection)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

21.8 VC alarm queue select by instance

VCAlarmQueueSelectByInstance (Viewer, AlarmID)

Description : *Select alarm from the queue on the viewer client.*

Code : ac_VCAlarmQueueSelectByInstance (256)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

AlarmID (instance ID) :
Type : __int64
Description : Instance ID of the alarm.

Text

```
VCAlarmQueueSelectByInstance (32, 64)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueSelectByInstance(
  const TPlcMediaChannelID& aViewer,
  const __int64& aAlarmID);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueSelectByInstance(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  __int64& aAlarmID);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueSelectByInstance(
  var aViewer: TPlcMediaChannelID;
  var aAlarmID: Int64)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueSelectByInstance(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aAlarmID: Int64)
  : Boolean; stdcall; external GscActionsDll;
```

21.9 VC alarm queue select by type

VCAlarmQueueSelectByType (Viewer, TypeID, SelectionMode)

Description : *Select alarm of specific type from the queue on the viewer client.*

Code : ac_VCAlarmQueueSelectByType (251)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

SelectionMode (selection mode) :
Type : PlcViewerAlarmQueueSelection
Description : Selection mode.

Text

```
VCAlarmQueueSelectByType (32, "event type", 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueSelectByType(
  const TPlcMediaChannelID& aViewer,
  const TPlcEventTypeID& aTypeID,
  const PlcViewerAlarmQueueSelection aSelectionMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueSelectByType(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcEventTypeID& aTypeID,
  PlcViewerAlarmQueueSelection& aSelectionMode);
```

Delphi – Create

```
function GscAct_CreateVCAlarmQueueSelectByType(
  var aViewer: TPlcMediaChannelID;
  var aTypeID: TPlcEventTypeID;
  aSelectionMode: PlcViewerAlarmQueueSelection)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueSelectByType(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aTypeID: TPlcEventTypeID;
  out aSelectionMode: PlcViewerAlarmQueueSelection)
  : Boolean; stdcall; external GscActionsDll;
```

21.10 VC change scene by name

VCChangeSceneByName (Viewer, Scene)

Description : *Viewer client change scene by name.*

Code : ac_VCChangeSceneByName (207)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Scene (scene) :
Type : widestring
Description : Scene name.

Text

```
VCChangeSceneByName (32, "scene")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCChangeSceneByName(
  const TPlcMediaChannelID& aViewer,
  const wchar_t* aScene);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCChangeSceneByName(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  const wchar_t*& aScene);
```

Delphi – Create

```
function GscAct_CreateVCChangeSceneByName(
  var aViewer: TPlcMediaChannelID;
  aScene: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCChangeSceneByName(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aScene: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

21.11 VC clear scene by name

VCClearSceneByName (Viewer, Scene)

Description : *Viewer client clear scene by name.*

Code : ac_VCClearSceneByName (209)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Scene (scene) :
Type : widestring
Description : Scene name.

Text

```
VCClearSceneByName (32, "scene")
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCClearSceneByName(
  const TPlcMediaChannelID& aViewer,
  const wchar_t* aScene);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCClearSceneByName(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  const wchar_t*& aScene);
```

Delphi – Create

```
function GscAct_CreateVCClearSceneByName(
  var aViewer: TPlcMediaChannelID;
  aScene: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCClearSceneByName(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aScene: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

21.12 VC full mode

VCFullMode (Viewer, FullMode, SensitiveAreaEnabled)

Description : *Viewer client full mode.*

Code : ac_VCFullMode (232)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

FullMode (full mode) :
Type : bool
Description : Switch viewer client in full mode.

SensitiveAreaEnabled (sensitive area enabled) :
Type : bool
Description : Sensitive area enabled.

Text

```
VCFullMode (32, 1, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCFullMode(
  const TPlcMediaChannelID& aViewer,
  const bool aFullMode,
  const bool aSensitiveAreaEnabled);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCFullMode(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  bool& aFullMode,
  bool& aSensitiveAreaEnabled);
```

Delphi – Create

```
function GscAct_CreateVCFullMode(
  var aViewer: TPlcMediaChannelID;
  aFullMode: Boolean;
  aSensitiveAreaEnabled: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCFullMode(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aFullMode: Boolean;
  out aSensitiveAreaEnabled: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

21.13 VC set audio level

```
VCSetAudioLevel (Viewer, AudioLevel)
```

Description : *Viewer client set audio level.*

GEUTEBRÜCK

Code : ac_VCSetAudioLevel (221)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

AudioLevel (audio level) :
Type : __int32
Description : Audio level in range 0 (audio off) till 100 (max. volume).

Text

```
VCSetAudioLevel (32, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCSetAudioLevel(
  const TPlcMediaChannelID& aViewer,
  const __int32 aAudioLevel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCSetAudioLevel(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  __int32& aAudioLevel);
```

Delphi – Create

```
function GscAct_CreateVCSetAudioLevel(
  var aViewer: TPlcMediaChannelID;
  aAudioLevel: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCSetAudioLevel(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aAudioLevel: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

21.14 VC show viewer text

VCShowViewerText (Viewer, ShowText)

Description : *Viewer client show viewer text.*

Code : ac_VCShowViewerText (231)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

ShowText (show text) :
Type : bool
Description : Show text in viewer client.

Text

```
VCShowViewerText (32, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCShowViewerText(
  const TPlcMediaChannelID& aViewer,
  const bool aShowText);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCShowViewerText(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  bool& aShowText);
```

Delphi – Create

```
function GscAct_CreateVCShowViewerText(
  var aViewer: TPlcMediaChannelID;
  aShowText: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCShowViewerText(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aShowText: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

21.15 VC stretch mode

VCStretchMode (Viewer, StretchMode)

Description : *Viewer client stretch mode.*

Code : ac_VCStretchMode (233)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

StretchMode (stretch mode) :
Type : bool
Description : Switch viewer client in stretch mode.

Text

```
VCStretchMode (32, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCStretchMode(
  const TPlcMediaChannelID& aViewer,
  const bool aStretchMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCStretchMode(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  bool& aStretchMode);
```

Delphi – Create

```
function GscAct_CreateVCStretchMode(
  var aViewer: TPlcMediaChannelID;
  aStretchMode: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCStretchMode(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aStretchMode: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

21.16 Viewer change scene

ViewerChangeScene (Viewer)

Description : *Viewer change scene.*

Code : ac_ViewerChangeScene (206)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Text

```
ViewerChangeScene (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerChangeScene(
  const TPlcMediaChannelID& aViewer);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerChangeScene(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateViewerChangeScene(
  var aViewer: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerChangeScene(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

21.17 Viewer change sync audio/video

ViewerSyncAudioAndVideo (Viewer, EnableSync)

Description : *Viewer change sync audio/video.*

Code : ac_ViewerSyncAudioAndVideo (220)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

EnableSync (enable sync) :
Type : bool
Description : Enable audio/video sync.

Text

```
ViewerSyncAudioAndVideo (32, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerSyncAudioAndVideo(
  const TPlcMediaChannelID& aViewer,
  const bool aEnableSync);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerSyncAudioAndVideo(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  bool& aEnableSync);
```

Delphi – Create

```
function GscAct_CreateViewerSyncAudioAndVideo(
  var aViewer: TPlcMediaChannelID;
  aEnableSync: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerSyncAudioAndVideo(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aEnableSync: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

21.18 Viewer clear

ViewerClear (Viewer)

Description : *Clear viewer.*

Code : ac_ViewerClear (202)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Text

```
ViewerClear (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerClear(
  const TPlcMediaChannelID& aViewer);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerClear(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateViewerClear(
  var aViewer: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerClear(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

21.19 Viewer clear scene

ViewerClearScene (Viewer)

Description : *Viewer clear scene.*

Code : ac_ViewerClearScene (208)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Text

```
ViewerClearScene (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerClearScene(
  const TPlcMediaChannelID& aViewer);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerClearScene(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer);
```

Delphi – Create

```
function GscAct_CreateViewerClearScene(
  var aViewer: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerClearScene(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

21.20 Viewer clear text output

ViewerClearTextOutput (Viewer)

Description : *Clear text output in the given viewer.*

Code : ac_ViewerClearTextOutput (241)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Text

```
ViewerClearTextOutput (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerClearTextOutput(
  const TPlcMediaChannelID& aViewer);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerClearTextOutput(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer);
```

Delphi – Create

```
function GscAct_CreateViewerClearTextOutput(
  var aViewer: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerClearTextOutput(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

21.21 Viewer connect

ViewerConnect (Viewer, Channel, PlayMode)

Description : *Connect camera to the viewer.*

Code : ac_ViewerConnect (201)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

PlayMode (play mode) :
Type : PlcViewerPlayMode
Description : Viewer play mode.

Text

```
ViewerConnect (32, 32, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerConnect(
  const TPlcMediaChannelID& aViewer,
  const TPlcMediaChannelID& aChannel,
  const PlcViewerPlayMode aPlayMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerConnect(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcMediaChannelID& aChannel,
  PlcViewerPlayMode& aPlayMode);
```

Delphi – Create

```
function GscAct_CreateViewerConnect(
  var aViewer: TPlcMediaChannelID;
  var aChannel: TPlcMediaChannelID;
  aPlayMode: PlcViewerPlayMode)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerConnect(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aChannel: TPlcMediaChannelID;
  out aPlayMode: PlcViewerPlayMode)
  : Boolean; stdcall; external GscActionsDll;
```

21.22 Viewer connect live

ViewerConnectLive (Viewer, Channel)

Description : *Connect live camera to the viewer.*

Code : ac_ViewerConnectLive (200)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

Text

```
ViewerConnectLive (32, 32)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerConnectLive(
  const TPlcMediaChannelID& aViewer,
  const TPlcMediaChannelID& aChannel);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerConnectLive(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcMediaChannelID& aChannel);
```

Delphi – Create

```
function GscAct_CreateViewerConnectLive(
  var aViewer: TPlcMediaChannelID;
  var aChannel: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerConnectLive(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aChannel: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

21.23 Viewer export picture

ViewerExportPicture (Viewer, FilePath)

Description : *Viewer export picture.*

Code : ac_ViewerExportPicture (223)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

FilePath (file path) :
Type : widestring
Description : File path (local or UNC). GSCView default path is used if left empty.

Text

```
ViewerExportPicture (32, "file path")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerExportPicture(
  const TPlcMediaChannelID& aViewer,
  const wchar_t* aFilePath);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerExportPicture(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  const wchar_t*& aFilePath);
```

Delphi – Create

```
function GscAct_CreateViewerExportPicture(
  var aViewer: TPlcMediaChannelID;
  aFilePath: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerExportPicture(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aFilePath: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

21.24 Viewer jump by time

```
ViewerJumpByTime (Viewer, Channel, PlayMode, TimeInSec)
```

Description : *Viewer jump by time.*

Code : ac_ViewerJumpByTime (215)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

PlayMode (play mode) :
Type : PlcViewerPlayMode
Description : Viewer play mode.

TimeInSec (time in sec) :
Type : __int32
Description : Time to jump in seconds. Use negative values to jump in past.

Text

```
ViewerJumpByTime (32, 32, 0, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerJumpByTime(
  const TPlcMediaChannelID& aViewer,
  const TPlcMediaChannelID& aChannel,
  const PlcViewerPlayMode aPlayMode,
  const __int32 aTimeInSec);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerJumpByTime(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcMediaChannelID& aChannel,
  PlcViewerPlayMode& aPlayMode,
  __int32& aTimeInSec);
```

Delphi – Create

```
function GscAct_CreateViewerJumpByTime(
  var aViewer: TPlcMediaChannelID;
  var aChannel: TPlcMediaChannelID;
  aPlayMode: PlcViewerPlayMode;
  aTimeInSec: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerJumpByTime(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aChannel: TPlcMediaChannelID;
  out aPlayMode: PlcViewerPlayMode;
  out aTimeInSec: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

21.25 Viewer maximize

ViewerMaximize (Viewer, Maximize)

Description : *Viewer maximize.*

Code : ac_ViewerMaximize (222)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Maximize (maximize) :
Type : bool
Description : Maximize.

Text

```
ViewerMaximize (32, 1)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerMaximize(
  const TPlcMediaChannelID& aViewer,
  const bool aMaximize);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerMaximize(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  bool& aMaximize);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateViewerMaximize(
  var aViewer: TPlcMediaChannelID;
  aMaximize: Boolean)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerMaximize(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aMaximize: Boolean)
  : Boolean; stdcall; external GscActionsDll;
```

21.26 Viewer play from time

ViewerPlayFromTime (Viewer, Channel, PlayMode, Time)

Description : *Viewer play from time.*

Code : ac_ViewerPlayFromTime (216)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

PlayMode (play mode) :
Type : PlcViewerPlayMode
Description : Viewer play mode.

Time (time) :
Type : TGLibDateTime
Description : Time to play from.

Text

```
ViewerPlayFromTime (32, 32, 0, "2013/09/05 14:59:59,999 GMT+02:00")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerPlayFromTime(
  const TPlcMediaChannelID& aViewer,
  const TPlcMediaChannelID& aChannel,
  const PlcViewerPlayMode aPlayMode,
  const TGLibDateTime& aTime);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerPlayFromTime(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcMediaChannelID& aChannel,
  PlcViewerPlayMode& aPlayMode,
  TGLibDateTime& aTime);
```

Delphi – Create

```
function GscAct_CreateViewerPlayFromTime(
  var aViewer: TPlcMediaChannelID;
  var aChannel: TPlcMediaChannelID;
  aPlayMode: PlcViewerPlayMode;
  var aTime: TGLibDateTime)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerPlayFromTime(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aChannel: TPlcMediaChannelID;
  out aPlayMode: PlcViewerPlayMode;
  out aTime: TGLibDateTime)
  : Boolean; stdcall; external GscActionsDll;
```

21.27 Viewer print picture

ViewerPrintPicture (Viewer)

Description : *Viewer print picture.*

Code : ac_ViewerPrintPicture (224)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Text

```
ViewerPrintPicture (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerPrintPicture(
  const TPlcMediaChannelID& aViewer);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerPrintPicture(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer);
```

Delphi – Create

```
function GscAct_CreateViewerPrintPicture(
  var aViewer: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerPrintPicture(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

21.28 Viewer select

```
ViewerSelect (Viewer)
```

Description : *Select viewer.*

Code : ac_ViewerSelect (230)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Text

```
ViewerSelect (32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerSelect(
  const TPlcMediaChannelID& aViewer);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerSelect(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer);
```

Delphi – Create

```
function GscAct_CreateViewerSelect(
  var aViewer: TPlcMediaChannelID)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerSelect(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID)
  : Boolean; stdcall; external GscActionsDll;
```

21.29 Viewer set play mode

ViewerSetPlayMode (Viewer, PlayMode, PlaySpeed)

Description : *Set play mode of the viewer.*

Code : ac_ViewerSetPlayMode (217)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

PlayMode (play mode) :
Type : PlcViewerPlayMode
Description : Viewer play mode.

PlaySpeed (play speed) :
Type : __int32
Description : Optional play speed parameter.

Text

```
ViewerSetPlayMode (32, 0, 32)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerSetPlayMode(
  const TPlcMediaChannelID& aViewer,
  const PlcViewerPlayMode aPlayMode,
  const __int32 aPlaySpeed);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerSetPlayMode(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  PlcViewerPlayMode& aPlayMode,
  __int32& aPlaySpeed);
```

Delphi – Create

```
function GscAct_CreateViewerSetPlayMode(
  var aViewer: TPlcMediaChannelID;
  aPlayMode: PlcViewerPlayMode;
  aPlaySpeed: Integer)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerSetPlayMode(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aPlayMode: PlcViewerPlayMode;
  out aPlaySpeed: Integer)
  : Boolean; stdcall; external GscActionsDll;
```

21.30 Viewer show alarm by instance

```
ViewerShowAlarmByInstance (Viewer, AlarmID, PlayMode)
```

Description : *Show alarm in viewer using alarm instance ID.*

Code : ac_ViewerShowAlarmByInstance (210)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

AlarmID (instance ID) :
Type : __int64
Description : Instance ID of the alarm.

PlayMode (play mode) :
Type : PlcViewerAlarmPlayMode
Description : Viewer alarm's play mode.

Text

```
ViewerShowAlarmByInstance (32, 64, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerShowAlarmByInstance(
  const TPlcMediaChannelID& aViewer,
  const __int64& aAlarmID,
  const PlcViewerAlarmPlayMode aPlayMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerShowAlarmByInstance(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  __int64& aAlarmID,
  PlcViewerAlarmPlayMode& aPlayMode);
```

Delphi – Create

```
function GscAct_CreateViewerShowAlarmByInstance(
  var aViewer: TPlcMediaChannelID;
  var aAlarmID: Int64;
  aPlayMode: PlcViewerAlarmPlayMode)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeViewerShowAlarmByInstance(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aAlarmID: Int64;
  out aPlayMode: PlcViewerAlarmPlayMode)
  : Boolean; stdcall; external GscActionsDll;
```

21.31 Viewer show alarm by key

ViewerShowAlarmByKey (Viewer, ForeignKey, PlayMode)

Description : *Show alarm in viewer using alarm foreign key.*

Code : ac_ViewerShowAlarmByKey (211)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

ForeignKey (foreign key) [ForeignKey] :
Type : __int64
Description : Foreign key used to start the alarm.

PlayMode (play mode) :
Type : PlcViewerAlarmPlayMode
Description : Viewer alarm's play mode.

Text

```
ViewerShowAlarmByKey (32, 64, 0)
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerShowAlarmByKey(
  const TPlcMediaChannelID& aViewer,
  const __int64& aForeignKey,
  const PlcViewerAlarmPlayMode aPlayMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerShowAlarmByKey(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  __int64& aForeignKey,
  PlcViewerAlarmPlayMode& aPlayMode);
```

Delphi – Create

```
function GscAct_CreateViewerShowAlarmByKey(
  var aViewer: TPlcMediaChannelID;
  var aForeignKey: Int64;
  aPlayMode: PlcViewerAlarmPlayMode)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerShowAlarmByKey(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aForeignKey: Int64;
  out aPlayMode: PlcViewerAlarmPlayMode)
  : Boolean; stdcall; external GscActionsDll;
```

21.32 Viewer show alarm by type

ViewerShowAlarmByType (Viewer, TypeID, *ForeignKey*, PlayMode)

Description : *Show alarm in viewer using alarm type and optional foreign key.*

Code : ac_ViewerShowAlarmByType (212)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

TypeID (alarm type) [Event] :
Type : TEventTypeID
Description : Type of the alarm.

ForeignKey (foreign key) [optional] [ForeignKey] :
Type : __int64
Description : Foreign key used to start the alarm.

PlayMode (play mode) :
Type : PlcViewerAlarmPlayMode
Description : Viewer alarm's play mode.

Text

```
ViewerShowAlarmByType (32, "alarm type", ForeignKey:  64,
0)
```

GEUTEBRÜCK

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerShowAlarmByType(
  const TPlcMediaChannelID& aViewer,
  const TPlcEventTypeID& aTypeID,
  const __int64* aForeignKey,
  const PlcViewerAlarmPlayMode aPlayMode);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerShowAlarmByType(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcEventTypeID& aTypeID,
  const __int64*& aForeignKey,
  PlcViewerAlarmPlayMode& aPlayMode);
```

Delphi – Create

```
function GscAct_CreateViewerShowAlarmByType(
  var aViewer: TPlcMediaChannelID;
  var aTypeID: TPlcEventTypeID;
  aForeignKey: PInt64;
  aPlayMode: PlcViewerAlarmPlayMode)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerShowAlarmByType(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aTypeID: TPlcEventTypeID;
  out aForeignKey: PInt64;
  out aPlayMode: PlcViewerAlarmPlayMode)
  : Boolean; stdcall; external GscActionsDll;
```

21.33 Viewer text output

ViewerTextOutput (Viewer, Text)

Description : *Display free string in the given viewer.*

Code : ac_ViewerTextOutput (240)

Class : ak_Viewer (9)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Text (text string) :
Type : widestring
Description : Text string to display.

Text

```
ViewerTextOutput (32, "text string")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerTextOutput(
  const TPlcMediaChannelID& aViewer,
  const wchar_t* aText);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerTextOutput(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  const wchar_t*& aText);
```

Delphi – Create

```
function GscAct_CreateViewerTextOutput(
  var aViewer: TPlcMediaChannelID;
  aText: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerTextOutput(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aText: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

22 Viewer notifications

All viewer notifications.

22.1 Image export notification

ImageExportNotification (User, Destination, DestinationType, TimeStamp, *TimeEnd*, Channels, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Notification on export or print of channel images.*

Code : ac_ImageExportNotification (225)

Class : ak_ViewerNotification (10)

Parameters :

User (user) [UserName] :
Type : widestring
Description : Name of the user connected to the system.

Destination (destination) :
Type : widestring
Description : Destination.

DestinationType (destination type) :
Type : PlcImageExportType
Description : Destination type.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

TimeEnd (end time) [optional] :
Type : TGLibDateTime
Description : End time.

Channels (channels) :
Type : widestring
Description : Channels.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
ImageExportNotification ("user", "destination",
0, "2013/09/05 14:59:59,999 GMT+02:00", TimeEnd:
"2013/09/05 14:59:59,999 GMT+02:00", "channels",
ClientHost:  "client host", ClientType:  0,
ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateImageExportNotification(
  const wchar_t* aUser,
  const wchar_t* aDestination,
  const PlcImageExportType aDestinationType,
  const TGLibDateTime& aTimeStamp,
  const TGLibDateTime* aTimeEnd,
  const wchar_t* aChannels,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeImageExportNotification(
  ConstHGscAction anAction,
  const wchar_t*& aUser,
  const wchar_t*& aDestination,
  PlcImageExportType& aDestinationType,
  TGLibDateTime& aTimeStamp,
  const TGLibDateTime*& aTimeEnd,
  const wchar_t*& aChannels,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateImageExportNotification(
  aUser: PWideChar;
  aDestination: PWideChar;
  aDestinationType: PlcImageExportType;
  var aTimeStamp: TGLibDateTime;
  aTimeEnd: PTGLibDateTime;
  aChannels: PWideChar;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeImageExportNotification(
  const anAction: HGscAction;
  out aUser: PWideChar;
  out aDestination: PWideChar;
  out aDestinationType: PlcImageExportType;
  out aTimeStamp: TGLibDateTime;
  out aTimeEnd: PTGLibDateTime;
  out aChannels: PWideChar;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

22.2 Scene store modification

SceneStoreModification (Viewer, SceneStoreID, SceneStoreName, TimeStamp, ModificationType, User, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Scene store modification.*

Code : ac_SceneStoreModification (243)

Class : ak_ViewerNotification (10)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

SceneStoreID (scene store GUID) :
Type : GUID
Description : Scene store GUID.

SceneStoreName (scene store name) :
Type : widestring
Description : Scene store name.

TimeStamp (time stamp) :
Type : TGLibDateTime
Description : Time stamp.

ModificationType (modification type) :
Type : PlcSceneStoreModificationType
Description : Modification type.

User (user) [UserName] :
Type : widestring
Description : Name of the user.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
SceneStoreModification (32, "00000000-0000-0000-0000-000000000000",
"scene store name", "2013/09/05 14:59:59,999 GMT+02:00",
0, "user", ClientHost:  "client host", ClientType:  0,
ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateSceneStoreModification(
  const TPlcMediaChannelID& aViewer,
  const GUID& aSceneStoreID,
  const wchar_t* aSceneStoreName,
  const TGLibDateTime& aTimeStamp,
  const PlcSceneStoreModificationType aModificationType,
  const wchar_t* aUser,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeSceneStoreModification(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  GUID& aSceneStoreID,
  const wchar_t*& aSceneStoreName,
  TGLibDateTime& aTimeStamp,
  PlcSceneStoreModificationType& aModificationType,
  const wchar_t*& aUser,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateSceneStoreModification(
  var aViewer: TPlcMediaChannelID;
  var aSceneStoreID: TGuid;
  aSceneStoreName: PWideChar;
  var aTimeStamp: TGLibDateTime;
  aModificationType: PlcSceneStoreModificationType;
  aUser: PWideChar;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeSceneStoreModification(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aSceneStoreID: TGuid;
  out aSceneStoreName: PWideChar;
  out aTimeStamp: TGLibDateTime;
  out aModificationType: PlcSceneStoreModificationType;
  out aUser: PWideChar;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

22.3 VC alarm queue notification

VCAlarmQueueNotification (Viewer, Notification, AlarmID, TypeID, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Alarm queue notification on the viewer client.*

Code : ac_VCAlarmQueueNotification (259)

Class : ak_ViewerNotification (10)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Notification (notification) :
Type : PlcViewerAlarmState
Description : Alarm state notification.

AlarmID (instance ID) :
Type : __int64
Description : Instance ID of the alarm.

TypeID (event type) [Event] :
Type : TEventTypeID
Description : Type of the event.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
VCAlarmQueueNotification (32, 0, 64, "event type",
ClientHost:  "client host", ClientType:  0,
ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCAlarmQueueNotification(
  const TPlcMediaChannelID& aViewer,
  const PlcViewerAlarmState aNotification,
  const __int64& aAlarmID,
  const TPlcEventTypeID& aTypeID,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCAlarmQueueNotification(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  PlcViewerAlarmState& aNotification,
  __int64& aAlarmID,
  TPlcEventTypeID& aTypeID,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateVCAlarmQueueNotification(
  var aViewer: TPlcMediaChannelID;
  aNotification: PlcViewerAlarmState;
  var aAlarmID: Int64;
  var aTypeID: TPlcEventTypeID;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCAlarmQueueNotification(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aNotification: PlcViewerAlarmState;
  out aAlarmID: Int64;
  out aTypeID: TPlcEventTypeID;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

22.4 VC scene changed

VCSceneChanged (Viewer, Scene)

Description : *Viewer client scene changed.*

Code : ac_VCSceneChanged (213)

Class : ak_ViewerNotification (10)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Scene (scene) :
Type : widestring
Description : Scene name.

Text

```
VCSceneChanged (32, "scene")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateVCSceneChanged(
  const TPlcMediaChannelID& aViewer,
  const wchar_t* aScene);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeVCSceneChanged(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  const wchar_t*& aScene);
```

Delphi – Create

```
function GscAct_CreateVCSceneChanged(
  var aViewer: TPlcMediaChannelID;
  aScene: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeVCSceneChanged(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aScene: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

22.5 Viewer cleared

ViewerCleared (Viewer, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Viewer cleared.*

Code : ac_ViewerCleared (204)

Class : ak_ViewerNotification (10)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
ViewerCleared (32, ClientHost:  "client host",
ClientType:  0, ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerCleared(
  const TPlcMediaChannelID& aViewer,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerCleared(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateViewerCleared(
  var aViewer: TPlcMediaChannelID;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerCleared(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

22.6 Viewer connected

ViewerConnected (Viewer, Channel, PlayMode, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Viewer connected to the camera.*

Code : ac_ViewerConnected (203)

Class : ak_ViewerNotification (10)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

PlayMode (play mode) :
Type : PlcViewerPlayMode
Description : Viewer play mode.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
ViewerConnected (32, 32, 0, ClientHost:  "client host",
ClientType:  0, ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerConnected(
  const TPlcMediaChannelID& aViewer,
  const TPlcMediaChannelID& aChannel,
  const PlcViewerPlayMode aPlayMode,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

GEUTEBRÜCK

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerConnected(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcMediaChannelID& aChannel,
  PlcViewerPlayMode& aPlayMode,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateViewerConnected(
  var aViewer: TPlcMediaChannelID;
  var aChannel: TPlcMediaChannelID;
  aPlayMode: PlcViewerPlayMode;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerConnected(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aChannel: TPlcMediaChannelID;
  out aPlayMode: PlcViewerPlayMode;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

22.7 Viewer play mode changed

ViewerPlayModeChanged (Viewer, Channel, PlayMode, *ChannelTime*, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Viewer play mode changed.*

Code : ac_ViewerPlayModeChanged (205)

Class : ak_ViewerNotification (10)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

PlayMode (play mode) :
Type : PlcViewerPlayMode
Description : Viewer play mode.

ChannelTime (channel time) [optional] :
Type : TGLibDateTime
Description : Channel time.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
ViewerPlayModeChanged (32, 32, 0, ChannelTime:
"2013/09/05 14:59:59,999 GMT+02:00", ClientHost:  "client
host", ClientType:  0, ClientAccount:  "client account")
```

C++ - Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerPlayModeChanged(
  const TPlcMediaChannelID& aViewer,
  const TPlcMediaChannelID& aChannel,
  const PlcViewerPlayMode aPlayMode,
  const TGLibDateTime* aChannelTime,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ - Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerPlayModeChanged(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcMediaChannelID& aChannel,
  PlcViewerPlayMode& aPlayMode,
  const TGLibDateTime*& aChannelTime,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

GEUTEBRÜCK

Delphi – Create

```
function GscAct_CreateViewerPlayModeChanged(
  var aViewer: TPlcMediaChannelID;
  var aChannel: TPlcMediaChannelID;
  aPlayMode: PlcViewerPlayMode;
  aChannelTime: PTGLibDateTime;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

Delphi – Decode

```
function GscAct_DecodeViewerPlayModeChanged(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aChannel: TPlcMediaChannelID;
  out aPlayMode: PlcViewerPlayMode;
  out aChannelTime: PTGLibDateTime;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

22.8 Viewer selection changed

ViewerSelectionChanged (Viewer, Channel, PlayMode, *ClientHost*, *ClientType*, *ClientAccount*)

Description : *Viewer selection was changed.*

Code : ac_ViewerSelectionChanged (242)

Class : ak_ViewerNotification (10)

Parameters :

Viewer (viewer) [VideoOutput] :
Type : TMediaChannelID
Description : Global viewer or viewer client no.

Channel (channel) [VideoInput] :
Type : TMediaChannelID
Description : Channel.

PlayMode (play mode) :
Type : PlcViewerPlayMode
Description : Viewer play mode.

ClientHost (client host) [optional] :
Type : widestring
Description : Host from where the connection is done.

ClientType (client type) [optional] :
Type : PlcClientType
Description : Client type.

ClientAccount (client account) [optional] :
Type : widestring
Description : User account from where the connection is done.

Text

```
ViewerSelectionChanged (32, 32, 0, ClientHost:  "client
host", ClientType:  0, ClientAccount:  "client account")
```

C++ – Create

```
GSCACTDLLCALL HGscAction  __stdcall
GscAct_CreateViewerSelectionChanged(
  const TPlcMediaChannelID& aViewer,
  const TPlcMediaChannelID& aChannel,
  const PlcViewerPlayMode aPlayMode,
  const wchar_t* aClientHost,
  const PlcClientType* aClientType,
  const wchar_t* aClientAccount);
```

C++ – Decode

```
GSCACTDLLCALL bool __stdcall
GscAct_DecodeViewerSelectionChanged(
  ConstHGscAction anAction,
  TPlcMediaChannelID& aViewer,
  TPlcMediaChannelID& aChannel,
  PlcViewerPlayMode& aPlayMode,
  const wchar_t*& aClientHost,
  const PlcClientType*& aClientType,
  const wchar_t*& aClientAccount);
```

Delphi – Create

```
function GscAct_CreateViewerSelectionChanged(
  var aViewer: TPlcMediaChannelID;
  var aChannel: TPlcMediaChannelID;
  aPlayMode: PlcViewerPlayMode;
  aClientHost: PWideChar;
  aClientType: PPlcClientType;
  aClientAccount: PWideChar)
  : HGscAction; stdcall; external GscActionsDll;
```

GEUTEBRÜCK

Delphi – Decode

```
function GscAct_DecodeViewerSelectionChanged(
  const anAction: HGscAction;
  out aViewer: TPlcMediaChannelID;
  out aChannel: TPlcMediaChannelID;
  out aPlayMode: PlcViewerPlayMode;
  out aClientHost: PWideChar;
  out aClientType: PPlcClientType;
  out aClientAccount: PWideChar)
  : Boolean; stdcall; external GscActionsDll;
```

A List of action classes

Code	Action class
0	System (p. 306)
1	Video (p. 359)
2	Audio (p. 58)
3	CameraControl (p. 103)
4	DigitalContacts (p. 193)
5	Device (p. 183)
6	ATM (p. 46)
7	Backup (p. 62)
8	CashManagement (p. 168)
9	Viewer (p. 436)
10	ViewerNotification (p. 474)
11	POS (p. 245)
12	RemoteExport (p. 272)
13	SkiData (p. 281)
14	LPS (p. 221)
15	Logistic (p. 288)
16	Lenel (p. 226)
17	Imex (p. 210)
18	PP (p. 259)

GEUTEBRÜCK

B List of action codes

Code	Action
1	SystemStarted (p. 347)
2	SystemTerminating (p. 348)
3	UserLogin (p. 353)
4	UserLoginFailed (p. 355)
5	UserLogout (p. 357)
6	SystemSettingsChanged (p. 345)
7	LiveCheck (p. 326)
8	CustomAction (p. 307)
9	DatabaseStarted (p. 315)
10	DeviceFound (p. 183)
11	DeviceRemoved (p. 189)
12	DeviceNewFirmware (p. 191)
13	DeviceReattached (p. 188)
14	DevicePluginError (p. 184)
15	DevicePluginState (p. 186)
16	SetupUploadProgress (p. 336)
17	SetupChanged (p. 334)
19	FRCNotification (p. 320)
20	SensorAudioAlarm (p. 61)
21	ABCConnect (p. 58)
22	ABCDisconnect (p. 59)
23	ABCPlayFile (p. 59)
25	SetClock (p. 332)
30	DigitalInput (p. 194)
31	SetDigitalOutput (p. 202)
32	SetLED (p. 203)
33	SetLEDBlink (p. 205)
35	KeyPressed (p. 200)
36	KeyReleased (p. 201)
37	RedundantPowerFailure (p. 327)
38	RedundantPowerOk (p. 328)
40	VideoSyncDetected (p. 433)
41	VideoSyncFailed (p. 434)
42	ChangeADParameterSet (p. 361)
43	ChangeVMDParameterSet (p. 365)
44	ChangeCPAParameterSet (p. 362)
45	VideoSourceChanged (p. 432)
46	VideoContrastDetected (p. 426)
47	VideoContrastFailed (p. 427)
48	MakeCPAReferenceImage (p. 390)
49	IASSettingsChanged (p. 384)
50	SystemInfo (p. 343)
51	SystemWarning (p. 349)
52	SystemError (p. 342)
53	SetWatchdog (p. 333)
54	StartEvent (p. 338)
55	StopEvent (p. 340)
56	StopEventByID (p. 341)
59	EventStarted (p. 317)
60	EventStopped (p. 318)

Code	Action
61	EventRecordingChanged (p. 316)
62	StopAllEvents (p. 339)
63	KillEvent (p. 324)
64	KillEventByID (p. 325)
65	KillAllEvents (p. 323)
68	BlockingFilterActivate (p. 306)
69	BlockingFilterDeactivate (p. 307)
70	CPAMeasurement (p. 360)
72	SetTestPictureMode (p. 422)
74	ChannelLiveCheck (p. 371)
75	IPCameraRawCommand (p. 388)
80	SensorVideoAlarm (p. 418)
81	SensorInhibitVideoAlarm (p. 416)
84	SensorAlarmFinished (p. 414)
85	SensorInhibitAlarmFinished (p. 415)
90	ChangeObtrackParameterSet (p. 364)
91	ObtrackFrameRawData (p. 403)
92	ObtrackObjectRawData (p. 409)
94	ObtrackChannelCounter (p. 398)
95	ObtrackGroupCounter (p. 404)
96	ObtrackChannelCounterThreshold (p. 400)
97	ObtrackGroupCounterThreshold (p. 406)
98	ObtrackChannelSetCounter (p. 401)
99	ObtrackGroupSetCounter (p. 407)
101	PanRight (p. 156)
102	PanLeft (p. 155)
103	PanStop (p. 157)
104	TiltUp (p. 163)
105	TiltDown (p. 161)
106	TiltStop (p. 162)
107	ZoomIn (p. 164)
108	ZoomOut (p. 165)
109	ZoomStop (p. 166)
110	FocusNear (p. 142)
111	FocusFar (p. 141)
112	FocusStop (p. 143)
113	IrisOpen (p. 145)
114	IrisClose (p. 144)
115	IrisStop (p. 146)
116	PrePosCallUp (p. 151)
117	DefaultPosCallUp (p. 150)
118	CameraTourStart (p. 131)
119	CameraTourStop (p. 132)
120	CameraLightOn (p. 111)
121	CameraLightOff (p. 110)
122	CameraOn (p. 115)
123	CameraOff (p. 114)
124	CameraPumpOn (p. 117)
125	CameraPumpOff (p. 116)
126	FastSpeedOn (p. 140)
127	FastSpeedOff (p. 139)
128	CameraWashOn (p. 136)

GEUTEBRÜCK

Code	Action
129	CameraWashOff (p. 135)
130	CameraSpecFuncUOn (p. 121)
131	CameraSpecFuncUOff (p. 120)
132	CameraSpecFuncVOn (p. 123)
133	CameraSpecFuncVOff (p. 122)
134	CameraSpecFuncXOn (p. 125)
135	CameraSpecFuncXOff (p. 124)
136	CameraSpecFuncYOn (p. 127)
137	CameraSpecFuncYOff (p. 126)
138	CameraStopAll (p. 128)
139	PrePosSave (p. 159)
140	DefaultPosSave (p. 158)
141	PrePosClear (p. 138)
142	DefaultPosClear (p. 137)
143	SetCameraText (p. 160)
144	CameraSelectCharMode (p. 118)
145	CameraClearPrePosText (p. 108)
146	CameraSetPrePosText (p. 119)
147	CameraRAWOutput (p. 104)
148	CameraVersionOn (p. 134)
149	CameraVersionOff (p. 133)
150	PanAuto (p. 154)
151	AutoFocusOn (p. 104)
152	AutoFocusOff (p. 103)
153	CameraManualIrisOn (p. 113)
154	CameraManualIrisOff (p. 112)
155	CameraTextOn (p. 130)
156	CameraTextOff (p. 129)
157	CameraDayNightMode (p. 109)
158	CameraBacklightCompensationMode (p. 107)
159	CameraApplyProfile (p. 106)
160	MoveToAbsolutePosition (p. 148)
161	MoveToRelativePosition (p. 152)
162	MoveToBySpeed (p. 147)
180	ChannelInfo (p. 370)
181	ChannelWarning (p. 372)
182	ChannelError (p. 368)
190	IPSwitchOperation (p. 322)
200	ViewerConnectLive (p. 459)
201	ViewerConnect (p. 458)
202	ViewerClear (p. 455)
203	ViewerConnected (p. 483)
204	ViewerCleared (p. 481)
205	ViewerPlayModeChanged (p. 484)
206	ViewerChangeScene (p. 453)
207	VCChangeSceneByName (p. 446)
208	ViewerClearScene (p. 456)
209	VCClearSceneByName (p. 447)
210	ViewerShowAlarmByInstance (p. 468)
211	ViewerShowAlarmByKey (p. 470)
212	ViewerShowAlarmByType (p. 471)
213	VCSceneChanged (p. 480)

Code	Action
215	ViewerJumpByTime (p. 461)
216	ViewerPlayFromTime (p. 464)
217	ViewerSetPlayMode (p. 467)
220	ViewerSyncAudioAndVideo (p. 454)
221	VCSetAudioLevel (p. 449)
222	ViewerMaximize (p. 463)
223	ViewerExportPicture (p. 460)
224	ViewerPrintPicture (p. 465)
225	ImageExportNotification (p. 474)
230	ViewerSelect (p. 466)
231	VCShowViewerText (p. 451)
232	VCFullMode (p. 448)
233	VCStretchMode (p. 452)
240	ViewerTextOutput (p. 472)
241	ViewerClearTextOutput (p. 457)
242	ViewerSelectionChanged (p. 486)
243	SceneStoreModification (p. 476)
250	VCAlarmQueueSelect (p. 442)
251	VCAlarmQueueSelectByType (p. 445)
252	VCAlarmQueueConfirm (p. 436)
253	VCAlarmQueueConfirmByType (p. 438)
254	VCAlarmQueueRemove (p. 439)
255	VCAlarmQueueRemoveByType (p. 441)
256	VCAlarmQueueSelectByInstance (p. 444)
257	VCAlarmQueueConfirmByInstance (p. 437)
258	VCAlarmQueueRemoveByInstance (p. 440)
259	VCAlarmQueueNotification (p. 478)
260	DatabaseRecordingInfoTotal (p. 311)
263	DatabaseRecordingInfoRing (p. 309)
270	AutoBackupCapacityMonitoringOutOfDiskSpace (p. 66)
272	AutoBackupCapacityMonitoringCapacityWarning (p. 68)
274	AutoBackupCapacityMonitoringFileAutoDeleted (p. 63)
280	StartAutoBackup (p. 101)
281	BackupEvent (p. 88)
282	AutoBackupScheduleStarted (p. 86)
284	AutoBackupScheduleDone (p. 85)
286	AutoBackupOperationStarted (p. 82)
288	AutoBackupOperationDone (p. 79)
290	AutoBackupFileStarted (p. 76)
292	AutoBackupFileProgress (p. 73)
294	AutoBackupFileDone (p. 70)
298	AbortAutoBackup (p. 62)
299	AbortAllAutoBackups (p. 62)
300	GEMOSalarm (p. 321)
301	ObtrackTunnelAlarm (p. 412)
303	SMTPMailSend (p. 331)
310	EventBackupStarted (p. 99)
312	EventBackupDone (p. 89)
314	EventBackupFileStarted (p. 96)
316	EventBackupFileProgress (p. 94)
318	EventBackupFileDone (p. 92)
330	ChangeCameraProfile (p. 367)

GEUTEBRÜCK

Code	Action
340	SMRPViewerConnected (p. 330)
341	SMRPViewerCleared (p. 329)
350	ATMRawData (p. 53)
351	ATMRawAnswer (p. 52)
352	ATMTransaction (p. 54)
354	ACSRawData (p. 51)
355	ACSRawAnswer (p. 49)
356	ACSAccessGranted (p. 48)
357	ACSAccessDenied (p. 46)
358	InterfaceRawData (p. 249)
359	InterfaceRawAnswer (p. 248)
360	POSStatus (p. 253)
361	POSData (p. 250)
363	TerminalArticleData (p. 254)
364	TerminalPaymentData (p. 256)
365	BarcodeData (p. 245)
366	FillingPumpStatus (p. 246)
370	ActivateExternalProcess (p. 359)
371	NPRRawData (p. 392)
375	NPRRecognition (p. 395)
380	TransferBinaryBuffer (p. 351)
381	TransferBinaryChannelBuffer (p. 352)
385	SetSystemTime (p. 421)
390	VideoSetImageContrast (p. 429)
391	VideoSetImageBrightness (p. 428)
392	VideoSetImageSaturation (p. 431)
395	MediaChannelSetupInfo (p. 391)
400	SafebagOpen (p. 174)
402	SafebagData (p. 171)
405	SafebagClose (p. 168)
407	SafebagPassingOfRiskStart (p. 177)
410	SafebagPassingOfRiskData (p. 175)
413	SafebagPassingOfRiskStop (p. 180)
420	IOI43WDActivate (p. 197)
421	IOI43WDDeactivate (p. 198)
422	IOI43WDTrigger (p. 199)
423	IOI43ResetMainboard (p. 195)
424	IOI43Temperature (p. 196)
430	SkidataControl (p. 281)
431	SkidataEntry (p. 283)
432	SkidataExit (p. 284)
433	SkidataTransaction (p. 286)
434	SkidataDeviceEvent (p. 282)
440	WatchdogActivate (p. 207)
441	WatchdogDeactivate (p. 208)
442	WatchdogTrigger (p. 208)
443	Temperature (p. 206)
444	ResetMainboard (p. 202)
445	CaseOpened (p. 193)
446	CaseClosed (p. 193)
450	SetExportMarker (p. 276)
451	InitializeRemoteExport (p. 275)

Code	Action
452	StartRemoteExport (p. 277)
454	ExportProgress (p. 274)
455	ExportFinished (p. 273)
456	CancelExport (p. 272)
457	StartSceneStore (p. 279)
460	LPSPositionData (p. 221)
463	LPSQueryPosition (p. 224)
465	LogBarcodeData (p. 299)
467	LogBarcodeDataLPS (p. 301)
469	LogNPRRecognition (p. 297)
480	GSCSVehicleAccessDenied (p. 288)
482	GSCSVehicleAccessGranted (p. 292)
484	GSCSVehicleAccessExpired (p. 290)
486	GSCSVehicleAccessPending (p. 294)
500	LenelSecurityEvent (p. 238)
502	LenelAccessEvent (p. 226)
506	LenelIntercomEvent (p. 233)
508	LenelVideoEvent (p. 241)
510	LenelFireEvent (p. 230)
512	LenelRawData (p. 236)
514	LenelRefreshNames (p. 237)
520	GTectVMXAlarm (p. 381)
522	GTectVMXAlarmFinished (p. 383)
524	GTectDualSensorAlarm (p. 379)
526	GTectDualSensorAlarmFinished (p. 381)
528	GTectSceneAlarm (p. 376)
530	GTectSceneAlarmFinished (p. 377)
532	GTectAnalyticsLiveCheck (p. 375)
534	ChangeGTectParameterSet (p. 363)
536	SetClientVCA (p. 374)
537	VCASetArmed (p. 423)
538	VCAStatusRequest (p. 425)
539	VCAStatusAnswer (p. 424)
540	IPCamFailoverNotification (p. 385)
541	IPCamFailoverRestore (p. 387)
550	ImexCapacityWarning (p. 214)
552	ImexCapacityOutOfDiskSpace (p. 212)
554	ImexCapacityFileAutoDeleted (p. 210)
560	ImexExportImageFromDB (p. 217)
562	ImexExportEventImage (p. 216)
564	ImexExportImageFromLiveStream (p. 219)
580	PPSubcellAlarm (p. 268)
581	PPDeviceAlarm (p. 259)
582	PPZoneAlarm (p. 269)
583	PPDeviceInput (p. 260)
584	PPSetDeviceOutput (p. 266)
585	PPQueryInterface (p. 265)
586	PPInterfaceOnline (p. 264)
587	PPInterfaceOffline (p. 264)
588	PPDeviceOnline (p. 262)
589	PPDeviceOffline (p. 261)

GEUTEBRÜCK

C Change history

Date	Version	Build	Changes
18.11.2004	2.19	11.4	first public release with UNICODE support

19.12.2004	2.20	16.0	new actions
			`SetLED`
			`SetLEDBlink`
			`SetWatchdog`
			complex types for media channels and time ranges

23.01.2005	2.21	21.0	new actions
			`SensorVideoAlarm`
			`SensorAlarmFinished`
			`SensorInhibitVideoAlarm`
			`SensorInhibitAlarmFinished`

25.02.2005	2.22	29.0	new action `SensorAudioAlarm` unused actions removed
16.03.2005	2.23	40.0	changed parameters in `SystemSettingsChanged`
05.04.2005	2.24	41.0	string resources changed

31.05.2005	2.25	62.0	changed parameters in
			`UserLoginFailed`
			`SystemError`
			`SystemWarning`
			new actions
			`SystemInfo`
			`ChangeADParameterSet`
			`ChangeVMDParameterSet`
			`DevicePluginError`
			`DevicePluginState`

10.06.2005	2.26	69.0	changed parameters in
			`SystemError`
			`SystemWarning`
			`SystemInfo`

Date	Version	Build	Changes

10.10.2005	2.27	110.0	new actions
			`AutoFocusOn`
			`AutoFocusOff`
			`ViewerConnectLive`
			`ViewerConnect`
			`ViewerClear`
			`ViewerConnected`
			`ViewerCleared`
			`ViewerPlayModeChanged`

25.04.2006	2.28	186.1	new actions
			`GEMOSalarm`
			`ViewerShowAlarmByInstance`
			`ViewerShowAlarmByKey`
			parameter `TimeStamp` removed from viewer actions.

19.05.2006	2.29	186.6	Default enum value for `ViewerConnect` changed.

25.07.2006	2.30	192	following actions have additional parameters
			`SensorInhibitVideoAlarm`
			`SensorVideoAlarm`
			`StartEvent`
			`EventStarted`
			Use Ex-functions to access new parameters. Old create and decode functions are still available, but marked as depricated.
			new sensor parameter set actions
			`ChangeCPAParameterSet`
			`ChangeObtrackParameterSet`
			new OBTRACK actions
			`ObtrackFrameRawData`
			`ObtrackObjectRawData`

GEUTEBRÜCK

Date	Version	Build	Changes
25.07.2006	2.30	192	new viewer actions ViewerChangeScene ViewerClearScene ViewerJumpByTime ViewerSyncAudioAndVideo ViewerSetAudioLevel ViewerMaximize ViewerExportPicture ViewerPrintPicture VCViewerSelect VCShowViewerText VCFullMode VCStretchMode new backup actions StartAutoBackup BackupEvent AutoBackupScheduleStarted AutoBackupScheduleDone AutoBackupOperationStarted AutoBackupOperationDone AutoBackupFileStarted AutoBackupFileProgress AutoBackupFileDone

Date	Version	Build	Changes
02.08.2006	2.31	197	OBTRACK actions have additional parameter ObtrackFrameRawData ObtrackObjectRawData new viewer actions ViewerChangeSceneByName ViewerClearSceneByName new backup actions AutoBackupCapacityMonitoringOutOfDiskSpace AutoBackupCapacityMonitoringCapacityWarning AutoBackupCapacityMonitoringFileAutoDeleted
31.08.2006	2.32	205	OBTRACK action has additional parameter ObtrackObjectRawData
18.09.2006	2.33	215	new event actions KillEvent KillEventByID KillAllEvents
18.09.2006	2.33	215	new backup actions AbortAutoBackup AbortAllAutoBackups
18.09.2006	2.33	215	new viewer actions ViewerTextOutput ViewerClearTextOutput
18.09.2006	2.33	215	new viewer actions VCAlarmQueueSelect VCAlarmQueueSelectByType VCAlarmQueueConfirm VCAlarmQueueConfirmByType
18.09.2006	2.33	215	new database actions DatabaseRecordingInfoTotal DatabaseRecordingInfoRing

GEUTEBRÜCK

Date	Version	Build	Changes
18.09.2006	2.33	215	actions renamed ViewerSetAudioLevel → VCSetAudioLevel VCViewerSelect → ViewerSelect ViewerChangeSceneByName → VCChangeSceneByName ViewerClearSceneByName → VCClearSceneByName
25.09.2006	2.34	219	new database action DatabaseStarted
25.09.2006	2.34	219	new backup actions EventBackupStarted EventBackupDone EventBackupFileStarted EventBackupFileProgress EventBackupFileDone
27.09.2006	2.35	220	new parameters in actions DatabaseRecordingInfoTotal DatabaseRecordingInfoRing
29.09.2006	2.36	220	new parameters in action ObtrackObjectRawData
29.09.2006	2.36	220	new actions VCAlarmQueueRemove VCAlarmQueueRemoveByType
04.10.2006	2.37	220	parameter changed in action DatabaseRecordingInfoRing
20.10.2006	2.38	223	parameter changed in action BackupEvent
31.10.2006	2.39	231	parameter changed in action BackupEvent
12.12.2006	2.40	242	parameter changed in action ObtrackObjectRawData

Date	Version	Build	Changes

02.03.2007	2.41	259	new parameter in action
			`BackupEvent`
			new enum values in types
			`SystemLED`
			`VideoSensorKind`

23.03.2007	2.42	269	new actions
			`KeyPressed`
			`KeyReleased`

14.05.2007	2.43	315	new parameters in actions
			`SensorVideoAlarm`
			`SensorInhibitVideoAlarm`
			new actions
			`SetupUploadProgress`
			`VCSceneChanged`
			`ViewerPlayFromTime`
			new enum values in type
			`PlcMessageCode`
			new enum type
			`VMDCycle`

12.07.2007	2.44	317	new actions
			`SafebagOpen`
			`SafebagData`
			`SafebagClose`
			new enum type
			`SafebagStep`
			some enums have different type id

GEUTEBRÜCK

Date	Version	Build	Changes
09.08.2007	2.45	340	new actions TunnelAlarm ObtrackChannelCounter ObtrackGroupCounter ObtrackChannelCounterThreshold ObtrackGroupCounterThreshold ObtrackChannelSetCounter ObtrackGroupSetCounter MoveToAbsolutePosition MoveToRelativePosition MoveToBySpeed new enum type PlcObtrackExceedingDirection PlcTunnelAlarmReason
24.08.2007	2.46	355	parameter type changed in actions MoveToAbsolutePosition MoveToRelativePosition MoveToBySpeed
28.09.2007	2.47	421	new actions ObtrackTunnelAlarm
08.11.2007	2.48	503	new actions MakeCPAReferenceImage IASSettingsChanged ABCConnect ABCDisconnect
15.11.2007	2.49	505	new parameter in action ObtrackTunnelAlarm
30.11.2007	2.50	507	text messages fixed for action IASSettingsChanged

Date	Version	Build	Changes

09.02.2008	2.51	508	new tele control actions
			`CameraManualIrisOn`
			`CameraManualIrisOff`
			`CameraTextOn`
			`CameraTextOff`
			new CPA action
			`CPAMeasurement`
			new viewer actions
			`VCAlarmQueueSelectByInstance`
			`VCAlarmQueueConfirmByInstance`
			`VCAlarmQueueRemoveByInstance`
			`VCAlarmQueueNotification`
			`ViewerSetPlayMode`
			new/changed enum types
			`PlcViewerPlayMode`
			`PlcViewerAlarmState`

23.02.2008	2.52	514	new enum type
			`PlcObtrackCounterType`
			additional parameter `CounterType` in
			`ObtrackChannelCounter`
			`ObtrackChannelCounterThreshold`
			`ObtrackChannelSetCounter`
			additional parameter `ChannelTime` in
			`ViewerPlayModeChanged`
			new notification action
			`ImageExportNotification`

GEUTEBRÜCK

Date	Version	Build	Changes
26.02.2008	2.53	520	new enum type PlcClientType additional parameter in UserLogin UserLoginFailed UserLogout ViewerConnected ViewerCleared ViewerPlayModeChanged ImageExportNotification VCAlarmQueueNotification new actions SetupChanged ATMRawData
11.03.2008	2.54	522	new ATM actions ATMRawData ATMRawAnswer ATMTransaction
02.04.2008	2.55	536	new enum values in types PlcViewerAlarmState VideoSensorKind new NPR actions ActivateExternalProcess NPRRawData NPRRecognition
16.04.2008	2.56	542	additional parameter in ObtrackGroupCounter ObtrackGroupCounterThreshold ObtrackGroupSetCounter

Date	Version	Build	Changes

29.04.2008	2.57	554	additional enum values in
			`PlcMessageSource`
			`PlcMessageCode`
			new enum type
			`PlcFRCNotification`
			new action
			`FRCNotification`
			additional parameter in
			`DatabaseRecordingInfoTotal`
			`DatabaseRecordingInfoRing`

24.06.2008	2.58	592	parameters changed in
			`DatabaseRecordingInfoTotal`
			`DatabaseRecordingInfoRing`
			additional parameter in
			`ObtrackGroupCounter`
			`ObtrackGroupCounterThreshold`
			`ObtrackGroupSetCounter`
			new actions
			`ABCPlayFile`
			`VideoSetImageContrast`
			`VideoSetImageBrightness`
			`VideoSetImageSaturation`
			`MediaChannelSetupInfo`
			`TransferBinaryBuffer`
			`TransferBinaryChannelBuffer`
			`SetSystemTime`

GEUTEBRÜCK

Date	Version	Build	Changes
19.09.2008	2.59	707	parameters changed in SetupChanged new actions POSStatus POSData new enum types PlcResourceKind PlcResourceChangeKind PlcResourceKind
15.10.2008	2.60	708	new ACS actions ACSRawData ACSRawAnswer ACSAccessGranted ACSAccessDenied
19.12.2008	2.61	710	new actions ChangeCameraProfile SMRPViewerConnected SMRPViewerCleared
09.03.2009	2.62	752	new actions ChannelInfo ChannelWarning ChannelError SMTPMailSend parameters changed in NPRRawData NPRRecognition
11.03.2009	2.63	756	additional sensor types
06.05.2009	2.64	789	new enum values
03.08.2009	2.65	790	new action IPCameraRawCommand

Date	Version	Build	Changes
14.08.2009	2.66	804	new action IOI43WDActivate IOI43WDDeactivate IOI43WDTrigger IOI43ResetMainboard IOI43Temperature
16.10.2009	2.66	804	new action BarcodeData
18.06.2010	2.67	874	new actions SafebagPassingOfRiskStart SafebagPassingOfRiskData SafebagPassingOfRiskStop InterfaceRawData InterfaceRawAnswer FillingPumpStatus TerminalArticleData TerminalPaymentData
12.09.2010	2.68	880	new enum values
16.12.2010	2.69	880	new actions CameraDayNightMode CameraBacklightCompensationMode
10.01.2011	2.70	880	new actions SkidataControl SkidataEntry SkidataExit SkidataTransaction

GEUTEBRÜCK

Date	Version	Build	Changes
22.02.2011	2.71	880	new actions SetExportMarker InitializeRemoteExport StartRemoteExport ExportProgress ExportFinished CancelExport
05.05.2011	2.72	900.17	new enum values for backup format new SKIDATA action SkidataDeviceEvent
05.05.2011	2.73	900.17	new LPS actions LPSPositionData LPSQueryPosition
20.07.2011	2.74	900.19	new Logistic actions LogBarcodeData LogBarcodeDataLPS LogNPRRecognition
11.08.2011	2.75	900.19	new Lenel action LenelAccessEvent
26.09.2011	2.76	900.22	new remote export actions SceneStoreModification StartSceneStore
26.09.2011	2.77	900.22	new lenel actions LenelSecurityEvent LenelAccessEvent LenelFireEvent LenelIntercomEvent LenelVideoEvent LenelRawData
02.10.2011	2.78	900.22	new lenel action LenelRefreshNames

Date	Version	Build	Changes
17.10.2011	2.79	900.30	lenel actions modified new action `SetClock`

29.02.2012	2.81	900.56	new actions
			`BlockingFilterActivate`
			`BlockingFilterDeactivate`

02.03.2012	2.82	900.56	new actions
			`RedundantPowerFailure`
			`RedundantPowerOk`

22.05.2012	2.83	910.1	new G-Tect actions
			`GTectVMXAlarm`
			`GTectVMXAlarmFinished`
			`GTectDualSensorAlarm`
			`GTectDualSensorAlarmFinished`
			`ChangeGTectParameterSet`

29.05.2012	2.84	910.1	new IPFailOver actions
			`IPCamFailoverNotification`
			`ChangeGTectParameterSet`

26.06.2012	2.85	910.13	new profile action `CameraApplyProfile`

06.07.2012	2.86	910.15	new generic device actions
			`WatchdogActivate`
			`WatchdogDeactivate`
			`WatchdogTrigger`
			`Temperature`
			`ResetMainboard`
			`CaseOpened`
			`CaseClosed`

GEUTEBRÜCK

Date	Version	Build	Changes
18.07.2012	2.87	910.19	new and modified G-Tect actions GTectVMXAlarm GTectVMXAlarmFinished GTectDualSensorAlarm GTectDualSensorAlarmFinished GTectSceneAlarm GTectSceneAlarmFinished GTectAnalyticsLiveCheck ChangeGTectParameterSet SetClientVCA new GSCS actions GSCSVehicleAccessDenied GSCSVehicleAccessGranted GSCSVehicleAccessExpired
07.02.2013	2.89	920.21	new Imex actions ImexCapacityWarning ImexCapacityOutOfDiskSpace ImexCapacityFileAutoDeleted ImexExportImageFromDB ImexExportEventImage ImexExportImageFromLiveStream
11.03.2013	2.90	930.4	new SCS actions GSCSVehicleAccessPending modified SCS actions GSCSVehicleAccessDenied GSCSVehicleAccessGranted GSCSVehicleAccessExpired

Date	Version	Build	Changes
11.03.2013	2.91	930.4	new perimeter protection actions PPSubcellAlarm PPDeviceAlarm PPZoneAlarm PPDeviceInput PPSetDeviceOutput PPQueryInterface PPInterfaceOnline PPInterfaceOffline PPDeviceOnline PPDeviceOffline
12.04.2013	2.92	940.4	modified SCS actions GSCSVehicleAccessPending new VCA actions VCASetArmed VCAStatusRequest VCAStatusAnswer
25.06.2013	2.93	940.24	modified backup actions AutoBackupOperationDone EventBackupDone
25.06.2013	2.94	940.24	new IP switch action IPSwitchOperation
03.09.2013	2.94	940.70	modified LPS actions LogBarcodeData LogBarcodeDataLPS

GEUTEBRÜCK

GEUTEBRÜCK

D Obsolete and replaced actions

ViewerSetAudioLevel

This action is replaced by `VCSetAudioLevel`.

VCViewerSelect

This action is replaced by `ViewerSelect`.

ViewerChangeSceneByName

This action is replaced by `VCChangeSceneByName`.

ViewerClearSceneByName

This action is replaced by `VCClearSceneByName`.

GEUTEBRÜCK

GEUTEBRÜCK

Index

GEUTEBRÜCK