XDigiLiveCore Interface

For Digifort 7.1.0.0

Index

Part I

1 Overview

1.1 Description

This document specifies the interface of XDigiLiveCore ActiveX control.

This ActiveX control allows the live view of cameras and maps and provides methods to externally control PTZ and should be used for high-level integration where live view and PTZ control is required.

This is the core control (Video Pane) and does not implement user interface for PTZ control, instead it publishes methods for controlling PTZ and manipulating objects on screen.

1.2 History

Version	Date	Revision	Comments
7.1.0.0	2015-Oct-1	Éric Fleming Bonilha	First version

Part II

2 Data types

This section describes the data types used throughout the ActiveX control.

2.1 Enumerated types (enums)

Enter topic text here.

2.1.1 TxScreenViewVisibility

Defines the visibility of a screen view.

TxScreenViewVisibility

Value	Description
svPrivate	Private screen view (Private to the user only)
svPublic	Public screen view (Accessible by any user)

2.2 Constants

This section describes the constants used by the ActiveX control.

2.2.1 Object types

Constants used to identify types of objects (Bitmask constants, values are in Hexadecimal notation)

Constant	Value
OBJECTTYPE_NONE	0x0
OBJECTTYPE_SCREENSTYLE	0x1
OBJECTTYPE_USER_SCREENVIEW	0x2
OBJECTTYPE_PUBLIC_SCREENVIEW	0x4
OBJECTTYPE_CAMERA	0x8
OBJECTTYPE_MAP	0x10

Part III

3 Interface

This section describes the ActiveX interface.

3.1 Versioning

Methods related to versioning.

3.1.1 GetVersion

Method used to retrieve control version.

Method parameters:

Parameter	Type	Description	Direction
Version	TxVersion	Version data	Output

TxVersion struct

Field	Type	Description
Major	Integer	Major version of control
Minor	Integer	Minor version of control
Release	Integer	Release version of control
Bugfix	Integer	Bugfix version of control
Build	Integer	Build number
ReleaseDate	DateTime	Date of release
ReleaseType	Integer	Type of release
TestVersion	Integer	Number of test version

Example:

```
TxVersion VersionData;

GetVersion(&VersionData);
```

3.2 Connection management

This section describes the methods and properties to manage connections with the servers.

3.2.1 Methods

This section describes the methods for server connection management.

3.2.1.1 Connect

This method is used to connect to servers (Multiple servers can be specified).

Method parameters:

Parameter	Type	Description	Direction
ConnectionString	String	String containing connection data to servers Refer to connection string format below	Input
ObjectTypes	Integer	Bitmask containing which object types the system must download Refer to object types below	Input

AutoReconnect	Boolean	Auto reconnect to servers if connection is down	Input
WaitForConnection	Boolean	When set to TRUE, the method will wait until connection is finished before returning control to the application (May hang the application for a while) When set to FALSE, connection will be made asynchronously and an event will be triggered when connection process is done	Input

Method result:

Boolean	Description
TRUE	Servers connected successfully
FALSE	Error upon connecting to servers (Only valid for WaitForConnection=TRUE)

Connection string:
The control can connect to multiple servers. In order to specify the connection data (IP, Username, Password..) you must build a connection string that should have the following syntax:

```
NA:SERVER_NAME,AD:ADDRESS,PO:PORT_NUMBER,US:USERNAME,PW:PASSWORD,CM:MODE
```

Being:

Parameter	Description
NA	Server name (Server name will be used consistently throughout the interface)
AD	Server address (IP address)
PO	Server port
US	Username in BASE64 format
PW	Password in BASE64 format
CM	Connection mode: 1 - Internal connection (Local networks) 2 - External connection (External networks, Internet)

In order to connect to multiple servers you have to building multiple connection strings and concatenate with semicolon:

```
CONNECTION_STRING1;CONNECTION_STRING2;CONNECTION_STRING3
```

Object Types:
You have the ability to control which types of objects the control must download, this can speedup connection time if you just need certain types of objects (Like Cameras) to be displayed.

Object types uses the constants defined in the constants section.

Since object types is a bitmask value, you can combine multiple types to specify which ones should be downloaded.

Example 1: Connect to a single server asynchronously, downloading cameras only

```
Connect("NA:Digifort Server,AD:192.168.10.12,PO:8600,US:YWRtaW4=,
PW:S7sdHsd=,CM:2", OBJECTTYPE_CAMERA, TRUE, FALSE);
```

Parameter	Server Info
NA	Digifort Server
AD	192.168.10.12
PO	8600
US	YWRtaW4=
PW	PW:S7sdHsd=
CM	2

Example 2: Connect to two different servers asynchronously, downloading cameras and maps

```
Connect("NA:Digifort Server,AD:192.168.10.12,PO:8600,US:YWRtaW4=,
PW:S7sdHsd=,CM:2;NA:Entrance,AD:192.168.10.45,PO:8600,US:YWRtaW4=,
PW:Yhsd8s7==,CM:1", OBJECTTYPE_CAMERA+OBJECTTYPE_MAP, TRUE, FALSE);
```

Parameter	Server 1 Info	Server 2 Info
NA	Digifort Server	Entrance
AD	192.168.10.12	192.168.10.45
PO	8600	8600
US	YWRtaW4=	YWRtaW4=
PW	PW:S7sdHsd=	Yhsd8s7==
CM	2	1

3.2.1.2 Disconnect

This method will disconnect from all servers

Example:

```
Disconnect;
```

3.2.2 Events

This section describes the events for server connection management.

3.2.2.1 OnConnectionStatus

During server connection this event will be triggered to notify you about current connection state

Event parameters:

Parameter	Type	Description	Direction
Server	String	Server name	Input
Status	Integer	Status code	Input
Msg	String	Message	Input / Output

Status codes	Value
CONNECTION_STATUS_NONE	0
CONNECTION_STATUS_CONNECTING	1
CONNECTION_STATUS_ERROR_CONNECTING	2
CONNECTION_STATUS_AUTHENTICATING	3
CONNECTION_STATUS_AUTHENTICATED	4
CONNECTION_STATUS_INVALID_AUTHENTICATION	5
CONNECTION_STATUS_INVALID_VERSION	6

CONNECTION_STATUS_LOGIN_CANCELLED	7
CONNECTION_STATUS_CONNECTION_CANCELLED	8
CONNECTION_STATUS_DISCONNECTED_BY_SERVER	9
CONNECTION_STATUS_INVALID_LOGIN_TIME	10
CONNECTION_STATUS_INVALID_IP_ADDRESS	11
CONNECTION_STATUS_ACCOUNT_BLOCKED	12
CONNECTION_STATUS_ACCOUNT_EXPIRED	13
CONNECTION_STATUS_SERVER_FULL*	14*
CONNECTION_STATUS_COMPLETED	15
CONNECTION_STATUS_LOGIN_LIMIT_REACHED	16
CONNECTION_STATUS_DOWNLOADING_CAMERAS	100
CONNECTION_STATUS_DOWNLOADING_SCREEN_STYLES	101
CONNECTION_STATUS_DOWNLOADING_USER_VIEWS	102
CONNECTION_STATUS_DOWNLOADING_MAP	103
CONNECTION_STATUS_DOWNLOADING_PUBLIC_VIEWS	104

* Deprecated values

The "Msg" parameter can be changed and this is the value that will be presented to the user on the connection status panel

3.2.2.2 OnConnectionComplete

This event will be triggered as soon as all servers are connected (Event if a server connection failed, this event will still be triggered)

3.3 Objects lists

Enter topic text here.

3.3.1 Methods

Methods to retrieve objects lists

3.3.1.1 GetServerCount

Query the number of servers (Added through Connect method).

Method result:
Integer - Number of servers

Example: Retrieve the number of servers
```
int ServerCount;

ServerCount = GetServerCount();
```

3.3.1.2 GetServer

Retrieve the data of a server. Use this method along with GetServerCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the record	Input
Server	TxServer	Server data	Output

TxServer struct

Field	Type	Description

Name	String	Server name
ID	String	Server ID

Method result:

Boolean	Description
TRUE	Server data was retrieved
FALSE	Server data was not retrieved

Example: Retrieve the data of server 0

```
TxServer ServerData;

if (true == GetServer(0, &ServerData) {
  ...
}
```

3.3.1.3 GetScreenStyleCount

Query the number of screen styles downloaded from servers.

This method should only be called after successfully connecting to servers.

Method result:
Integer - Number of screen styles

Example: Retrieve the number of screen styles

```
int ScreenStyleCount;

ScreenStyleCount = GetScreenStyleCount();
```

3.3.1.4 GetScreenStyle

Retrieve the data of a screen style. Use this method along with GetScreenStyleCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the record	Input
ScreenStyle	TxScreenStyle	Screen Style data	Output

TxScreenStyle struct

Field	Type	Description
ID	Integer	Style ID
Data	String	Style data

Method result:

Boolean	Description
TRUE	Screen Style data was retrieved
FALSE	Screen Style data was not retrieved

Example: Retrieve the data of screen style 0

```
TxScreenStyle ScreenStyle;

if (true == GetScreenStyle(0, &ScreenStyle) {
  ...
}
```

3.3.1.5 GetScreenViewCount

Query the number of screen views downloaded from servers.

This method should only be called after successfully connecting to servers.

Method parameters:

Parameter	Type	Description	Direction
ScreenStyleID	Integer	ID of the ScreenStyle	Input

Method result:
Number of screen views for the specified screen style

Example: Retrieve the number of views from Screen Style 1399

```
int ViewCount;

ViewCount = GetScreenViewCount(1399);
```

3.3.1.6 GetScreenView

Retrieve the data of a screen view. Use this method along with GetScreenViewCount.

Method parameters:

Parameter	Type	Description	Direction
ScreenStyleID	Integer	ID of the screenstyle	Input
Index	Integer	Index of the record	Input
ScreenView	TxScreenView	Screen View data	Output

TxScreenView struct

Field	Type	Description
Name	String	Name of the screen view
Visibility	TxScreenViewVisibility	Visibility of the screen view
Data	String	Screen view data

Method result:

Boolean	Description
TRUE	Screen view data was retrieved
FALSE	Screen view data was not retrieved

Example: Retrieve the data of screen view 0 from screen style 1399

```
TxScreenView ViewData;

if (true == GetScreenView(1399, 0, &ViewData) {
  ...
}
```

3.3.1.7 GetCameraCount

Query the number of cameras downloaded from servers.

This method should only be called after successfully connecting to servers.

Method result:
Integer - Number of cameras

Example: Retrieve the number of cameras

```
int CamCount;

CamCount = GetCameraCount();
```

3.3.1.8 GetCamera

Retrieve the data of a camera. Use this method along with GetCameraCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the record	Input
Camera	TxCamera	Camera data	Output

TxCamera struct

Field	Type	Description
Server	String	Server name
Name	String	Camera name
Description	String	Camera description
Activated	Boolean	Camera is activated or not

Method result:

Boolean	Description
TRUE	Camera data was retrieved
FALSE	Camera data was not retrieved

Example: Retrieve the data of camera 0

```
TxCamera CamData;

if (true == GetCamera(0, &CamData) {
  ...
}
```

3.3.1.9 GetMapCount

Query the number of maps downloaded from servers.

This method should only be called after successfully connecting to servers.

Method result:
Integer - Number of maps

Example: Retrieve the number of maps

```
int MapCount;

MapCount = GetMapCount();
```

3.3.1.10 GetMap

Retrieve the data of a map. Use this method along with GetMapCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the record	Input
Map	TxMap	Map data	Output

TxMap struct

Field	Type	Description
Server	String	Server name
Name	String	Map name
Description	String	Map description

Method result:

Boolean	Description
TRUE	Map data was retrieved
FALSE	Map data was not retrieved

Example: Retrieve the data of map 0

```
TxMap MapData;

if (true == GetMap(0, &MapData) {
   ...
}
```

3.3.2 Events

This section describes the events for objects lists.

3.3.2.1 OnSystemObjectUpdated

This event will be fired whenever the data of an object has been updated by the administrator.

Event parameters:

Parameter	Type	Description	Direction
ObjectType	Integer	Type of object	Input
Server	String	Server name	Input
Name	String	Object name	Input
CustomData	String	Custom data	Input
UpdateType	Integer	Type of object update	Input

Constant	Value
OBJECT_UPDATE_ADDED	0
OBJECT_UPDATE_MODIFIED	1
OBJECT_UPDATE_DELETED	2

3.4 User rights

This section describes methods for querying user rights

3.4.1 Methods

Methods for user rights

3.4.1.1 GetUserRights

Method used to retrieve user rights.

Method Result:

Parameter	Description
TxUserRights	User rights struct

TxUserRights struct

Field	Type	Description
ScreenViewsUser	Boolean	Rights to save user screen views
ScreenViewsPublic	Boolean	Rights to save public screen views

Example:

```
TxUserRights UserRights;

UserRights = GetUserRights();
```

3.5 Settings

This section provides info on configuring and retrieving local control settings.

3.5.1 Methods

Methods for storing and retrieving settings.

3.5.1.1 GetConfig

Retrieve local ActiveX control settings

Method parameters:

Parameter	Type	Description	Direction
Config	Integer	Configuration type	Input
Value	Variant	Configuration value. Type of value will change according to config type, see list below for types of value	Output

Config constants	Value
CONFIG_LOCAL_RECORDING_PATH	0
CONFIG_SHOW_OBJECT_DESCRIPTION	100
CONFIG_SHOW_OBJECT_NAME	101
CONFIG_SHOW_CAMERA_RECONNECTION_MESSAGE	200
CONFIG_SHOW_CAMERA_LOCAL_RECORDING_CONTROL	201
CONFIG_SHOW_CAMERA_FRAME_RATE	202
CONFIG_SHOW_CAMERA_IMAGE_RESOLUTION	203
CONFIG_SHOW_CAMERA_TRANSFER_RATE	204
CONFIG_SHOW_CAMERA_DECODER	205
CONFIG_SHOW_CAMERA_CONNECTION_STATUS	206
CONFIG_SHOW_CAMERA_DATE	207
CONFIG_SHOW_CAMERA_TIME	208
CONFIG_RESIZE_TYPE	300
CONFIG_RESIZE_BILINEAR	301
CONFIG_VIDEO_TYPE	400
CONFIG_MOTION_ACTIVATE	500
CONFIG_MOTION_COLOR	501
CONFIG_MOTION_SENSITIVITY	502

Config Type	Output Value Type
CONFIG_LOCAL_RECORDING_PATH	String
CONFIG_SHOW_OBJECT_DESCRIPTION	Boolean
CONFIG_SHOW_OBJECT_NAME	Boolean
CONFIG_SHOW_CAMERA_RECONNECTION_MESSAGE	Boolean
CONFIG_SHOW_CAMERA_LOCAL_RECORDING_CONTROL	Boolean
CONFIG_SHOW_CAMERA_FRAME_RATE	Boolean
CONFIG_SHOW_CAMERA_IMAGE_RESOLUTION	Boolean
CONFIG_SHOW_CAMERA_TRANSFER_RATE	Boolean
CONFIG_SHOW_CAMERA_DECODER	Boolean
CONFIG_SHOW_CAMERA_CONNECTION_STATUS	Boolean
CONFIG_SHOW_CAMERA_DATE	Boolean
CONFIG_SHOW_CAMERA_TIME	Boolean
CONFIG_RESIZE_TYPE	Integer
CONFIG_RESIZE_BILINEAR	Boolean
CONFIG_VIDEO_TYPE	Integer
CONFIG_MOTION_ACTIVATE	Boolean
CONFIG_MOTION_COLOR	Integer
CONFIG_MOTION_SENSITIVITY	Integer

Resize type values (Used with CONFIG_RESIZE_TYPE)	Value
CONFIG_RESIZE_TYPE_DO_NOT_RESIZE	0
CONFIG_RESIZE_TYPE_STRETCH	1
CONFIG_RESIZE_TYPE_PROPORTIONAL	2

Video type values (Used with CONFIG_VIDEO_TYPE)	Value
CONFIG_VIDEO_TYPE_GDI	0
CONFIG_VIDEO_TYPE_DIRECTDRAW	1

Method result:

Boolean	Description
TRUE	Config value was retrieved
FALSE	Error retrieving config value

Example: Retrieve the type of resize

```
int ResizeType;

if (true == GetConfig(CONFIG_RESIZE_TYPE, ResizeType)) {
  ...
}
```

3.5.1.2 SetConfig

Set local ActiveX control settings

Method parameters:

Parameter	Type	Description	Direction
Config	Integer	Configuration type **Config constants** — **Value** CONFIG_LOCAL_RECORDING_PATH — 0	Input

CONFIG_SHOW_OBJECT_DESCRIPTION	100
CONFIG_SHOW_OBJECT_NAME	101
CONFIG_SHOW_CAMERA_RECONNECTION_MESSAGE	200
CONFIG_SHOW_CAMERA_LOCAL_RECORDING_CONTROL	201
CONFIG_SHOW_CAMERA_FRAME_RATE	202
CONFIG_SHOW_CAMERA_IMAGE_RESOLUTION	203
CONFIG_SHOW_CAMERA_TRANSFER_RATE	204
CONFIG_SHOW_CAMERA_DECODER	205
CONFIG_SHOW_CAMERA_CONNECTION_STATUS	206
CONFIG_SHOW_CAMERA_DATE	207
CONFIG_SHOW_CAMERA_TIME	208
CONFIG_RESIZE_TYPE	300
CONFIG_RESIZE_BILINEAR	301
CONFIG_VIDEO_TYPE	400
CONFIG_MOTION_ACTIVATE	500
CONFIG_MOTION_COLOR	501
CONFIG_MOTION_SENSITIVITY	502

Value	Variant	Configuration value. Type of value will change according to config type, see list below for types of value	Input

Config Type	Input Value Type
CONFIG_LOCAL_RECORDING_PATH	String
CONFIG_SHOW_OBJECT_DESCRIPTION	Boolean
CONFIG_SHOW_OBJECT_NAME	Boolean
CONFIG_SHOW_CAMERA_RECONNECTION_MESSAGE	Boolean
CONFIG_SHOW_CAMERA_LOCAL_RECORDING_CONTROL	Boolean
CONFIG_SHOW_CAMERA_FRAME_RATE	Boolean
CONFIG_SHOW_CAMERA_IMAGE_RESOLUTION	Boolean
CONFIG_SHOW_CAMERA_TRANSFER_RATE	Boolean
CONFIG_SHOW_CAMERA_DECODER	Boolean
CONFIG_SHOW_CAMERA_CONNECTION_STATUS	Boolean
CONFIG_SHOW_CAMERA_DATE	Boolean
CONFIG_SHOW_CAMERA_TIME	Boolean
CONFIG_RESIZE_TYPE	Integer
CONFIG_RESIZE_BILINEAR	Boolean
CONFIG_VIDEO_TYPE	Integer
CONFIG_MOTION_ACTIVATE	Boolean
CONFIG_MOTION_COLOR	Integer
CONFIG_MOTION_SENSITIVITY	Integer

Resize type values (Used with CONFIG_RESIZE_TYPE)	Value
CONFIG_RESIZE_TYPE_DO_NOT_RESIZE	0
CONFIG_RESIZE_TYPE_STRETCH	1
CONFIG_RESIZE_TYPE_PROPORTIONAL	2

Video type values (Used with CONFIG_VIDEO_TYPE)	Value
CONFIG_VIDEO_TYPE_GDI	0
CONFIG_VIDEO_TYPE_DIRECTDRAW	1

Method result:

Boolean	Description
TRUE	Config value was changed
FALSE	Error setting config

Example 1: Set resize type to stretch

```
if (true == SetConfig(CONFIG_RESIZE_TYPE, CONFIG_RESIZE_TYPE_STRETCH)) {
  ...
}
```

Example 2: Set video type to GDI

```
if (true == SetConfig(CONFIG_VIDEO_TYPE, CONFIG_VIDEO_TYPE_GDI)) {
  ...
}
```

Example 3: Set control to show object description on title

```
if (true == SetConfig(CONFIG_SHOW_OBJECT_DESCRIPTION, true)) {
  ...
}
```

3.5.1.3 GetContextMenus

Retrieve which context menus should be displayed for a given type of object.

Method parameters:

Parameter	Type	Description	Direction
ObjectType	Integer	Type of object	Input

Constant	Value
OBJECTTYPE_CAMERA	0x8
OBJECTTYPE_MAP	0x10

Method result:
Integer bitmask containing all active context menus.

Context menu for Cameras	Value
CONTEXT_MENU_CAMERA_REMOVE	0x1
CONTEXT_MENU_CAMERA_MEDIA_PROFILE	0x2
CONTEXT_MENU_CAMERA_MOTION_DETECTION	0x4
CONTEXT_MENU_CAMERA_FILTERS	0x8
CONTEXT_MENU_CAMERA_PTZ	0x10
CONTEXT_MENU_CAMERA_SCREENSHOT	0x20

Context menu for Maps	Value
CONTEXT_MENU_MAP_REMOVE	0x1

Example: Retrieve context menus for cameras and check if PTZ menu is active

```
int ContextMenus;

ContextMenus = GetContextMenus(OBJECTTYPE_CAMERA);
```

```
if (CONTEXT_MENU_CAMERA_PTZ == (ContextMenus & CONTEXT_MENU_CAMERA_PTZ)) {
  ...
}
```

3.5.1.4 SetContextMenus

Set which context menus should be displayed for a given type of object.

Method parameters:

Parameter	Type	Description	Direction

ObjectType	Integer	Type of object
		
		Constant / Value:
		OBJECTTYPE_CAMERA = 0x8
		OBJECTTYPE_MAP = 0x10
			Input

Menus	Integer	Bitmask with combination of menus to display	Input

Constant	Value
OBJECTTYPE_CAMERA	0x8
OBJECTTYPE_MAP	0x10

Context menu for Cameras	Value
CONTEXT_MENU_CAMERA_REMOVE	0x1
CONTEXT_MENU_CAMERA_MEDIA_PROFILE	0x2
CONTEXT_MENU_CAMERA_MOTION_DETECTION	0x4
CONTEXT_MENU_CAMERA_FILTERS	0x8
CONTEXT_MENU_CAMERA_PTZ	0x10
CONTEXT_MENU_CAMERA_SCREENSHOT	0x20

Context menu for Maps	Value
CONTEXT_MENU_MAP_REMOVE	0x1

Example: Set camera objects to show only "Remove" and "PTZ" context menus

```
int ContextMenus;

ContextMenus = CONTEXT_MENU_CAMERA_REMOVE | CONTEXT_MENU_CAMERA_PTZ;

SetContextMenus(OBJECTTYPE_CAMERA, ContextMenus);
```

3.6 Objects Matrix

This section defines the methods and properties to control the object layout matrix.

3.6.1 Methods

This section describes the methods to control the camera layout matrix.

3.6.1.1 GetMatrixSpotCount

Return the number of spots on current matrix.

Method result:
Integer - Number of layout spots on current matrix style

Example: Retrieve the number of spots

```
int SpotCount;

```

```
SpotCount = GetMatrixSpotCount();
```

3.6.1.2 GetMatrixObject

Retrieve the data of an object that is being displayed on matrix. Use this method along with GetMatrixSpotCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on matrix	Input
MatrixObject	TxMatrixObject	Live object data	Output

TxMatrixObject struct

Field	Type	Description
ObjectType	Integer	Type of object
Server	String	Server ID
Name	String	Object name

Method result:

Boolean	Description
TRUE	Object data was retrieved
FALSE	Error retrieving object data

Note:
If matrix spot has no object, this method will return OBJECTTYPE_NONE on MatrixObject.ObjectType and result will be TRUE.

Example: Retrieve the data of object on spot 0

```
TxMatrixObject ObjData;

if (true == GetMatrixObject(0, &ObjData) {
  ...
}
```

3.6.1.3 MatrixAddObject

Add a new object on the screen matrix.

Method parameters:

Parameter	Type	Description	Direction
ObjectType	Integer	Type of object to add	Input
Server	String	Server name	Input
Name	String	Object name	Input
Spot	Integer	Spot index (Starting at 0). -1 to add on next available spot	Input

Method result:

Boolean	Description
TRUE	Object was added to the screen
FALSE	Error adding object

Example 1: Add camera "Entrance" from server "Local" on spot 1

```
if (true == MatrixAddObject(OBJECTTYPE_CAMERA, "Local", "Entrance", 1) {
  ...
}
```

Example 2: Add map "Overview" from server "Local" on next available spot

```
if (true == MatrixAddObject(OBJECTTYPE_MAP, "Local", "Overview", -1) {
  ...
}
```

3.6.1.4 MatrixRemoveObject

Remove an object from matrix.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Object index to remove	Input

Example 1: Remove objects from spots 0, 1 and 2

```
MatrixRemoveObject(0);
MatrixRemoveObject(1);
MatrixRemoveObject(2);
```

3.6.1.5 MatrixRemoveObjects

Remove all objects from matrix

Example 1: Remove objects

```
MatrixRemoveObjects();
```

3.6.1.6 MatrixLoadScreenView

Load a previously saved layout view on matrix.

Method parameters:

Parameter	Type	Description	Direction
ViewName	String	Name of the view to load	Input
Visibility	TxScreenViewVisibility	Screen view visibility	Input

Example 1: Load private view "Entrance"

```
MatrixLoadScreenView("Entrance", svPrivate);
```

Example 2: Load public view "All cameras"

```
MatrixLoadScreenView("All cameras", svPublic);
```

3.6.1.7 MatrixSaveScreenView

Save the current position of objects on matrix as a new layout view.

Method parameters:

Parameter	Type	Description	Direction
Name	String	Name for the new view	Input
Visibility	TxScreenViewVisibility	Visibility of the new view	Input
Data	String	Custom data - Reserved for internal use only	Input
SaveType	Integer	Type of view being saved	Input

Constant	Value

		SAVEVIEW_TYPE_NORMAL 0 SAVEVIEW_TYPE_TIMER 1	

Method result:

Constant	Value
SAVEVIEW_OK	0
SAVEVIEW_ERROR	1
SAVEVIEW_ERROR_NO_OBJECTS	2
SAVEVIEW_ERROR_NO_NAME	3
SAVEVIEW_ERROR_NO_SCREENSTYLE	4
SAVEVIEW_ERROR_NO_RIGHTS	5
SAVEVIEW_ERROR_INVALID_TYPE	6

Example: Save current objects as a new public view called "Entrance"

```
int SaveResult;

SaveResult == MatrixSaveScreenView("Entrance", svPublic, "",
                                   SAVEVIEW_TYPE_NORMAL);

if (SAVEVIEW_OK == SaveResult) {
  ...
}
```

3.6.1.8 MatrixDeleteScreenView

Delete a layout view

Method parameters:

Parameter	Type	Description	Direction
ScreenStyle	Integer	ScreenStyle ID	Input
Name	String	Name of the view	Input
Visibility	TxScreenViewVisibility	Visibility of the view	Input

Method result:

Boolean	Description
TRUE	View was deleted
FALSE	View was not deleted

Example 1: Delete public view "Entrance" from screen style 6278

```
if (true == MatrixDeleteScreenView(6278, "Entrance", svPublic) {
  ...
}
```

3.6.1.9 MatrixTimer

Control a timer view.

Method parameters:

Parameter	Type	Description	Direction
Command	Integer	Timer view command Constant | Value	Input

		SEQ_TIMER_PLAY 0 SEQ_TIMER_PAUSE 1 SEQ_TIMER_NEXT 2 SEQ_TIMER_PREV 3	

Example 1: Pause current timer view
```
MatrixTimer(SEQ_TIMER_PAUSE);
```

Example 2: Jump to next item on timer view
```
MatrixTimer(SEQ TIMER NEXT);
```

3.6.1.10 GetMatrixMainScreenViewName

Return the name of the current main view loaded on screen.

Method result:
String - View name

Example: Retrieve the name of current main view
```
string ViewName;

ViewName = GetMatrixMainScreenViewName();
```

3.6.1.11 GetMatrixMainScreenViewVisibility

Return the visibility of the current main view loaded on screen.

Method result:
TxScreenViewVisibility - Visibility of the view

Example: Retrieve the visibility of current main view
```
TxScreenViewVisibility Visibility;

Visibility = GetMatrixMainScreenViewVisibility();
```

3.6.1.12 GetMatrixMainScreenViewData

Return the data of the current main view loaded on screen.

This method is reserved for internal use only.

Method result:
String - View data

Example: Retrieve the data of current main view
```
string ViewData;

ViewData = GetMatrixMainScreenViewData();
```

3.6.1.13 GetMatrixCurrentScreenViewName

Return the name of the current view loaded on screen.

This method is similar to GetMatrixMainScreenViewName but instead of returning the name of the main view, it will return from the current view. This is only valid for TIMER views where the main name is the name of the timer view itself and the current name is the name of the internal object that timer view is currently showing.

Method result:
String - View name

Example: Retrieve the name of current view
```
string ViewName;

ViewName = GetMatrixCurrentScreenViewName();
```

3.6.1.14 GetMatrixCurrentScreenViewVisibility

Return the visibility of the current view loaded on screen.

This method is similar to GetMatrixMainScreenViewVisibility but instead of returning the visibility of the main view, it will return from the current view.

Method result:
TxScreenViewVisibility - Visibility of the view

Example: Retrieve the visibility of current view
```
TxScreenViewVisibility Visibility;

Visibility = GetMatrixCurrentScreenViewVisibility();
```

3.6.1.15 GetMatrixCurrentScreenViewData

Return the data of the current view loaded on screen.

This method is reserved for internal use only.

Method result:
String - View data

Example: Retrieve the data of current view
```
string ViewData;

ViewData = GetMatrixCurrentScreenViewData();
```

3.6.2 Properties

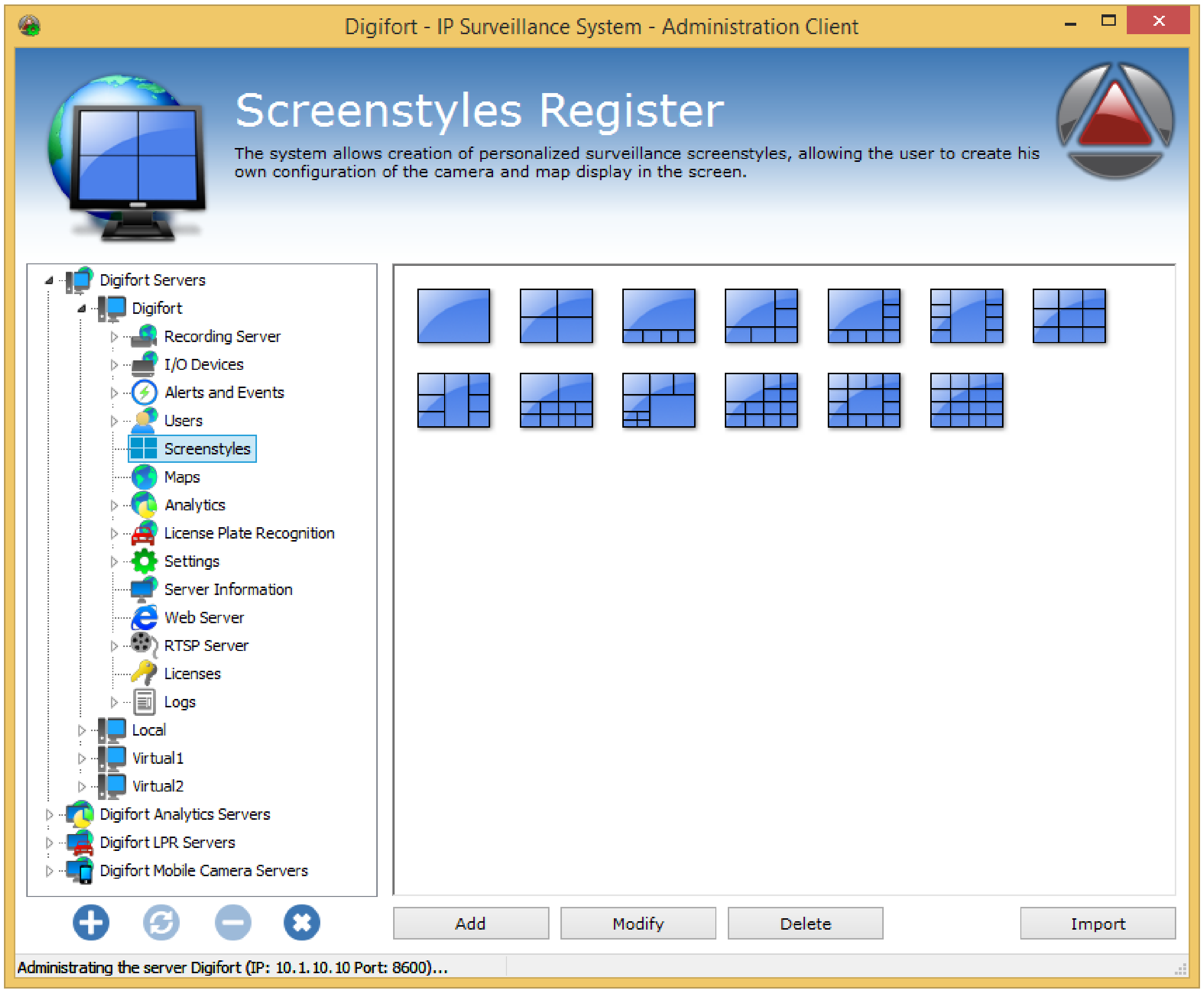
This section contains the properties to control the camera layout matrix.

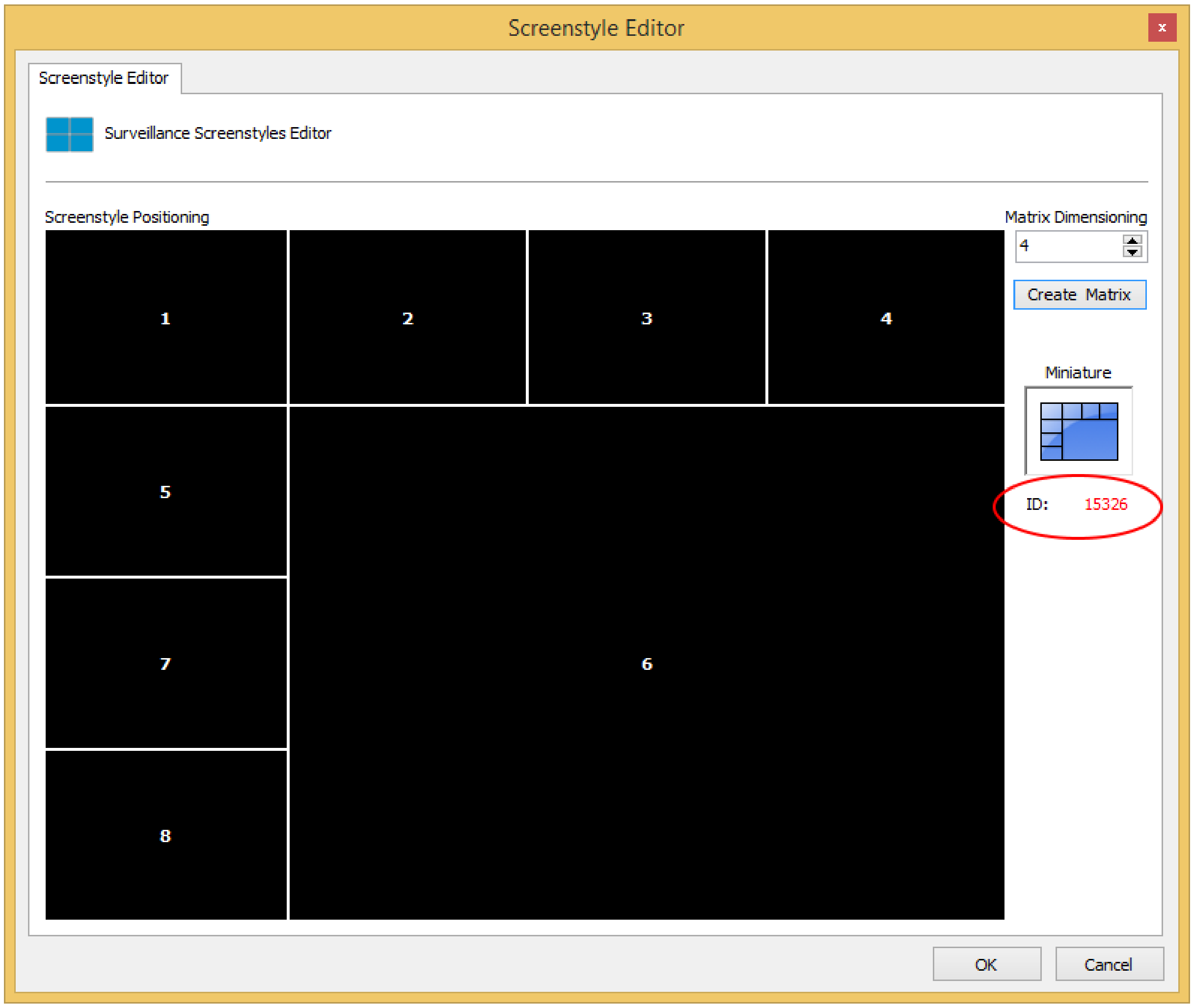
3.6.2.1 MatrixScreenStyle

This property defines the style of layout matrix.

Value: Integer;

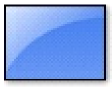
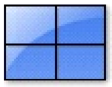
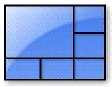
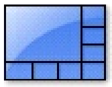
In Standard, Professional and Enterprise editions it is possible to create new styles by using the Administration Client. To recover the ID of the style double click on the desired style, as shown below:

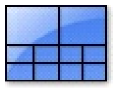

Digifort - IP Surveillance System - Administration Client
Screenstyles Register
The system allows creation of personalized surveillance screenstyles, allowing the user to create his own configuration of the camera and map display in the screen.
Digifort Servers
Digifort
Recording Server
I/O Devices
Alerts and Events
Users
Screenstyles
Maps
Analytics
License Plate Recognition
Settings
Server Information
Web Server
RTSP Server
Licenses
Logs
Local
Virtual1
Virtual2
Digifort Analytics Servers
Digifort LPR Servers
Digifort Mobile Camera Servers
Add
Modify
Delete
Import
Administrating the server Digifort (IP: 10.1.10.10 Port: 8600)...

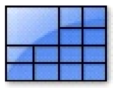


ID: 15326

In Explorer edition, only the default styles can be used, following the list below:

	Automatic screenstyle – ID: 0
	1 camera - ID: 1399
	4 cameras - ID: 6278
	6 cameras - ID: 9983
	8 cameras - ID: 13538


10 cameras - ID: 18393


13 cameras - ID: 25660

Example: Set style for 4 cameras:
```
MatrixScreenStyle = 6278;
```

3.6.2.2 MatrixSelectedIndex

Returns and set the selected object on matrix.

Value: Integer

Example 1: Get current selected object index
```
int ObjectIndex;

ObjectIndex = MatrixSelectedIndex;
```

Example 2: Select object 1 from screen (Index starts at 0)
```
MatrixSelectedIndex = 1;
```

3.6.2.3 MatrixSelectable

This property defines if matrix is user-selectable or not. If activated, user will be able to select cameras (with a bounding red box).

Value: Boolean

Example: Set matrix as not selectable
```
MatrixSelectable = false;
```

3.6.3 Events

This section describes the events for objects matrix.

3.6.3.1 OnMatrixRemoveObject

This event will be fired whenever an object is removed from the matrix (By code or by user action).

It is important to note that this event will be fired before the object is removed, giving the possibility to know which object is being removed.

Event parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on the matrix	Input

3.6.3.2 OnMatrixRemoveObjects

This event will be fired whenever all objects are being removed from objects matrix.

3.6.3.3 OnMatrixSpotSelected

This event will be fired whenever a spot from the matrix is selected by the user (By clicking on it).

Event parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the spot on the matrix	Input

3.6.3.4 OnMatrixSpotDeselected

This event will be fired whenever a spot from the matrix is deselected by the user (By clicking on it).

Event parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the spot on the matrix	Input

3.6.3.5 OnMatrixTimerOperation

This event will be fired whenever a timer view operation is performed.

Event parameters:

Parameter	Type	Description	Direction
Operation	Integer	Operation ID	Input

Constant	Value
TIMER_OP_PLAY	0
TIMER_OP_PAUSE	1
TIMER_OP_CHANGED	2

3.7 Cameras

This section describes the methods and properties to work with camera objects.

3.7.1 Methods

Methods for camera control

3.7.1.1 SaveSnapshot

Save a snapshot from a camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Format	Integer	File format	Input

Constant	Value
SNAPSHOT_FORMAT_BITMAP	1
SNAPSHOT_FORMAT_JPEG	2

FilePath	String	Full file path (With filename)	Input

Method result:

Boolean	Description
TRUE	Snapshot saved
FALSE	Error saving snapshot

Example: Save snapshot from camera 0 (Matrix index) as JPEG in "c:\temp\snap.jpg"

```
if (true == SaveSnapshot(0, SNAPSHOT_FORMAT_JPEG, "c:\temp\snap.jpg")) {
  ...
}
```

3.7.1.2 GetCameraPrivacyMode

Return if camera privacy mode is activated

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:

Boolean	Description
TRUE	Privacy mode is activated
FALSE	Privacy mode is deactivated

Example: Check if privacy mode is activated for camera 0 (Matrix index)

```
if (true == GetCameraPrivacyMode(0)) {
  ...
}
```

3.7.1.3 SetCameraPrivacyMode

Activate / Deactivate privacy mode from a camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
PrivacyMode	Boolean	Activate or Deactivate privacy mode	Input

Method result:

Boolean	Description
TRUE	Privacy mode state was set
FALSE	Error setting privacy mode state

Example: Activate privacy mode for camera 0 (Matrix index)

```
if (true == SetCameraPrivacyMode(0, true)) {
  ...
}
```

3.7.2 Events

This section describes the events related to cameras.

3.7.2.1 OnCameraPrivacyModeState

This event will be fired whenever the state of Privacy Mode of a camera has changed.

Event parameters:

Parameter	Type	Description	Direction
Server	String	Server name	Input
Camera	String	Camera name	Input
PrivacyMode	Boolean	Privacy mode state TRUE - Privacy mode is activated FALSE - Privacy mode is deactivated	Input

3.8 PTZ

This section describes the methods to control camera´s PTZ.

3.8.1 Constants

The following constants are used along with PTZ commands

PTZ Result	Value
PTZ_OK	0
PTZ_ERROR	1
PTZ_ERROR_NO_PTZ_RIGHT	2
PTZ_ERROR_LOCKED	3
PTZ_ERROR_DISABLED	4
PTZ_ERROR_NOT_SUPPORTED	5
PTZ_ERROR_INVALID_OPERATION	6
PTZ_ERROR_INVALID_INPUT	7

3.8.2 Methods

This section provides all methods for controlling PTZ.

3.8.2.1 PTZAvailable

Query if a given camera on screen matrix has PTZ control.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:

Boolean	Description
TRUE	Camera has PTZ ability
FALSE	Camera does not have PTZ ability or user does not have PTZ control rights

Example:

```
PTZAvailable(0);
```

3.8.2.2 PTZSupport

Return which PTZ commands the specified camera supports

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:
Integer bitmask containing all supported methods.

Type of PTZ operation (Bitmask)	Value
PTZ_OPERATION_NONE	0x0
PTZ_OPERATION_SIMPLE	0x1
PTZ_OPERATION_RELATIVE	0x2
PTZ_OPERATION_ABSOLUTE	0x4
PTZ_OPERATION_CONTINUOUS	0x8
PTZ_OPERATION_AUTOFOCUS	0x10
PTZ_OPERATION_AUTOIRIS	0x20
PTZ_OPERATION_MENUCONTROL	0x40
PTZ_OPERATION_CALL_PRESET	0x80
PTZ_OPERATION_SET_PRESET	0x100
PTZ_OPERATION_CALL_PATTERN	0x200
PTZ_OPERATION_WIPER	0x400
PTZ_OPERATION_AUXILIARY	0x800

Example: Query supported operations from camera 0 (Matrix index 0) and check if it has support to PTZSimple command

```
Supported = PTZSupport(0);

if (PTZ_OPERATION_SIMPLE == (Supported & PTZ_OPERATION_SIMPLE)) {
  ...
}
```

3.8.2.3 PTZContinuousSupport

Return which PTZ continuous commands the specified camera supports

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:
Integer bitmask containing all supported continuous commands

PTZ Continuous Operation (Bitmask)	Value
PTZ_CONTINUOUS_NONE	0x0
PTZ_CONTINUOUS_PAN	0x1
PTZ_CONTINUOUS_TILT	0x2
PTZ_CONTINUOUS_ZOOM	0x4
PTZ_CONTINUOUS_FOCUS	0x8
PTZ_CONTINUOUS_IRIS	0x10

Example: Query supported continuous operations from camera 0 (Matrix index 0) and check if it has support to continuous Pan

```
Supported = PTZContinuousSupport(0);
```

```
if (PTZ_CONTINUOUS_PAN == (Supported & PTZ_CONTINUOUS_PAN)) {
  ...
}
```

3.8.2.4 PTZSimple

Send simple PTZ controls to the specified camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform.	Input
Speed	Integer	Speed of the operation (From 1 to 100)	Input

Simple PTZ operation	Value
PTZ_SIMPLE_MOVE_LEFT	0
PTZ_SIMPLE_MOVE_RIGHT	1
PTZ_SIMPLE_MOVE_UP	2
PTZ_SIMPLE_MOVE_DOWN	3
PTZ_SIMPLE_MOVE_UP_LEFT	4
PTZ_SIMPLE_MOVE_UP_RIGHT	5
PTZ_SIMPLE_MOVE_DOWN_LEFT	6
PTZ_SIMPLE_MOVE_DOWN_RIGHT	7
PTZ_SIMPLE_HOME	8
PTZ_SIMPLE_ZOOM_TELE	9
PTZ_SIMPLE_ZOOM_WIDE	10
PTZ_SIMPLE_FOCUS_NEAR	11
PTZ_SIMPLE_FOCUS_FAR	12
PTZ_SIMPLE_IRIS_OPEN	13
PTZ_SIMPLE_IRIS_CLOSE	14

Method result:
PTZ Result

Example: Move camera 0 (Matrix index) to LEFT with speed of 80

```
int OperationResult;

OperationResult = PTZSimple(0, PTZ_SIMPLE_MOVE_LEFT, 80);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.5 PTZRelative

Send relative PTZ controls to the specified camera. Relative control will move the camera to the specified direction by N degrees.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform	Input

Relative PTZ operation	Value
PTZ_RELATIVE_PAN	0
PTZ_RELATIVE_TILT	1
PTZ_RELATIVE_ZOOM	2
PTZ_RELATIVE_FOCUS	3
PTZ_RELATIVE_IRIS	4

Value	Integer	Degrees to move	Input

Operation	Values of the operation
Pan	-360 to 360 Negative values = Left Positive values = Right
Tilt	-360 to 360 Negative value = Up Positive values = Down
Zoom	-100 to 100 Negative values = Zoom Out Positive values = Zoom In
Focus	-100 to 100 Negative values = Focus Near Positive values = Focus Far
Iris	-100 a 100 Negative values = Close Iris Positive values = Open Iris

Speed	Integer	Speed of the operation (From 1 to 100)	Input

Method result:
PTZ Result

Example: Move camera 0 (Matrix index) to LEFT by 50 degrees with speed of 80

```
int OperationResult;

OperationResult = PTZRelative(0, PTZ_RELATIVE_PAN, -50, 80);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.6 PTZAbsolute

Send absolute PTZ controls to the specified camera. Absolute control will move the camera to the specified position in degrees

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform	Input

Absolute PTZ operation	Value
PTZ_ABSOLUTE_PAN	0
PTZ_ABSOLUTE_TILT	1
PTZ_ABSOLUTE_ZOOM	2
PTZ_ABSOLUTE_FOCUS	3
PTZ_ABSOLUTE_IRIS	4

Parameter	Type	Description	Direction
Value	Integer	Degrees to move	Input

Operation	Values of the operation
Pan	-180 to 180 Negative values = Left Positive values = Right
Tilt	-180 to 180 Negative values = Up Positive values = Down
Zoom	1 to 100
Focus	1 to 100
Iris	1 to 100

Parameter	Type	Description	Direction
Speed	Integer	Speed of the operation (From 1 to 100)	Input

Method result:
PTZ Result

Example: Move PAN of camera 0 (Matrix index) to 70 degrees with speed of 80

```
int OperationResult;

OperationResult = PTZAbsolute(0, PTZ_ABSOLUTE_PAN, 70, 80);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.7 PTZContinuous

Send continuous PTZ controls to the specified camera. Continuous commands are used to control the camera with operations like Joystick.

Use PTZContinuousSupport method to check which operations are supported by the camera.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Pan	Integer	Speed of pan movement ranging from -100 to 100	Input

Type	Description
0	Stop
Negative values	Left
Positive values	Right

Tilt	Integer	Speed of tilt movement ranging from -100 to 100	Input

Type	Description
0	Stop
Negative values	Up
Positive values	Down

Zoom	Integer	Speed of zoom operation ranging from -100 to 100	Input

Type	Description
0	Stop
Negative values	Zoom Out
Positive values	Zoom In

Focus	Integer	Speed of focus operation ranging from -100 to 100	Input

Type	Description
0	Stop
Negative values	Zoom Near
Positive values	Zoom Far

Iris	Integer	Speed of Iris operation ranging from -100 to 100	Input

Type	Description
0	Stop
Negative values	Close Iris
Positive values	Open Iris

Method result:
PTZ Result

Example: Continuously move camera 0 (Matrix index) to LEFT and UP with speed of 20

```
int OperationResult;

OperationResult = PTZContinuous(0, -20, 20, 0, 0, 0);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.8 PTZAutoFocus

Activate / Deactivate camera auto-focus

Method parameters:

Parameter	Type	Description	Direction

Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform.	Input

Auto Focus operation	Value
PTZ_AUTOFOCUS_OFF	0
PTZ_AUTOFOCUS_ON	1

Method result:
PTZ Result

Example: Activate auto-focus of camera 0 (Matrix index)
```
int OperationResult;

OperationResult = PTZAutoFocus(0, PTZ_AUTOFOCUS_ON);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.9 PTZAutoIris

Activate / Deactivate camera auto-iris

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform.	Input

Auto Iris operation	Value
PTZ_AUTOIRIS_OFF	0
PTZ_AUTOIRIS_ON	1

Method result:
PTZ Result

Example: Activate auto-iris of camera 0 (Matrix index)
```
int OperationResult;

OperationResult = PTZAutoIris(0, PTZ_AUTOIRIS_ON);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.10 PTZMenuControl

Control the OSD Menu from analog PTZ cameras.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform.	Input

Menu Operation	Value
PTZ_MENU_OPEN	0
PTZ_MENU_CLOSE	1
PTZ_MENU_LEFT	2
PTZ_MENU_RIGHT	3
PTZ_MENU_UP	4
PTZ_MENU_DOWN	5
PTZ_MENU_ENTER	6
PTZ_MENU_CANCEL	7

Method result:
PTZ Result

Example: Open OSD menu from camera 0 (Matrix index)
```
int OperationResult;

OperationResult = PTZMenuControl(0, PTZ_MENU_OPEN);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.11 PTZCallPreset

Call a preset from a camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Preset	Integer	Preset number	Input
Speed	Integer	Speed of the operation (From 1 to 100)	Input

Method result:
PTZ Result

Example: Call preset 2 from camera 0 (Matrix index) with speed of 80
```
int OperationResult;

OperationResult = PTZCallPreset(0, 2, 80);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.12 PTZSetPreset

Set a preset in a camera using current position

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Preset	Integer	Preset number	Input

Description	String	Preset description	Input

Method result:
PTZ Result

Example: Store preset 3 on camera 0 (Matrix index) with name "Entrance"

```
int OperationResult;

OperationResult = PTZSetPreset(0, 3, "Entrance");

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.13 PTZCallPattern

Call a pattern from a camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Pattern	Integer	Pattern number	Input

Method result:
PTZ Result

Example: Call pattern 2 from camera 0 (Matrix index)

```
int OperationResult;

OperationResult = PTZCallPattern(0, 2);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.14 PTZWiper

Activate / Deactivate camera wiper

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform. Wiper operation | Value PTZ_WIPER_OFF | 0 PTZ_WIPER_ON | 1	Input

Method result:
PTZ Result

Example: Activate wiper of camera 0 (Matrix index)

```
int OperationResult;
```

```
OperationResult = PTZWiper(0, PTZ_WIPER_ON);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.15 PTZAuxiliary

Activate / Deactivate camera wiper

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform. **Auxiliary operation** | **Value** PTZ_AUXILIARY_OFF | 0 PTZ_AUXILIARY_ON | 1	Input
Value	Integer	Auxiliary number	Input

Method result:
PTZ Result

Example: Activate auxiliary 2 from camera 0 (Matrix index)

```
int OperationResult;

OperationResult = PTZAuxiliary(0, PTZ_AUXILIARY_ON, 2);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.16 PTZLock

Lock / Unlock the specified camera for exclusive use.

If this method is called on a locked camera, it will unlock, otherwise it will lock.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:
PTZ Result

Example: Lock camera 0 (Matrix index)

```
int OperationResult;

OperationResult = PTZLock(0);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.17 PTZPatrol

Operates the PTZ patrol of the specified camera.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Operation	Integer	Identification of the operation to perform.	Input
Patrol	Integer	Patrol number	Input

Patrol operation	Value
PTZ_PATROL_PAUSE	0
PTZ_PATROL_PLAY	1

Method result:
PTZ Result

Example: Start patrol 1 from camera 0 (Matrix index)

```
int OperationResult;

OperationResult = PTZPatrol(0, PTZ_PATROL_PLAY, 1);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.18 AutoPTZPatrolActive

Check if the Auto PTZ-Patrol from a camera is activated or deactivated

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Active	Boolean	Patrol is activated or deactivated	Output

Method result:

Boolean	Description
TRUE	Value was retrieved
FALSE	Error retrieving value

Example: Check if Auto PTZ-Patrol is activated for camera 0

```
bool Active;

if (true == AutoPTZPatrolActive(0, Active)) {

  if (true == Active) {
    ...
  }

}
```

3.8.2.19 GetPTZControlType

Retrieve the type of PTZ control for the specified camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
ControlType	Integer	Patrol is activated or deactivated	Output

Type of PTZ control	Value
PTZ_CONTROL_TYPE_NORMAL	0
PTZ_CONTROL_TYPE_DIGITAL	1

Method result:

Boolean	Description
TRUE	Value was retrieved
FALSE	Error retrieving value

Example: Check if PTZ control from camera 0 (Matrix index) is Normal

```
bool ControlType;

if (true == GetPTZControlType(0, ControlType)) {

  if (PTZ_CONTROL_TYPE_NORMAL == ControlType) {
    ...
  }

}
```

3.8.2.20 SetPTZControlType

Set the type of PTZ control for the specified camera.

The control allows for two types of PTZ control, Normal (Which will actually move the camera itself or 360 dewarp) and Digital (Used by Digital zoom).

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
ControlType	Integer	Patrol is activated or deactivated	Output

Type of PTZ control	Value
PTZ_CONTROL_TYPE_NORMAL	0
PTZ_CONTROL_TYPE_DIGITAL	1

Example: Set PTZ control type of camera 0 (Matrix index) to Digital

```
SetPTZControlType(0, PTZ_CONTROL_TYPE_DIGITAL);
```

3.8.2.21 GetPTZVisualJoystick

Check if visual joystick from a camera is activated

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:

Boolean	Description
TRUE	Visual joystick is activated
FALSE	Visual joystick is deactivated

Example: Check if visual joystick from camera 0 (Matrix index) is active

```
if (true == GetPTZVisualJoystick(0)) {
  ...
}
```

3.8.2.22 SetPTZVisualJoystick

Activate / Deactivate the visual joystick from a camera.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
Active	Boolean	Activate or Deactivate the visual joystick	Input

Method result:
PTZ Result

Example: Activate visual joystick of camera 0 (Matrix index)

```
int OperationResult;

OperationResult = SetPTZVisualJoystick(0, true);

if (PTZ_OK == OperationResult) {
  ...
}
```

3.8.2.23 GetPTZPresetCount

Query the number of presets of a camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:
Number of presets

Example: Retrieve the number of presets from camera 0 (Matrix index)

```
int PresetCount;

PresetCount = GetPTZPresetCount(0);
```

3.8.2.24 GetPTZPreset

Retrieve the data of a camera preset. Use this method along with GetPTZPresetCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
PresetIndex	Integer	Index of the preset (Starting with 0)	Input
Preset	TxPreset	Preset data	Output

TxPreset struct

Field	Type	Description
ID	Integer	Preset ID
Description	String	Preset description

Method result:

Boolean	Description
TRUE	Preset data was retrieved
FALSE	Preset data was not retrieved

Example: Retrieve the data of preset 0 from camera 0 (Matrix index)

```
TxPreset PresetData;

if (true == GetPTZPreset(0, 0, &PresetData) {
  ...
}
```

3.8.2.25 GetPTZPatrolSchemeCount

Query the number of PTZ Patrols of a camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:
Number of patrols

Example: Retrieve the number of patrols from camera 0 (Matrix index)

```
int PatrolCount;

PatrolCount = GetPTZPatrolSchemeCount(0);
```

3.8.2.26 GetPTZPatrolScheme

Retrieve the data of a camera patrol. Use this method along with GetPTZPatrolSchemeCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
PTZPatrolIndex	Integer	Index of the patrol (Starting with 0)	Input
PTZPatrol	TxPTZPatrolScheme	Patrol data	Output

TxPTZPatrolScheme struct

Field	Type	Description
ID	Integer	Patrol ID
Name	String	Patrol name

Description	String	Patrol description

Method result:

Boolean	Description
TRUE	Patrol data was retrieved
FALSE	Patrol data was not retrieved

Example: Retrieve the data of patrol 0 from camera 0 (Matrix index)

```
TxPTZPatrolScheme PatrolData;

if (true == GetPTZPatrolScheme(0, 0, &PatrolData) {
  ...
}
```

3.8.2.27 GetPTZAuxiliaryCount

Query the number of auxiliaries from a camera

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input

Method result:
Number of auxiliaries

Example: Retrieve the number of auxiliaries from camera 0 (Matrix index)

```
int AuxCount;

AuxCount = GetPTZAuxiliaryCount(0);
```

3.8.2.28 GetPTZAuxiliary

Retrieve the data of a camera auxiliary. Use this method along with GetPTZAuxiliaryCount.

Method parameters:

Parameter	Type	Description	Direction
Index	Integer	Index of the object on screen matrix	Input
AuxiliaryIndex	Integer	Index of the auxiliary (Starting with 0)	Input
Auxiliary	TxPTZAuxiliary	Auxiliary data	Output

TxPTZAuxiliary struct

Field	Type	Description
ID	Integer	ID of auxiliar command
Description	String	Description of auxiliar command

Method result:

Boolean	Description
TRUE	Auxiliary data was retrieved
FALSE	Auxiliary data was not retrieved

Example: Retrieve the data of auxiliar 0 from camera 0 (Matrix index)

```
TxPTZAuxiliary AuxData;

if (true == GetPTZAuxiliary(0, 0, &AuxData) {
  ...
}
```

3.8.3 Properties

This section describes properties of PTZ control.

3.8.3.1 PTZSimultaneous

This property activates and deactivates PTZ Simultaneous control.

PTZ Simultaneous is also known as "Click and Center" and is the ability to click on the image and send a command to the camera to center on the clicked position.

By activating this property, whenever user clicks on image, the control will send a command to the camera to center on the clicked coordinates

Value: Boolean

Example: Activate PTZ Simutaneous control

```
PTZSimultaneous = true;
```

3.8.4 Events

This section describes the events related to PTZ.

3.8.4.1 OnPTZUsage

This event will be fired whenever an user from the system controls a PTZ camera.

Event parameters:

Parameter	Type	Description	Direction
Server	String	Server name	Input
Camera	String	Camera name	Input
InUse	Boolean	TRUE - PTZ is in use FALSE - User stopped using PTZ	Input
InUseByID	Integer	ID of the user controlling the PTZ	Input
InUseBy	String	Name of the user controlling the PTZ	Input

3.8.4.2 OnPTZLockState

This event will be fired whenever a PTZ camera is locked or unlocked for exclusive use.

Event parameters:

Parameter	Type	Description	Direction
Server	String	Server name	Input
Camera	String	Camera name	Input
Locked	Boolean	TRUE - PTZ is locked FALSE - PTZ is unlocked	Input
LockedByID	Integer	ID of the user that locked the PTZ	Input
LockedByStr	String	Name of the user that locked the PTZ	Input

3.8.4.3 OnPTZLockError

This event will be fired if a PTZLock operation failed.

Event parameters:

Parameter	Type	Description	Direction
Server	String	Server name	Input
Camera	String	Camera name	Input
Code	Integer	Error code Constant / Value PTZ_LOCK_ERROR_LOCK 1 PTZ_LOCK_ERROR_UNLOCK 2 PTZ_LOCK_ERROR_NORIGHTS 3	Input
LockedBy	String	Name of the user that locked the PTZ	Input

3.8.4.4 OnPTZPatrolState

This event will be fired whenever the state of PTZ Patrol from a camera has changed.

Event parameters:

Parameter	Type	Description	Direction
Server	String	Server name	Input
Camera	String	Camera name	Input
SchemeName	String	Current scheme	Input
SchemeNumber	Integer	Number of current scheme	Input
Paused	Boolean	PTZ Patrol state. TRUE - Patrol is paused FALSE - Patrol is running	Input

3.9 Localization

This section defines the properties to localize the control.

3.9.1 Properties

This section defines the properties to localize the control.

3.9.1.1 LanguageID

Use the property to change the language of the control.

Value: String

The language IDs are defined by Windows Language Code Identifier Reference (http://msdn.microsoft.com/en-us/library/ms533052%28v=vs.85%29.aspx).

The supported languages are:

ID	Descrição
PT-BR	Brazilian portuguese
EN-US	English
ES	Spanish
FR	French
TR	Turkish

KO	Korean
ZH-CN	Simplified chinese
ZH-TW	Traditional chinese
IT	Italian
RU	Russian
PL	Polish
NL-NL	Dutch
CS	Czech
LT	Lithuanian
JA	Japanese
HU	Hungarian
TH	Thai

Exemplo1: Change the language to english:

```
LanguageID = 'EN-US';
```

Exemplo2: Change the language to spanish:

```
LanguageID = 'ES';
```

Part IV

4 Usage example

Here we will provide a few usage examples of the recommended steps for showing objects on the objects matrix. Be advised that this is just examples rather than working code samples.

```
// First step should be connecting to a server

// In order to connect to a server we must provide the types of objects
// to download. Here we will download all object types.
int ObjectTypes = OBJECTTYPE_SCREENSTYLE | OBJECTTYPE_USER_SCREENVIEW |
  OBJECTTYPE_PUBLIC_SCREENVIEW | OBJECTTYPE_CAMERA | OBJECTTYPE_MAP;

// Here we are connecting to server 192.168.10.12 and naming it "Server"
// for future references
// We configure the connection to asynchronous mode, so connection result
// will be informed on an event
Connect("NA:Server,AD:192.168.10.12,PO:8600,US:YWRtaW4=,
PW:S7sdHsd=,CM:2", ObjectTypes, TRUE, FALSE);
```

```
// This is an example of event handler to receive connection status
OnConnectionComplete {

  // Connection to servers is now completed, now we add a few cameras
  // to the screen

  // First we set the matrix to 2x2 layout
  MatrixScreenStyle = 6278;

  // Now we add 4 cameras
  MatrixAddObject(OBJECTTYPE_CAMERA, "Server", "Camera1", -1);
  MatrixAddObject(OBJECTTYPE_CAMERA, "Server", "Camera2", -1);
  MatrixAddObject(OBJECTTYPE_CAMERA, "Server", "Camera3", -1);
  MatrixAddObject(OBJECTTYPE_CAMERA, "Server", "Camera4", -1);

}
```