GlobeSpotter

API specification

GlobeSpotter API version 2.1

© 2011 ***CycloMedia Technology B.V.***

Table of Contents

© 2011 *CycloMedia Technology B.V.*

© 2011 *CycloMedia Technology B.V.*

© 2011 *CycloMedia Technology B.V.*

© 2011 *CycloMedia Technology B.V.*

Version history

Version	Status	Date	Details of change	Author(s)
0.1	Draft	20100908	Initial version	SBr
0.5	Draft	20100923	Reworked. Added core API functionality	SBr
0.9.0	Draft	20101007	Applied corrections & improvements. Added to core API. Added Multi-Window API	RBu
0.9.1	Pre-Release	20101013	Fixed some omissions.	RBu
0.9.2	Release	20101013	Added permission description	SBr
1.0	Release	20101015	Added flash player requirements	SBr
1.1	Release	20110328	Added MDI, Full API, measurement functions and events	SBr
1.1.1	Release	20110401	Added permissions table	SBr

GlobeSpotter API version 2.1 .
Author SBr
Document version 1.1

Copyright 2011, CycloMedia Technology B.V.
All rights reserved

1 Introduction and Overview

This document describes a global overview of the GlobeSpotter API. The GlobeSpotter API can be used by integrators to control one or multiple cyclorama windows embedded in a web environment or in a desktop application.

1.1 API packages

The GlobeSpotter API is split into three API packages:

- Single Window API
- Multi Window API
- Multi Window API + map

The Single Window API may be used when the integrator only requires a single cyclorama-window. A cyclorama-window (also called a Viewer) is the component used to view a cyclorama and any applicable overlays.

The Multi Window API may be used when the integrator requires multiple windows. Please note that due to the uncertainty introduced by measuring in a single window only, this level of the API is required to do proper measurements. The Single Window API should *not* be used for measurements.

The following platforms are supported, for all three API packages:

- Web HTML
- Web HTML with JavaScript
- Microsoft Windows .Net

All API packages use Adobe's Flash player. In order to use the GlobeSpotter API , Adobe Flash Player version 10.0.22.18 or higher is required. The player can be installed from the Adobe website. On the website you can find the installation instructions.
During integration development it's highly recommended to use the debug player to identify problems caused by the player, web services, core API or integration software.

For all web integrations, it's the user's choice which browser will be used and also which flash player will be used (Internet explorer, Mozilla/Netscape).
The Microsoft Windows .Net integration always uses the Internet explorer flash player as an ActiveX plug-in.

© 2011 *CycloMedia Technology B.V.*

1.2 Revision structure

The GlobeSpotter API is available on the following location:

- v2 (current revision)

Single Window API:
https://www.globespotter.nl/v2/api/bapi/viewer_bapi.swf

Multi-Window API:
https://www.globespotter.nl/v2/api/mapi/viewer_mapi.swf

Multi-Window API + map:
https://www.globespotter.nl/v2/api/mapi/viewer_fapi.swf

It's highly recommended to download the swf each time the integration is started. The advantage of downloading the swf is always having the latest (minor) revision of the GlobeSpotter API. New major revisions of the GlobeSpotter API will have a different url (`/v2_1/api/..`).

1.3 Permission on functionalities

The GlobeSpotter API has been divided in several functionality groups. An end-user can have permission on one or multiple functionality groups. The API gives full access to integrate these functionalities but as soon as a user wants to use this functionality, there should be sufficient permissions. The API function `getPermissions` returns the runtime available permissions on the available functionality groups. The following API related permissions can be returned:

Permission name	Description	GlobeSpotter Pro license required by end user
Basic	Basic functionalities like: viewing panoramic images see Image recordings, print, save, etc.	No
AddLayerWMS	Add WMS layers	Yes
AddLayerWFS	Add WFS layers	Yes
MeasureHeight	Perform height measurement with height entity	Yes
MeasureVolume	Perform Volume measurement with volume entity	Yes
MeasureArea	Perform Area measurement with area entity	Yes
MeasureAdvanced	Perform point and line measurements with advanced measurement tools	Yes

© 2011 *CycloMedia Technology B.V.*

2 Web HTML

The Web HTML package can be used to facilitate easy integration into static or dynamically generated web pages. The following example illustrates this:

```
<html>
    <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
            id="viewer_api" width="800" height=400"
        <param name="movie" value="viewer_api" />
        <param name="allowScriptAccess" value="always" />
        <param name="allowFullScreen" value="true" />
        <embed src="https://www.globespotter.nl/v2/api/bapi/viewer_bapi.swf"
             quality="high" bgcolor="#888888"
             width="800" height="400" name="viewer_api" align="middle"
             play="true"
             loop="false"
             quality="high"
             allowScriptAccess="always"
             type="application/x-shockwave-flash"
             pluginspage="http://www.adobe.com/go/getflashplayer"
             allowFullScreen="true"
             flashvars="APIKey= abcdefghijklmnopqrstuvwxyz1234567890
                        &MapSRSName=EPSG:28992&posx=145394&posy=427084">
        </embed>
    </object>
</html>
```

This example doesn't check if there's a flash player available and if its version matches the supported version. It also doesn't set any credentials, which means that GlobeSpotter itself will ask for a username/password combination.

The following **case-sensitive** 'flashvars' must be applied:

- `APIKey`: Supplied API key
- `MapSRSName`: The required coordinate system using the EPSG format

All of the supported coordinate systems are listed here:
https://atlas.cyclomedia.com/Recordings/wfs?service=WFS&Request=GetCapabilities&Version=1.1.0

The following **case-sensitive** 'flashvars' can be used to open an image:

- `imageid`: To open an image based on ID
- `address`: To open an image based on address
- `posx, posy`: To open an image based on XY location
- `yaw`: Horizontal viewing direction in degrees (0 – 360)
- `pitch`: Vertical viewing direction in degrees (-90 – +90)
- `hFov`: Horizontal field of view in degrees (30 – 130)
- TID: temporary identification key

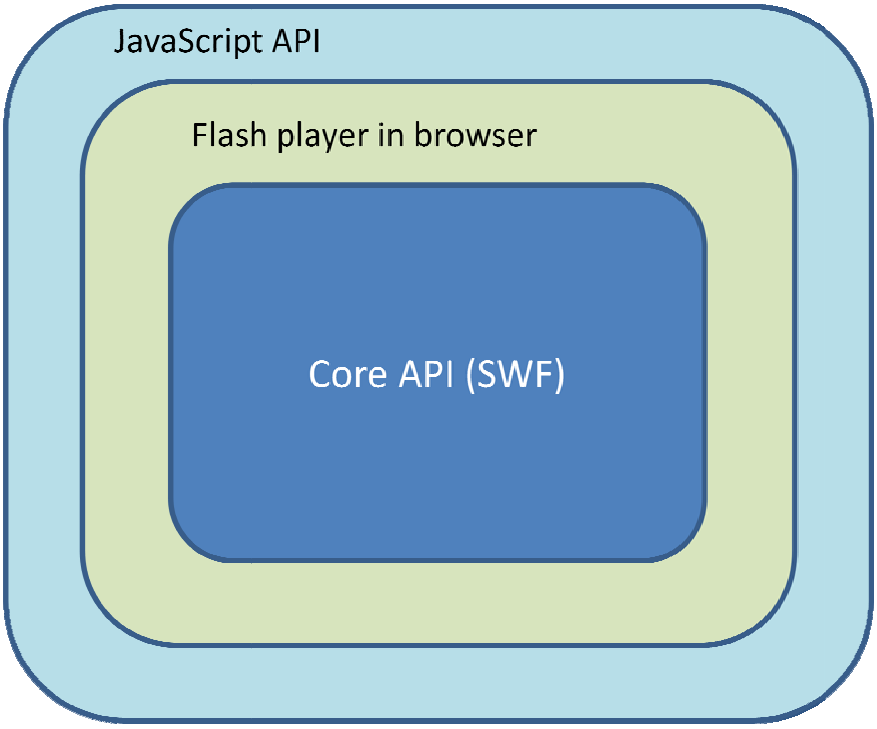
Examples:
`&imageid=5B043KT4&posx=145394&posy=427084`
`&address=3016AB 8&yaw=40`

© 2011 *CycloMedia Technology B.V.*

3 Web HTML with JavaScript

The JavaScript API can be used when more runtime interaction with the core API is needed in a web based application. When using the JavaScript API, the full functionality of the Core API can be used when the SWF is embedded in a web page.

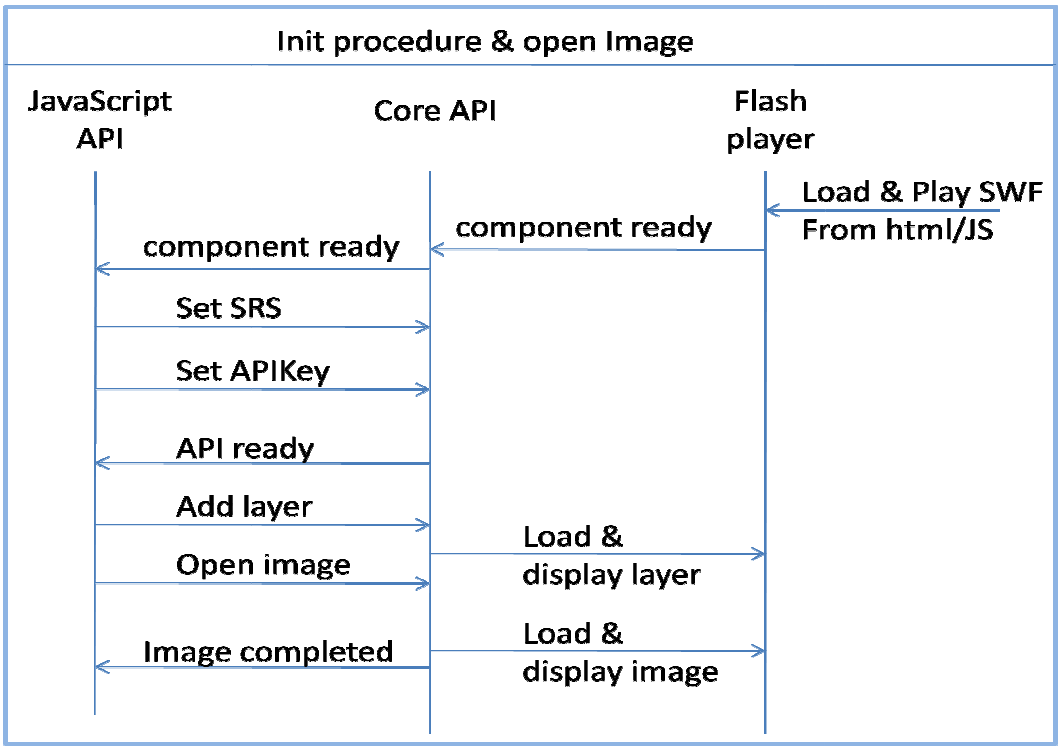
The following image shows the building blocks of the JavaScript API.



The Core API is packed in a Flash SWF component. This SWF component can be embedded in a webpage as explained in chapter 3 or it can be loaded using Adobe's JavaScript functions to load and play a SWF component. Loading of the SWF is performed by the Flash player which runs in a web browser context.

© 2011 *CycloMedia Technology B.V.*

The next diagram shows how the API can be initialized and how an image can be opened:



After the SWF has been loaded by the flash player, the Core API receives a Component ready event from the flash player and performs an internal initialisation procedure. When this is finished it will try to call `hst_componentReady` that should be available on the JavaScript side.

Next step is to set the spatial reference system (SRS) using an EPSG code (EPSG:28992 for example, may be used in the Netherlands) and API-key. With this information the Core API can finish its initialisation and calls the `hst_apiReady` function on the JavaScript side.

The SRS and the API-key are the only values that are absolutely necessary for the integrator to set in order to proceed to the *onAPIReady*-event. Other such parameters (recording location service, address service, tile services) have default values. Note that in this version of the API, changing these parameters is not supported after the *onAPIReady*-event has been given, and as a result, any such parameters will need to be set in the window between the *onComponentReady*-event and the *onAPIReady*-event. The same time-window can be used to set the credentials of the user in order to have single point authorisation.

From this point on, the API can be fully used and it's possible to add one or multiple layers using the *AddLayer*-functionality. For more information see the examples in chapter 6.

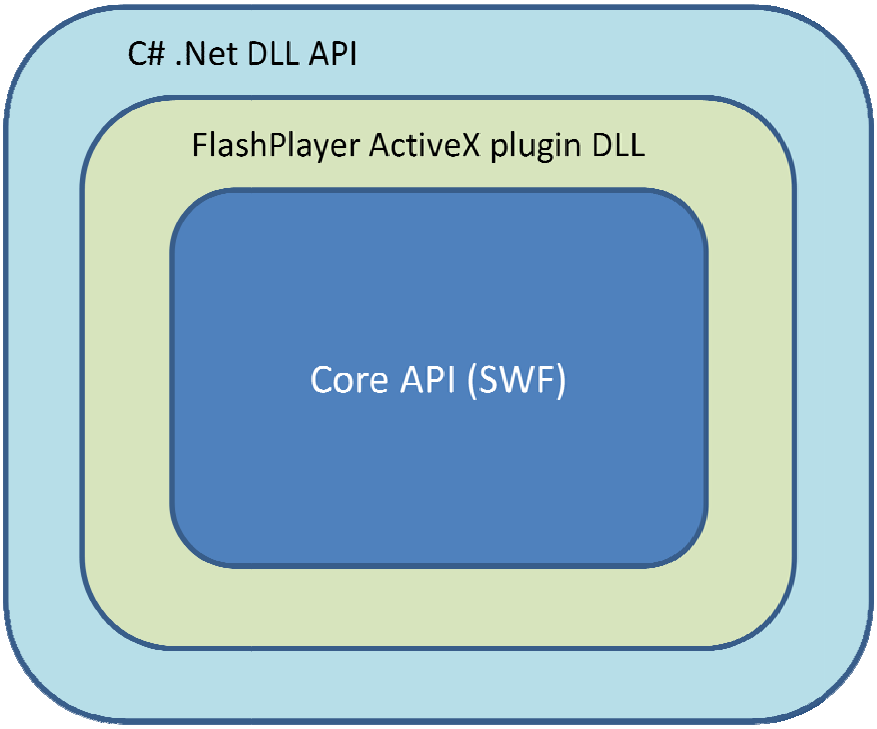
© 2011 *CycloMedia Technology B.V.*

4 Microsoft Windows .Net

The Microsoft Windows .Net API may be used when integration in a desktop application on the Windows platform is required. When using the Microsoft Windows .Net API, the full functionality of the Core API can be used.

4.1 Overview

The following image shows the building blocks of the Microsoft Windows .Net API.



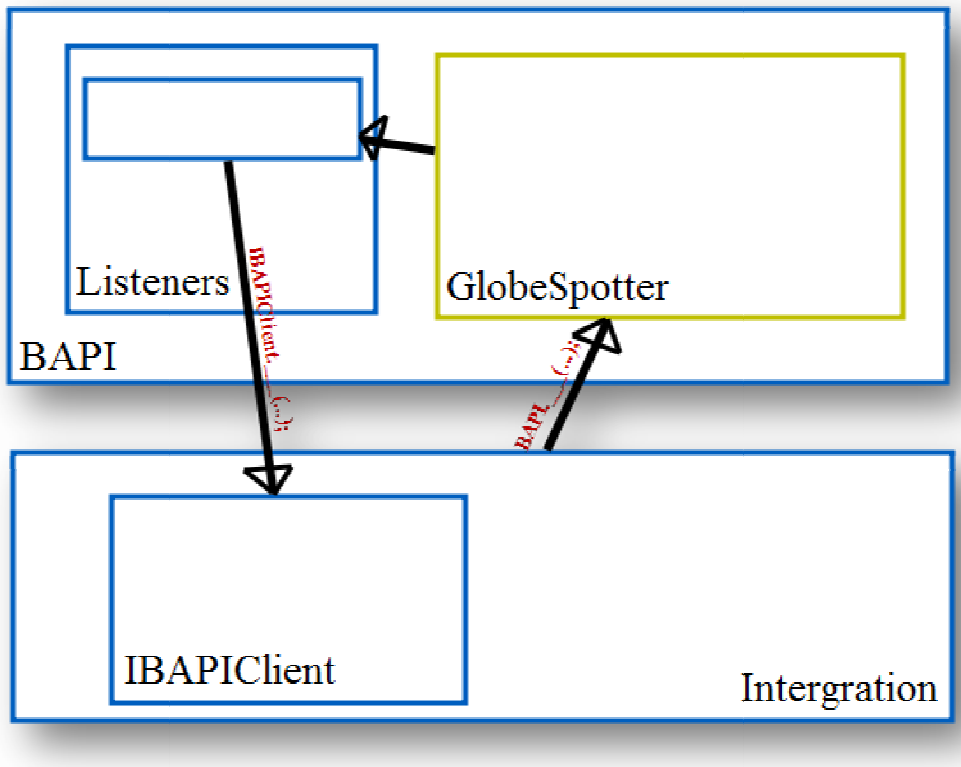
The Core API is packed in a Flash SWF component. This SWF is loaded at runtime from a remote location. The flash player is integrated in the Microsoft Windows .Net API as an ActiveX plug-in DLL. This makes it possible to use SWF components outside a web browser context. The Microsoft Windows .Net API exposes the Core API functions through a well-defined API and makes integration of this SWF component an easy task.

The ***GlobeSpotter Basic API Tutorial for Microsoft .Net*** (see chapter 8) illustrates how integration with a desktop application may be accomplished.

© 2011 *CycloMedia Technology B.V.*

4.2 DLL communication

The following image explains how the internal components of the Microsoft .Net API communicate with each other. To understand the explanation it's recommended to read the Microsoft .Net GlobeSpotter Basic API tutorial first.



Once a *BAPI*-object is created, the integrator can add a listener (either an *IBAPIClient* or an *AbstractBAPIClient*) with the *Initialize*-method contained in the *BAPI*-class. The *Initialize*-method also initializes the process of starting-up GlobeSpotter, after wich the integrator may expect the *OnSWFReady*-method to be called by the *BAPI*.

This process is examplary for the rest of the communication between the intergation and the GlobeSpotter-API. If the integration must do something with GlobeSpotter, it must call the methods in a *BAPI*-object. When GlobeSpotter communicates with the integration, it will do so via calls to the listener(s) the integrator has added as parameters of the *Initialize*-method (each of which may be either an *IBAPIClient* or an *AbstractBAPIClient*).

© 2011 *CycloMedia Technology B.V.*

5 API compatibility

The following table shows all functionality with corresponding API version and an additional 'deprecated since' version to indicate in which version this particular functionality got deprecated.

Name	Introduced in API version	Deprecated since
addAreaEntity	2.0.1024	
addGMLLayer	2.0.1024	
addHeightEntity	2.0.1024	
addLineMeasurement	2.0.1502	
addMapGMLLayer	2.0.1479	
addMapOSMLayer	2.0.1479	
addMapWFSLayer	2.0.1479	
addMapWMSLayer	2.0.1479	
addPointMeasurement	2.0.1502	
addVolumeEntity	2.0.1024	
addWFSLayer	2.0.1024	
cancelMeasurement	2.0.1502	
clearDrawing	2.0.1024	
clearDrawings	2.0.1024	
closeMap	2.0.1479	
drawMarkerAtHV	2.0.1024	
drawMarkerAtXY	2.0.1024	
drawMarkerInDirection	2.0.1024	
finishMeasurement	2.0.1502	
getAbsMaxViewers	2.0.1024	
getAPIReadyState	2.0.1024	
getApplicationName	2.0.1024	
getAutoTileViewers	2.0.1024	2.0.1479
getEntityData	2.0.1024	
getEntityDescription	2.0.1024	
getEntityName	2.0.1024	
getFocusEntity	2.0.1024	
getGamma	2.0.1024	
getHFov	2.0.1024	
getImageID	2.0.1024	
getLanguageLocale	2.0.1024	
getMajorVersion	2.0.1024	
getMapCenter	2.0.1479	
getMapClickMode	2.0.1479	
getMapExtent	2.0.1479	
getMapScreenshot	2.0.1479	
getMapState	2.0.1479	
getMapZoom	2.0.1479	
getMaxViewers	2.0.1024	
getMDICanvasDimensions	2.0.1479	
getMDIWindowingMode	2.0.1479	
getMinorVersion	2.0.1024	
getPermissions	2.0.1024	
getPitch	2.0.1024	
getRecordingLocation	2.0.1024	
getRecycleFirstViewer	2.0.1024	
getSupportedLanguageLocales	2.0.1024	
getViewerClickMode	2.0.1024	
getViewerCount	2.0.1024	
getViewerScreenshot	2.0.1479	
getViewerState	2.0.1479	
getViewerWindowColor	2.0.1479	
getViewshotData	2.0.1024	2.0.1479
getYaw	2.0.1024	
hideDrawingLayer	2.0.1024	
hideEntityLayer	2.0.1024	
hideMapEntityLayer	2.0.1479	
hideMapRLSLayer	2.0.1479	
hideRecordingLocations	2.0.1024	
hst_apiReady	2.0.1024	
hst_componentReady	2.0.1024	
hst_entityDataChanged	2.0.1024	
hst_entityFocusChanged	2.0.1024	
hst_imageChanged	2.0.1024	
hst_imageCompleted	2.0.1024	
hst_imageFailed	2.0.1024	
hst_imagePreviewCompleted	2.0.1024	
hst_imageSegmentLoaded	2.0.1024	
hst_mapAdded	2.0.1479	
hst_mapClicked	2.0.1479	

© 2011 *CycloMedia Technology B.V.*

hst_mapExtentChanged	2.0.1479	
hst_mapRemoved	2.0.1479	
hst_mapWindowMaximized	2.0.1479	
hst_mapWindowMinimized	2.0.1479	
hst_mapWindowMoved	2.0.1479	
hst_mapWindowResized	2.0.1479	
hst_mapWindowRestored	2.0.1479	
hst_mapWindowSelected	2.0.1479	
hst_markerClicked	2.0.1024	
hst_measurementCanceled	2.0.1479	
hst_measurementCreated	2.0.1479	
hst_measurementFinished	2.0.1479	
hst_measurementUpdated	2.0.1479	
hst_observationAdded	2.0.1479	
hst_observationRemoved	2.0.1479	
hst_observationUpdated	2.0.1479	
hst_openImageError	2.0.1024	
hst_openImageResultEmpty	2.0.1024	
hst_viewChanged	2.0.1024	
hst_viewerAdded	2.0.1024	
hst_viewerClicked	2.0.1024	
hst_viewerRemoved	2.0.1024	
hst_viewerWindowMaximized	2.0.1479	
hst_viewerWindowMinimized	2.0.1479	
hst_viewerWindowMoved	2.0.1479	
hst_viewerWindowResized	2.0.1479	
hst_viewerWindowRestored	2.0.1479	
hst_viewerWindowSelected	2.0.1479	
hst_viewLoaded	2.0.1024	
lookAtCoordinate	2.0.1024	
maximizeMapWindow	2.0.1479	
maximizeViewerWindow	2.0.1479	
minimizeMapWindow	2.0.1479	
minimizeViewerWindow	2.0.1479	
moveMapWindow	2.0.1479	
moveViewerWindow	2.0.1479	
openImage	2.0.1024	
openMap	2.0.1479	
removeAllEntities	2.0.1024	
removeEntity	2.0.1024	
removeGMLLayer	2.0.1024	
removeMapLayer	2.0.1024	
removeViewer	2.0.1024	
removeWFSLayer	2.0.1024	

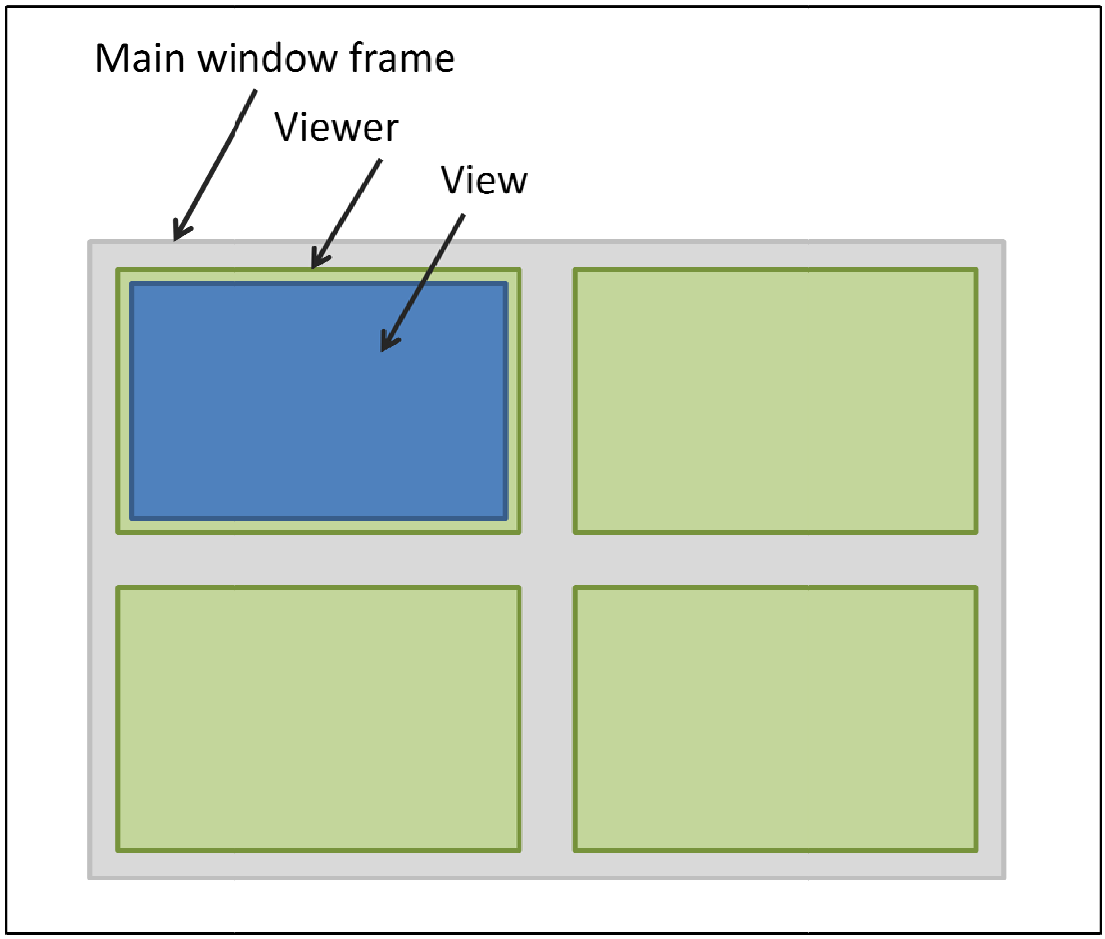
resizeMapWindow	2.0.1479	
resizeViewerWindow	2.0.1479	
restoreMapWindow	2.0.1479	
restoreViewerWindow	2.0.1479	
rotateDown	2.0.1024	
rotateLeft	2.0.1024	
rotateRight	2.0.1024	
rotateUp	2.0.1024	
selectMapWindow	2.0.1479	
selectViewerWindow	2.0.1479	
setApplicationParameter	2.0.1024	
setAutoTileViewers	2.0.1024	2.0.1479
setDrawingColor	2.0.1024	
setDrawingImageURL	2.0.1024	
setDrawingMode	2.0.1024	
setDrawingSize	2.0.1024	
setEntityDescription	2.0.1024	
setFocusEntity	2.0.1024	
setGamma	2.0.1024	
setHFov	2.0.1024	
setLanguageLocale	2.0.1024	
setMapCenter	2.0.1479	
setMapClickMode	2.0.1479	
setMapExtent	2.0.1479	
setMapRLSLayerDateRange	2.0.1479	
setMapZoom	2.0.1479	
setMaxViewers	2.0.1024	
setMDIWindowingMode	2.0.1479	
setPitch	2.0.1024	
setRecycleFirstViewer	2.0.1479	
setTID	2.0.1479	
setUserNamePassword	2.0.1024	
setViewerClickMode	2.0.1024	
setYaw	2.0.1024	
showDrawingLayer	2.0.1024	
showEntityLayer	2.0.1024	
showMapEntityLayer	2.0.1479	
showMapPrintDialog	2.0.1479	
showMapRLSLayer	2.0.1479	
showMapSaveDialog	2.0.1479	
showRecordingLocations	2.0.1024	
showViewerPrintDialog	2.0.1479	
showViewerSaveDialog	2.0.1479	
takeViewshot	2.0.1024	2.0.1479

© 2011 *CycloMedia Technology B.V.*

6 Core API

This chapter describes the functionality provided by the core API. The core API functionality can be used from a AS3 or JavaScript environment. When the API is used in a Microsoft .Net environment, the Microsoft .Net API must be used.

To understand how the Core API deals with the main window frame, viewers and views, a diagram is listed below.



The Single Window API (BAPI) consists of a main window frame with one viewer and one view.

The Multi Window API consists of a main window frame with multiple viewers. Each viewer has its own view. The 'view' is used, for example, in *(X,Y)* click-mode where *(left,top)* has coordinates *(0,0).*

The Multi Window API + map is similar to the Multi Window API but with an additional map window.

© 2011 ***CycloMedia Technology B.V.***

6.1 Host side call-backs

This section describes the functions that will be called from the host (core API). The integrator should make them available on the integration side. (See also the JavaScript examples in chapter 7.)

An important detail is that, within the API, the call-backs are wrapped inside a try-catch statement, and any exceptions occurring within the dispatch are handled internally. This means that if the language the integrator allows silent fails (as JavaScript does, for example) instead of blocking on an unexpected fail, there is no way to assess what went wrong if something fails, *unless* the integrator wraps any potentially unsafe code occurring within the call-back with a try-catch block of its own.

6.1.1 General API functions

6.1.1.1 hst_componentReady

Function name:	hst_componentReady	
Parameter	**Type**	**Description**

Description:
Indicates to the host that the component is ready for use. After this event the user is able to set various required application parameters to be able to use the components API functionality.

6.1.1.2 hst_apiReady

Function name:	hst_apiReady	
Parameter	**Type**	**Description**
readyState	Boolean	The ready status of the API

Description:
Indicates to the host that the API is ready for use. Full API functionality should be available based on the user's credentials.

6.1.2 Open image events

6.1.2.1 hst_openImageResultEmpty

Function name:	hst_openImageResultEmpty	
Parameter	**Type**	**Description**
input	string	'value' parameter from previously called openImage
type	uint	OpenImage type (See openImage for more info)
viewerParameters	object	Parameters of viewer (see openImage for more info)
viewerId	uint	ID of viewer

Description:

Indicates to the host that the API was unable to open an image based on the input values of the openImage function most likely caused by one of the following reasons:

- ImageId not found
- Address not found
- No image found near provided address
- No image found near provided X,Y location

The parameters of the function reflect those entered in the openImage function.

6.1.2.2 hst_openImageError

Function name:	hst_openImageError	
Parameter	**Type**	**Description**
input	string	'value' parameter from previously called openImage
type	uint	OpenImage type (See openImage for more info)
viewerParameters	object	Parameters of viewer (see openImage for more info)
viewerId	uint	ID of viewer

Description:

Indicates to the host that the API was unable to open an image due to an error generated during the opening process. The parameters of the function reflect those entered in the `openImage` function most likely caused by one of the following reasons:

- Web service timeout
- Invalid request
- Insufficient rights to use the services

© 2011 *CycloMedia Technology B.V.*

6.1.3 Viewer events

6.1.3.1 hst_imageChanged

Function name:	hst_imageChanged	
Parameter	Type	Description
viewerID	uint	The id of the viewer of which its image changed

Description:

Indicates to the host that the image was changed within a certain viewer. This event occurs after the `openImage` function has been called, after clicking on a recording location, etc.

6.1.3.2 hst_imagePreviewCompleted

Function name:	hst_imagePreviewCompleted	
Parameter	Type	Description
viewerID	uint	The id of the viewer of which its image changed

Description:

Indicates to the host that the preview of an image completed. The preview is a low resolution image that is used to quickly show a result and give the user the experience of progress.

6.1.3.3 hst_imageSegmentLoaded

Function name:	hst_imageSegmentLoaded	
Parameter	Type	Description
viewerID	uint	The id of the viewer of which its image changed

Description:

Indicates to the host that a segment of the high resolution image has been loaded.

© 2011 *CycloMedia Technology B.V.*

6.1.3.4 hst_imageCompleted

Function name:	hst_imageCompleted	
Parameter	Type	Description
viewerID	uint	The id of the viewer of which its image changed

Description:

Indicates to the host that the entire high resolution image has been loaded.

6.1.3.5 hst_imageFailed

Function name:	hst_imageFailed	
Parameter	Type	Description
viewerID	uint	The id of the viewer of which its image changed

Description:

Indicates to the host that any part of the image could not be completely loaded. This means that the event is dispatched when either the preview or any segment fails to load.

6.1.3.6 hst_viewLoaded

Function name:	hst_viewLoaded	
Parameter	Type	Description
viewerID	uint	The id of the viewer of which its image changed

Description:

Indicates to the host that all image parts of the current view have been completely loaded.

6.1.3.7 hst_viewChanged

Function name:	hst_viewChanged	
Parameter	**Type**	**Description**
viewerID	uint	The id of the viewer of which its image changed
Yaw	float	horizontal orientation of a view in degrees
Pitch	float	vertical orientation of a view in degrees
HFov	float	horizontal field of view in degrees

Description:

Indicates to the host that the current view is changed due to a change in zoom level and / or orientation.

6.1.3.8 hst_viewerClicked

Function name:	hst_viewerClicked	
Parameter	**Type**	**Description**
viewerID	uint	The id of the viewer of which its image changed
c0	float	Click parameter 0
c1	float	Click parameter 1
c2	float	Click parameter 2 (optional)

Description:

Indicates to the host that the user clicked on a viewer. The parameters of the function depend on the viewer click mode. See *setViewerClickMode* for more information

6.1.4 Layer events

6.1.4.1 hst_markerClicked

Function name:	hst_markerClicked	
Parameter	**Type**	**Description**
viewerID	uint	The id of the viewer of which its image changed
drawingID	uint	The id of the marker that was clicked
x	float	The x-element of the click direction vector
y	float	The y-element of the click direction vector
z	float	The z-element of the click direction vector

Description:

Indicates to the host that a marker from the drawing layer was clicked. Returns the position of the clicked marker similar to the "vector" viewer click mode.

6.1.4.2 hst_entityDataChanged

Function name:	hst_entityDataChanged	
Parameter	**Type**	**Description**
entityID	uint	The id of the entity of which its data is updated.
entityData	[object]	The data of the entity that updated of which its contents depend on its type.

Description:

Indicates to the host that the data of an entity is updated.

6.1.4.3 hst_entityFocusChanged

Function name:	hst_entityFocusChanged	
Parameter	**Type**	**Description**
entityID	int	The id of the entity or -1 if the focus changed to null.
entityData	[object]	The data of the entity that updated of which its contents depend on its type.

Description:

Indicates to the host that the focus of an entity has changed.

6.2 Callable methods core API

6.2.1 General

6.2.1.1 setApplicationParameter

Function name:	setApplicationParameter	
Parameter	**Type**	**Description**
value	string	The value of the parameter
type	uint	The parameter type specifies what the value parameter means

Description:

Set the value of an application parameter. The type dictates which parameter is set:

`TYPE_URL_RECORDING_LOCATION_SERVICE = 1`
This type is used to set the recording location service URL. This services is used to obtain positioning meta data of an image and allows for various spatial data requests.

`TYPE_URL_ADDRESS_SERVICE = 2`
This type is used to set the address service URL. This service allows for various spatial requests based on address data. <b>This type is described for reference only. The value of this type cannot be set and the default service only supports address' within the Netherlands.
`TYPE_URL_TILE_SERVICE = 3`
This type is used to set the tile service URL. This service is used to obtain images.

`TYPE_SRS_NAME_MAP = 4`
This type is used to set the srs name of the map. The srs name is required in order to correctly display many visual features.

`TYPE_API_KEY = 5`
This type is used to set a key to give an integrator access to API functionality.

The SRS and the API-key are required in order to complete initialization. Setting any key after the `hst_onAPIReady` call-back is not supported.

6.2.1.2 getApplicationName

Function name:	getApplicationName	
Input parameter	Type	Description
Result	Type	Description
name	string	The name of the application

Description:

This function returns the name of the current application. It can be used for debugging purposes.

6.2.1.3 getMajorVersion

Function name:	getMajorVersion	
Input parameter	Type	Description
Result	Type	Description
version	string	The major release version

Description:

Returns the major release version of the current application. It indicates global changes within the application and can be used for debugging purposes. Global changes may indicate a change in the interface of the application.

6.2.1.4 getMinorVersion

Function name:	getMinorVersion	
Input parameter	Type	Description
Result	Type	Description
version	string	The minor release version

Description:

Returns the minor release version of the current application. It indicates minor changes within the application and can be used for debugging purposes (specify bugs).

6.2.1.5 getAPIReadyState

Function name:	getAPIReadyState	
Input parameter	Type	Description
Result	Type	Description
readyState	boolean	The ready state of the API

Description:

Returns whether or not the API can be used for general use.

6.2.1.6 getViewerClickMode

Function name:	getViewerClickMode	
Input parameter	Type	Description
Result	Type	Description
clickMode	uint	The click mode

Description:

Gets the current viewer click mode.

6.2.1.7 setViewerClickMode

Function name:	setViewerClickMode	
Input parameter	Type	Description
clickMode	uint	The click mode
Result	Type	Description

Description:

Sets the viewer click mode. The click mode defines the behaviour of the viewer clicked event.The following modes are defined:

`CLICK_MODE_X_Y = 1`
The image coordinates within the current view. The center coordinate of the top left pixel is defined to be (0.5, 0.5). The viewer clicked event returns x and y values.

`CLICK_MODE_H_V = 2`
Horizontal and vertical spherical coordinates within the entire image (cyclorama). The center of the image is defined to be (0.0, 0.0). The viewer clicked event returns H and V values.

`CLICK_MODE_VECTOR = 3`
Unit vector x, y and z coordinates that define the location within the image. The center of the image is defined to be (0.0, 1.0, 0.0). The viewer clicked event returns x, y and z values.

6.2.1.8 getPermissions

Function name:	getPermissions	
Input parameter	Type	Description
Result	Type	Description
	object	Permissions of the user based on the users credentials.

Description:

Retrieves the permissions of the user based on the users credentials.

6.2.1.9 setUserNamePassword

Function name:	setUserNamePassword	
Input parameter	Type	Description
userName	String	The name or ID of the user.
password	String	The password that belongs to the user-name or ID.
Result	Type	Description

Description:

Sets the credentials of the user. When successful, the user will not be asked to provide his or her credentials before (almost) each action.

Please note that this method may only be called in scope of *hst_onComponentReady* (before *hst_onAPIReady*)

6.2.1.10 setTID

Function name:	setTID	
Input parameter	Type	Description
TID	String	Temporary identification string
Result	Type	Description

Description:

Sets the temporary identification string that will be used for authentication of the requested images and recordinglocations.

Please note that this method may only be called in scope of *hst_onComponentReady* (before *hst_onAPIReady*)

6.2.2 Viewer properties

6.2.2.1 getImageID

Function name:	getImageID	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to obtain data from
Result	Type	Description
imageID	string	The id of the image
Description:		
Gets the image id of a viewer.		

6.2.2.2 setYaw

Function name:	setYaw	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply the yaw to
yaw	float	The horizontal angle of the view (degrees)
Result	Type	Description
Description:		
Sets the horizontal orientation of a view in degrees		

6.2.2.3 getYaw

Function name:	getYaw	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to obtain the data from
Result	Type	Description
yaw	float	The horizontal angle of the view (degrees)
Description:		
Gets the horizontal orientation of a view in degrees		

© 2011 *CycloMedia Technology B.V.*

6.2.2.4 getPitch

Function name:	getPitch	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to obtain the data from
Result	Type	Description
pitch	float	The vertical angle of the view (degrees)

Description:
Gets the vertical orientation of a view in degrees

6.2.2.5 setPitch

Function name:	setPitch	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply the pitch to
pitch	float	The vertical angle of the view
Result	Type	Description

Description:
Sets the vertical orientation of a view in degrees

6.2.2.6 getHFov

Function name:	getHFov	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to obtain the data from
Result	Type	Description
hFov	float	The horizontal field of view in degrees

Description:
Gets the horizontal field of view in degrees

6.2.2.7 setHFov

Function name:	setHFov	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply the hFov to
hFov	float	The horizontal field of view in degrees
Result	Type	Description

Description:

Sets the horizontal field of view in degrees

6.2.2.8 getGamma

Function name:	getGamma	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to obtain the data from
Result	Type	Description
gamma	float	A gamma value within the range of [0.1..3.0]

Description:

Gets the gamma of a viewer

6.2.2.9 setGamma

Function name:	setGamma	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply the gamma to
gamma	float	A gamma value within the range of [0.1..3.0]
Result	Type	Description

Description:

Sets the gamma of a viewer

6.2.2.10 rotateLeft

Function name:	rotateLeft	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply rotation on
value	float	A rotation value in degrees
Result	Type	Description

Description:

Rotate a viewer to the left by a certain angle in degrees

6.2.2.11 rotateRight

Function name:	rotateRight	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply rotation on
value	float	A rotation value in degrees
Result	Type	Description

Description:

Rotate a viewer to the right by a certain angle in degrees

6.2.2.12 rotateUp

Function name:	rotateUp	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply rotation on
value	float	A rotation value in degrees
Result	Type	Description

Description:

Rotate a viewer upwards by a certain angle in degrees

6.2.2.13 rotateDown

Function name:	rotateDown	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to apply rotation on
value	float	A rotation value in degrees
Result	Type	Description

Description:

Rotate a viewer downwards by a certain angle in degrees
Gets the recording location meta data of an opened image by viewer id. The recording location isreturned as an object, having the following properties:

6.2.2.14 getRecordingLocation

Function name:	getRecordingLocation	
Input parameter	Type	Description
viewerID	uint	The viewer id of which to obtain the data from
Result	Type	Description
recLoc	object	The recording location meta data as an object

Description:

Gets the recording location meta data of an opened image by viewer id. The recording location is returned as an object, having the following properties:

- id [string] : The recording location id, which is equal to the image id.
- recordedAt [string] : The recording date and time as an ISO 8601 DateTime string
- usedInMeasurements [boolean] : If this recording location may be used in measurements
- x [float] : The x-element of the recording location defined in the set srs name
- y [float] : The y-element of the recording location defined in the set srs name
- z (optional) [float] : The z-element of the recording location defined in the set srs name
- sx (optional) [float] : The x-element of the recording location standard deviations in meters
- sy (optional) [float] : The y-element of the recording location standard deviations in meters
- sz (optional) [float] : The z-element of the recording location standard deviations in meters
- height (optional) [float] : Used if the z-coordinate is unknown
- sHeight (optional) [float] : The standard deviation of the height variable

6.2.3 Viewer functionality

6.2.3.1 openImage

Function name:	openImage	
Input parameter	**Type**	**Description**
value	string	The request value
type	uint	The request type (See description below)
viewerParams	object	Optional parameter. Default = null. Details about how to open an image
viewerID	uint	Optional parameter. Default = -1. The viewer in which to open the image
Result	**Type**	**Description**

Description:

This function attempts to open an image via image id, coordinate or address. Address searches are currently only supported within the Netherlands when the SRS name has been set to EPSG:28992. The various open image types expect the following value format:

`OPEN_IMAGE_TYPE_IMAGE_ID = 1`

The identification of an image.

`OPEN_IMAGE_TYPE_ADDRESS = 2`

The value is expected to be one of the following formats: *(x;y)* or *x y* (The fractions separator can be either a dot or a comma.)

`OPEN_IMAGE_TYPE_COORDINATE = 3`

The value is expected to be any of the following: *street house_nr, city,* or *zipcode house_nr*

The viewerParams object is expected to have the following properties:

- yaw [float] in degrees
- Pitch [float] in degrees
- hFov [float] in degrees
- dateFrom [ISO 8601 DateTime string]
- dateTo [ISO 8601 DateTime string] Optional parameter. Default = null

The viewer id can be used to replace an image within an existing viewer. When the viewer id is set to -1, the API will attempt to open a new viewer, or when the maximum amount of viewers is reached it will attempt to replace the image within the first viewer. An error is thrown if the image id was -1 but the recycle first image flag of the API was set to false (MAPI only).

6.2.3.2 getViewerState

Function name:	getViewerState	
Input parameter	Type	Description
viewerID	uint	The viewer of which to get the status.
Result	Type	Description
state	int	State of the viewer

Description:

WARNING: Since the API works in an asynchronous way, this status is only a snapshot of the moment the call was made.

Retrieves the status of a viewer by id.
Note that only the viewer with id = 0 can be active in the Basic API (BAPI), since the BAPI only contains only one viewer.

States:

```
UNAVAILABLE = -2
ERROR = -1
LOADING = 0
READY = 1
```

6.2.3.3 lookAtCoordinate

Function name:	lookAtCoordinate	
Input parameter	Type	Description
viewerID	uint	The viewer of which to change its view direction
x	float	The x-element of a terrestrial coordinate
y	float	The y-element of a terrestrial coordinate
z	float	The z-element of a terrestrial coordinate
Result	Type	Description

Description:

Centers the view direction of a viewer to a terrestrial coordinate as defined by the spatial reference system used within the API.

6.2.3.4 takeViewshot

Function name:	takeViewshot	
Input parameter	**Type**	**Description**
viewerID	uint	The viewer to make a 'screenshot' from
Result	**Type**	**Description**
Screenshot dimensions	int[]	[0]=width, [1]=height of the resulting image

Description: **Deprecated. Use *getViewerScreenshot* instead**

Takes a 'screenshot' of a single viewer. The image can be retrieved with a call to `getViewshotData`, with the right viewerID.

6.2.3.5 getViewshotData

Function name:	getViewshotData	
Input parameter	**Type**	**Description**
viewerID	uint	The viewer of which to retrieve the last taken screenshot.
Result	**Type**	**Description**
screenshot	string	Base64-encoded bitmap-data.

Description: **Deprecated. Use *getViewerScreenshot* instead**

Retrieves a viewshot that was taken with `takeViewshot`.

6.2.3.6 getViewerScreenshot

Function name:	getViewerScreenshot	
Input parameter	**Type**	**Description**
viewerID	uint	The viewer of which to retrieve a screenshot.
Result	**Type**	**Description**
screenshot	object	Object containing screenshot data

Description:

Gets a screenshot of a viewer. The screenshot is returned as an object, containing a `width`, `height`, and `data` property.
The `data` property consists of the base64 encoded pixel values of the image, where each pixel is represented by a uint. Each uint holds an RGB value of which the two most significant bytes reference the color red.

6.2.3.7 showViewerSaveDialog

Function name:	showViewerSaveDialog	
Input parameter	Type	Description
viewerID	uint	The viewer of which the image will be saved
Result	Type	Description
Description:		
Shows the used a save-dialog for a viewer, in order to save the image of the specified viewer to disk		

6.2.3.8 showViewerPrintDialog

Function name:	showViewerPrintDialog	
Input parameter	Type	Description
viewerID	uint	The viewer of which the image will be printed
Result	Type	Description
Description:		
Shows the user a print-dialog for a viewer, in order to print the image of the specified viewer.		

6.2.4 Layers: Recording locations, WFS and GML

6.2.4.1 showRecordingLocations

Function name:	showRecordingLocations	
Input parameter	Type	Description
name	string	User definable name of the layer
Result	Type	Description
Description:		
Show the recording locations layer in all viewers		

6.2.4.2 hideRecordingLocations

Function name:	hideRecordingLocations	
Input parameter	Type	Description
Result	Type	Description

Description:

Hide the recording locations layers

6.2.4.3 addWFSLayer

Function name:	addWFSLayer	
Input parameter	Type	Description
name	string	Name of the layer
url	string	URL of the WFS service used to draw feature data
typeName	string	Name(s) of the features to be drawn on the layer
version	string	The WFS version to be used. For example `"1.1.0"`
color	uint	The color used for drawing the vector data. A 24-bits RGB value is expected . For example 0x0000FF for blue
useProxy	boolean	Route WFS data through a CycloMedia proxy server (optional). Default is false
Result	Type	Description
layerId	uint	A reference to the added layer

Description:

Add a new WFS layer to all viewers.

Due to security restrictions in the flash player, it is sometimes required to route data via the CycloMeda proxy server. This restriction occurs when the domain on which the WFS service is host does not explicitly allow a flash client from another domain to use its data. (no crossdomain.xml available)

6.2.4.4 removeWFSLayer

Function name:	removeWFSLayer	
Input parameter	Type	Description
layerID	uint	A reference to the layer to be removed
Result	Type	Description
Description:		
Remove a WFS layer from all viewers		

6.2.4.5 addGMLLayer

Function name:	addGMLLayer	
Input parameter	Type	Description
name	string	Name of the layer to be added
gml	string	GML data
color	uint	The color used for drawing the vector data. A 24-bits RGB value is expected.
Result	Type	Description
layerID	uint	A reference to the added layer
Description:		
Add GML as a new layer onto all viewers		

6.2.4.6 removeGMLLayer

Function name:	removeGMLLayer	
Input parameter	Type	Description
layerID	uint	A reference to the layer to be removed
Result	Type	Description
Description:		
Remove a GML layer from all viewers		

6.2.5 Layers: Drawing markers

6.2.5.1 showDrawingLayer

Function name:	showDrawingLayer	
Input parameter	Type	Description
name	string	User definable name of the layer
Result	Type	Description

Description:

Makes the drawing layer visible

6.2.5.2 hideDrawingLayer

Function name:	hideDrawingLayer	
Input parameter	Type	Description
Result	Type	Description

Description:

Makes the drawing layer invisible. It is still possible to draw on the layer, only the result will not be visible

6.2.5.3 setDrawingMode

Function name:	setDrawingMode	
Input parameter	Type	Description
mode	uint	Drawing mode
Result	Type	Description

Description:

Sets the drawing mode. The drawing mode determines the form of the drawing

```
DRAWING_MODE_POINT = 1;
```

Draws a point

```
DRAWING_MODE_URL_IMAGE = 2
```

Draws an image. The image url must be set

```
DRAWING_MODE_CROSS_HAIR = 3
```

Draws a cross hair image

6.2.5.4 drawMarkerAtXY

Function name:	drawMarkerAtXY	
Input parameter	Type	Description
viewerId	Uint	ID of the viewer to draw the marker on
x	float	X coordinate of the pixel location
y	float	Y coordinate of the pixel location
label	string	Label for the marker (optional) Defaults to null
Result	Type	Description
drawingID	uint	ID of the added drawing
Description:		
Draws a marker at pixel x,y in a viewer (left, top = 0,0)		

6.2.5.5 drawMarkerAtHV

Function name:	drawMarkerAtHV	
Input parameter	Type	Description
viewerId	uint	ID of the viewer to draw the marker on
h	float	Horizontal direction in degrees
v	float	Vertical direction in degrees
label	string	Label for the marker (optional) Defaults to null.
Result	Type	Description
drawingID	uint	ID of the added drawing
Description:		
Draws a marker at position h,v in a viewer		

6.2.5.6 drawMarkerInDirection

Function name:	drawMarkerInDirection	
Input parameter	Type	Description
viewerId	uint	ID of the viewer to draw the marker on
x	float	X component of the direction
y	float	Y component of the direction
z	float	Z component of the direction
label	string	Label for the marker (optional) Defaults to null.
Result	Type	Description
drawingID	uint	ID of the added drawing

Description:

Draws a marker in direction x,y,z in a viewer.

6.2.5.7 clearDrawing

Function name:	clearDrawing	
Input parameter	Type	Description
viewerId	uint	The ID of the viewer to remove the drawing from
drawingID	uint	The ID of the drawing to be removed
Result	Type	Description

Description:

Removes a drawing from a viewer.

6.2.5.8 clearDrawings

Function name:	clearDrawings	
Input parameter	Type	Description
viewerId	uint	The ID of the viewer to remove the drawings from
Result	Type	Description

Description:

Removes all drawings from a viewer.

6.2.5.9 setDrawingImageURL

Function name:	setDrawingImageURL	
Input parameter	Type	Description
imageURL	string	The URL to a small PNG or JPEG image
Result	Type	Description

Description:

Sets the image url used for drawing.

Note that an crossdomain.xml should be available on the server where the image is hosted

6.2.5.10 setDrawingColor

Function name:	setDrawingColor	
Input parameter	Type	Description
color	uint	The color of a drawing as a 24-bit RGB value
Result	Type	Description

Description:

Sets the color used for drawing. This only applies to the non image-url drawing modes (dot, cross-hair, etc)

6.2.5.11 setDrawingSize

Function name:		
Input parameter	Type	Description
size	uint	Size of the drawing in pixels
Result	Type	Description

Description:

Sets the size used for drawing.

6.2.6 Layers: Measurement objects

Measurement objects are placed in a so-called '*entityLayer'*. From this point on measurement objects are called '*entities'*.

6.2.6.1 showEntityLayer

Function name:	showEntityLayer	
Input parameter	Type	Description
name	string	Name of the layer
Result	Type	Description
Description:		
Show the entity layer in all viewers.		

6.2.6.2 hideEntityLayer

Function name:	hideEntityLayer	
Input parameter	Type	Description
Result	Type	Description
Description:		
Hide the entity layers. (It's still possible to add entities)		

6.2.6.3 addHeightEntity

Function name:	addHeightEntity	
Input parameter	Type	Description
name	string	User defined name of the entity.
height	float	The initial height.
point	object	A 3D position referenced in the coordinate system as used by the API. Object height is measured relative to this position.
Result	Type	Description
	uint	ID of the entity.
Description:		
Adds a height entity to the list of entities, which can be used for measuring the height and/or position of an object.		

6.2.6.4 addAreaEntity

Function name:	addAreaEntity	
Input parameter	Type	Description
name	string	User defined name of the entity.
height	float	The initial height.
point1	object	A 2D position referenced in the coordinate system as used by the API.
point2	object	A 2D position referenced in the coordinate system as used by the API.
z	float	The height is measured relative to this value.
Result	Type	Description
	uint	ID of the entity.

Description:

Adds an area entity to the list of entities, which can be used for measuring the height, width and/or area of an object.

6.2.6.5 addVolumeEntity

Function name:	addVolumeEntity	
Input parameter	Type	Description
name	string	User defined name of the entity.
height	float	The initial height.
points	object[]	A list of 2D positions referenced in the coordinate system as used by the API.
z	float	The height is measured relative to this value.
Result	Type	Description

Description:

Adds a volume entity to the list of entities, which can be used for measuring the height, ground area, volume and/or perimeter of an object.

6.2.6.6 getEntityName

Function name:	getEntityName	
Input parameter	Type	Description
entityID	uint	A reference to an entity
Result	Type	Description
name	string	The name of the referenced entity
Description:		
Gets the name of a referenced entity.		

6.2.6.7 getEntityDescription

Function name:	getEntityDescription	
Input parameter	Type	Description
entityID	uint	A reference to an entity
Result	Type	Description
description	string	The description of the referenced entity
Description:		
Gets the description of a referenced entity.		

6.2.6.8 setEntityDescription

Function name:	setEntityDescription	
Input parameter	Type	Description
entityID	uint	A reference to an entity
description	string	A description of the entity or something referenced by the entity
Result	Type	Description
Description:		
Sets the description of a referenced entity.		

6.2.6.9 getEntityData

Function name:	getEntityData	
Input parameter	Type	Description
entityID	uint	A reference to an entity
Result	Type	Description
data	object	The data object

Description:

Get the data object associated with an entity. The data of the object depends on the entity referenced.

6.2.6.10 removeEntity

Function name:	removeEntity	
Input parameter	Type	Description
entityID	uint	A reference to an entity
Result	Type	Description

Description:

Remove an entity so it is no longer drawn by the application.

6.2.6.11 removeAllEntities

Function name:	removeAllEntities	
Input parameter	Type	Description
Result	Type	Description

Description:

Remove all entities from the application.

6.2.6.12 getFocusEntity

Function name:	getFocusEntity	
Input parameter	Type	Description
Result	Type	Description
	int	The id of the entity or -1 if null.

Description:

Retrieves the unique id of the entity that has focus.

6.2.6.13 setFocusEntity

Function name:	setFocusEntity	
Input parameter	Type	Description
entityID	int	The id of the entity or -1 if null.
Result	Type	Description

Description:

Sets the focus-entity by ID.

6.2.7 Language Settings

6.2.7.1 getLanguageLocale

Function name:	getLanguageLocale	
Input parameter	Type	Description
Result	Type	Description
	string	The locale of the language.

Description:

Returns the current language-locale.

6.2.7.2 setLanguageLocale

Function name:	setLanguageLocale	
Input parameter	Type	Description
locale	string	The locale of the language.
Result	Type	Description
	boolean	If the locale is supported.

Description:

Sets the current language-locale.

6.2.7.3 getSupportedLanguageLocales

Function name:	getSupportedLanguageLocales	
Input parameter	Type	Description
Result	Type	Description
	string[]	All supported locales.

Description:

Retrieves all supported language-locales.

7 Multi-Window API

This section describes the additional call-backs and methods available to the integrator when working with the Multi-Window level of the API.

7.1 Host side call-backs

Note that the remarks made at the start of section 5.1 are also valid for this section.

7.1.1 Viewer events

7.1.1.1 hst_viewerAdded

Function name:	hst_componentReady	
Parameter	Type	Description
viewerID	uint	The id of the viewer that was added.

Description:

Indicates to the host that a viewer was added from the MDI canvas.

7.1.1.2 hst_viewerRemoved

Function name:	hst_viewerRemoved	
Parameter	Type	Description
viewerID	uint	The id of the viewer that was removed.

Description:
Indicates to the host that a viewer was removed from the MDI canvas.

7.1.1.3 hst_viewerWindowResized

Function name:	hst_viewerWindowResized	
Parameter	Type	Description
viewerID	uint	The id of the viewer that was resized.
width	Number	The width of the window
height	Number	The height of the window

Description:
The id of the viewer which had its window resized

7.1.1.4 hst_viewerWindowSelected

Function name:	hst_viewerWindowSelected	
Parameter	Type	Description
viewerID	uint	The id of the viewer that was selected.

Description:
Indicates to the host that a viewer-window was selected

7.1.1.5 hst_viewerWindowMoved

Function name:	hst_viewerWindowMoved	
Parameter	**Type**	**Description**
viewerID	uint	The id of the viewer that was moved.
x	Number	The x coordinate of the new position of the window on the canvas
y	Number	The y coordinate of the new position of the window on the canvas

Description:

Indicates to the host that a viewer-window was moved.

7.1.1.6 hst_viewerWindowMinimized

Function name:	hst_viewerWindowMinimized	
Parameter	**Type**	**Description**
viewerID	uint	The id of the viewer that was minimized.

Description:

Indicates to the host that a viewer-window was minimized

7.1.1.7 hst_viewerWindowMaximized

Function name:	hst_viewerWindowMaximized	
Parameter	**Type**	**Description**
viewerID	uint	The id of the viewer that was maximized
width	Number	The new width of the window
height	Number	The new height of the window

Description:

Indicates to the host that a viewer-window was maximized

7.1.1.8 hst_viewerWindowRestored

Function name:	hst_viewerWindowRestored	
Parameter	**Type**	**Description**
viewerID	uint	The id of the viewer that was removed.
x	Number	The x-coord of the position
y	Number	The y-coord of the position
hidth	Number	The width of the window
height	Number	The height of the window

Description:

Indicates to the host that a viewer-window was restored

7.1.2 Measure events

7.1.2.1 hst_measurementCreated

Function name:	hst_measurementCreated	
Parameter	**Type**	**Description**
entityID	uint	The id of the measurement (entity) that has been created
entityType	String	The type of measurement created

Description:

Indicates to the host that a new measurement has been created. A measurement can only be edited until it is considered finished (after calling finishMeasurement) after which it is locked. After creating a new measurement, it can be interacted with using so called observations. An observation is a 2D point in an image that represents the projection the 3D point that is to be measured. Observations are added by clicking on an image, and can be manipulated by clicking on another location in the same image, or by dragging it around using mouse interaction. A measurement can only be created and edited through user interaction. After a measurement is finalized, it is locked from edition. Measurements are considered entities and can be deleted using the `removeEntity` function.

7.1.2.2 hst_measurementFinished

Function name:	hst_measurementFinished	
Parameter	**Type**	**Description**
entityID	uint	A reference to the measurement (entity)
entityData	Object	Measurement data of which the contents depends on the entity type

Description:

Indicates to the host that the created measurement has been finalized. After a measurement is finalized it is locked from edition. Measurements are considered entities and can be deleted using the removeEntity function.

7.1.2.3 hst_measurementUpdated

Function name:	hst_measurementUpdated	
Parameter	**Type**	**Description**
entityID	uint	A reference to the measurement (entity)
entityData	Object	Measurement data of which the contents depends on the entity type

Description:

Indicates to the host that the measurement was changed due to user interaction. This event is only fired after a measurement has been created, until the measurement is either finished or canceled.

7.1.2.4 hst_measurementCanceled

Function name:	hst_measurementCanceled	
Parameter	**Type**	**Description**
entityID	uint	A reference to the measurement (entity)

Description:

Indicates to the host that the user canceled the last active measurement. A measurement is canceled when the cancelMeasurement is called or if it was removed using `removeEntity` function.

7.1.2.5 hst_observationAdded

Function name:	hst_observationAdded	
Parameter	Type	Description
viewerD	uint	A reference to the viewer to which the observation was added
directions	Array	An array containing directional vectors

Description:

Indicates to the host that an observation (single click in a panoramic image) was added for the active measurement.

7.1.2.6 hst_observationRemoved

Function name:	hst_observationRemoved	
Parameter	Type	Description
viewerID	uint	A reference to the viewer in which the observation was removed

Description:

Indicates to the host that an observation was removed for the active measurement. An observation can be removed by closing the viewer that contained it.

7.1.2.7 hst_observationUpdated

Function name:	hst_observationUpdated	
Parameter	Type	Description
viewerID	uint	A reference to the viewer in which the observation was updated
directions	Array	An array containing directional vectors

Description:

Indicates to the host that an observation was updated for the active measurement.

7.2 Callable methods Multi-Window API

7.2.1 General MDI functionality

7.2.1.1 getViewerCount

Function name:	getViewerCount	
Input parameter	Type	Description
Result	Type	Description
	uint	The number of viewers opened.
Description:		
Get the number of viewers currently opened within the MDI.		

7.2.1.2 getMaxViewers

Function name:	getMaxViewers	
Input parameter	Type	Description
Result	Type	Description
	uint	The maximum number of viewers
Description:		
Returns the maximum number of viewers that can be opened by the MDI canvas.		

7.2.1.3 getAbsMaxViewers

Function name:	getAbsMaxViewers	
Input parameter	Type	Description
Result	Type	Description
	uint	The absolute maximum of viewers.
Description:		
Retrieves the absolute maximum number of viewers that can be opened by the MDI canvas.		

7.2.1.4 setMaxViewers

Function name:	setMaxViewers

Input parameter	Type	Description
	uint	The requested maximum number of viewers.
Result	Type	Description
	uint	The bounded maximum number of viewers.

Description:

Sets the maximum number of viewers (bounded by *[1...absMaxViewers]*).

7.2.1.5 getAutoTileViewers

Function name:	getAutoTileViewers	
Input parameter	Type	Description
Result	Type	Description
	boolean	Auto-tile.

Description:

Returns the API auto-tile flag.

7.2.1.6 setAutoTileViewers

Function name:	setAutoTileViewers	
Input parameter	Type	Description
autoTile	boolean	Auto-tile.
Result	Type	Description

Description:

Sets the API auto-tile flag which indicates that viewers are automatically arranged when a viewer is removed or added.

7.2.1.7 getRecycleFirstViewer

Function name:	getRecycleFirstViewer	
Input parameter	Type	Description

Result	Type	Description
recycle	boolean	Recycle-flag.

Description:

Returns the API recycle-flag which denotes a recycle of the first viewer in case a new image is opened but the maximum amount of viewers is reached.

7.2.1.8 setRecycleFirstViewer

Function name:	setRecycleFirstViewer	
Input parameter	Type	Description
recycle	boolean	Recycle-flag.
Result	Type	Description

Description:

Sets the API to recycle the first viewer in case a new image is opened but the maximum amount of viewers is reached.

7.2.1.9 getMDICanvasDimensions

Function name:	getMDICanvasDimensions	
Input parameter	Type	Description
Result	Type	Description
dimensions	Object	The dimensions of the canvas as an object having width and height as properties

Description:

Gets the dimensions of the canvas used for displaying all windows.

7.2.1.10 getMDIWindowingMode

Function name:	getMDIWindowingMode	
Input parameter	Type	Description
Result	Type	Description
windowingMode	uint	The current windowing mode

Description:

Gets the current MDI windowing mode. The selected mode determines the positioning behaviour of windows when they are added or removed to or from the MDI canvas.

See setMDIWindowingMode for available modes

7.2.1.11 setMDIWindowingMode

Function name:	setMDIWindowingMode	
Input parameter	Type	Description
windowingMode	uint	The windowing mode to be used
Result	Type	Description

Description:

Sets the current MDI windowing mode. The selected mode determines the positioning behaviour of windows when they are added or removed to or from the MDI canvas.

Supported windowing modes:

`TILE=1`
Automatically positions and resizes each window to fill the entire MDI canvas without overlapping the windows.

`CASCADE=2`
Windows are stacked on top of each other in a cascade like manner when added to the MDI canvas.

`MEASURE=3`
This windowing mode positions the map at the bottom half of the MDI canvas and tiles all other windows at the top half of the MDI canvas. If no map is available, all windows will be tiled horizontally in one row.

7.2.2 MDI calls per viewer window

7.2.2.1 selectViewerWindow

Function name:	selectViewerWindow	
Input parameter	**Type**	**Description**
viewerID	uint	ID of window to be set focus on
Description:		
Sets the focus on the indicated viewer-window		

7.2.2.2 moveViewerWindow

Function name:	moveViewerWindow	
Input parameter	**Type**	**Description**
viewerID	uint	ID of window to be set focus on
x	uint	The new x coordinate of the position of the window
y	uint	The new y coordinate of the position of the window
Description:		
Move the viewer window to an assigned position on the MDI canvas		

7.2.2.3 resizeViewerWindow

Function name:	resizeViewerWindow	
Input parameter	**Type**	**Description**
viewerID	uint	ID of window to be set focus on
width	uint	The new width of the size of the window
height	uint	The new height of the size of the window
Description:		
Resize the indicated viewer window to dimensions (width, height)		

7.2.2.4 minimizeViewerWindow

Function name:	minimizeViewerWindow	
Input parameter	Type	Description
viewerID	uint	A reference to a viewer window
Description:		
Minimizes the indicated viewer window.		

7.2.2.5 maximizeViewerWindow

Function name:	maximizeViewerWindow	
Input parameter	Type	Description
viewerID	uint	A reference to a viewer window
Description:		
Maximizes the indicated viewer window.		

7.2.2.6 restoreViewerWindow

Function name:	restoreViewerWindow	
Input parameter	Type	Description
viewerID	uint	A reference to a viewer window
Description:		
Restores the indicated viewer window.		

7.2.2.7 getViewerWindowColor

Function name:	restoreViewerWindow	
Input parameter	Type	Description
viewerID	uint	A reference to a viewer window
Result	Type	Description
color	uint	The color of the viewer window canvas as an RGB value. The two most significant bytes encode the color red
Description:		
Gets the color associated with the specified viewer window. The window colors provide a visual reference between opened location features on the map and its respective viewer window.		

7.2.3 Viewer functionality

7.2.3.1 removeViewer

Function name:	removeViewer	
Input parameter	Type	Description
viewerID	uint	The ID of a viewer.
Result	Type	Description

Description:

Removes a viewer from the MDI canvas using a viewer ID.
In order to add a viewer, simply open an image (see `openImage`) with a valid but unused viewerID.

7.2.4 Measurements

7.2.4.1 addPointMeasurement

Function name:	addPointMeasurement	
Input parameter	Type	Description
Name	string	Arbitrary name of entity
Result	Type	Description
Entity ID	uint	ID of measurement entity

Description:

Adds a point measurement entity to the list of entities, which can be used for measuring a 3D point. 3D points need to be measured using two or more cyclorama windows. After calling this function the user is able to click in the cyclorama windows to get a 3D point. The function finishPointMeasurement should be called when the measurement is finished.

Various events are fired that allows tracking of measurement creation, deletion and updates. The following events are fired throughout measuring:

- hst_measurementCreated
- hst_measurementFinished
- hst_measurementCanceled
- hst_measurementUpdated
- hst_observationAdded
- hst_observationRemoved
- hst_observationUpdated

7.2.4.2 addLineMeasurement

Function name:	addLineMeasurement	
Input parameter	Type	Description
Name	string	Arbitrary name of entity
Result	Type	Description
Entity ID	uint	ID of measurement entity

Description:

Adds a line measurement entity to the list of entities, which can be used for measuring a 3D line. After calling this function the user is able to click in the cyclorama windows to get a 3D line. The function finishMeasurement should be called when the measurement is finished

7.2.4.3 cancelMeasurement

Function name:	cancelMeasurement	
Input parameter	Type	Description
Entity ID	uint	ID of measurement entity
Result	Type	Description

Description:

Cancels currently active measurement

7.2.4.4 finishMeasurement

Function name:	finishMeasurement	
Input parameter	Type	Description
Entity ID	uint	ID of measurement entity
Result	Type	Description

Description:

Finalizes currently active measurement. A `hst_measurementFinished` event will be dispatched with the measurement results.

8 Full API

This section describes the additional call-backs and methods available to the integrator when working with the Full API.

8.1 Host-side call-backs

Note that the remarks made at the start of section 5.1 are also valid for this section.

8.1.1 General

8.1.1.1 hst_mapAdded

Function name:	hst_mapAdded	
Parameter	Type	Description
Description:		
Indicates to the host that the map was opened.		

8.1.1.2 hst_mapRemoved

Function name:	hst_mapRemoved	
Parameter	Type	Description
Description:		
Indicates to the host that the map was closed.		

8.1.1.3 hst_mapClicked

Function name:	hst_mapClicked	
Parameter	Type	Description
x	Number].	x-Coord of click in either world or window coords
y	Number	y-Coord of click in either world or window coords
Description:		
Indicates to the host that the map was clicked.		

8.1.1.4 hst_mapExtentChanged

Function name:	hst_mapExtentChanged	
Parameter	Type	Description
extent	Object	Bounds of the map (top, bottom, left, right)
center	Object	Position of the centre of the map (x, y)
zoom	Number	Zoom of the map
Description:		
Indicates to the host that the map extent has changed. Note that adapting the zoom or centre of the map changes the extent too.		

8.1.2 MDI events

8.1.2.1 hst_mapWindowResized

Function name:	hst_mapWindowResized	
Parameter	Type	Description
w		The width of the window.
h		The height of the window.
Description:		
Indicates to the host that the map-window has been resized.		

8.1.2.2 hst_mapWindowSelected

Function name:	hst_mapWindowSelected	
Parameter	Type	Description
Description:		
Indicates to the host that the map-window has been selected.		

8.1.2.3 hst_mapWindowMoved

Function name:	hst_mapWindowMoved	
Parameter	Type	Description
x		The x-coord. of the new position of the window on the canvas.
y		The y-coord. of the new position of the window on the canvas.
Description:		
Indicates to the host that the map-window has been moved.		

8.1.2.4 hst_mapWindowMinimized

Function name:	hst_mapWindowMinimized	
Parameter	Type	Description
Description:		
Indicates to the host that the map-window has been minimized.		

8.1.2.5 hst_mapWindowMaximized

Function name:	hst_mapWindowMaximized	
Parameter	Type	Description
w		The width of the window.
h		The height of the window.
Description:		
Indicates to the host that the map-window has been maximized.		

8.1.2.6 hst_mapWindowRestored

Function name:	hst_mapWindowRestored	
Parameter	Type	Description
x		The x-coord. of the position.
y		The y-coord. of the position.
w		The width of the window.
h		The height of the window.
Description:		
Indicates to the host that a viewer-window was restored.		

8.2 Callable Methods

8.2.1 General Map functionality

8.2.1.1 openMap

Function name:	openMap	
Input parameter	Type	Description
name	String	Name of the base layer
type	uint	Type of the base layer
params	Object	Layer parameters specific to the layer type
useProxy	Boolean	Route layer data through a CycloMedia proxy server (optional) Default is false.
Result	Type	Description
Description:		
Open the map having a specified base layer.		

8.2.1.2 getMapState

Function name:	getMapState	
Input parameter	Type	Description
Result	Type	Description
Description:		

Note: Since the API works in an asynchronous way, this status is only a snapshot of the moment the call was made.

Retrieves the status of the map.

```
UNAVAILABLE = -2
```

Indicates that the map does not exist at this time. Note that calls to methods on the map will almost certainly result in some exception.

```
READY = 1
```

Indicates that the map-window is open.

```
LOADING = 0
```

Indicates that the map exists but is still retrieving data.

8.2.1.3 closeMap

Function name:	closeMap	
Input parameter	Type	Description
Result	Type	Description
Description:		
Close the map.		

8.2.1.4 setMapClickMode

Function name:	setMapClickMode	
Input parameter	Type	Description
Result	Type	Description
Description:		

Retrieves the click-mode of the map:
`WINDOW = 0`
Returns window-coordinates to the host if the user clicks on the map.
`WORLD = 1`
Returns world-coordinates to the host if the user clicks on the map.		

8.2.1.5 getMapClickMode

Function name:	getMapClickMode	
Input parameter	Type	Description
Result	Type	Description
Description:		
Sets the click-mode of the map. For accepted values, please see setMapClickMode.		

8.2.2 Map properties

8.2.2.1 getMapExtent

Function name:	getMapExtent	
Input parameter	Type	Description
Result	Type	Description
Description:		
Retrieves the bounds of the map.		

8.2.2.2 setMapExtent

Function name:	setMapExtent	
Input parameter	Type	Description
bounds	Object	Object with 'top', 'bottom', 'left', 'right'
Result	Type	Description
Description:		
Sets the bounds of the map.		

8.2.2.3 getMapCenter

Function name:	getMapCenter	
Input parameter	Type	Description
Result	Type	Description
Description:		
Retrieve the maps centre.		

8.2.2.4 setMapCenter

Function name:	setMapCenter	
Input parameter	Type	Description
center	Object	Object with 'x', 'y' (Note: Either specify center as object OR x,y as numbers)
x	Number	x-Coord (Note: Either specify center as object OR x,y as numbers)
y	Number	y-Coord (Note: Either specify center as object OR x,y as numbers)
Result	Type	Description
Description:		
Sets the centre of the map. Either with a point-object, or with two loose coordinates.		

8.2.2.5 getMapZoom

Function name:	getMapZoom	
Input parameter	Type	Description
Result	Type	Description
Description:		
Returns the zoom of the map.		

8.2.2.6 setMapZoom

Function name:	setMapZoom	
Input parameter	Type	Description
zoom	Number	The zoom lvl of the map
Result	Type	Description
Description:		
Sets the zoom level of the map.		

8.2.3 Map MDI functionality

8.2.3.1 selectMapWindow

Function name:	selectMapWindow	
Input parameter	Type	Description
Result	Type	Description
Description:		
Sets the focus on the map-window.		

8.2.3.2 moveMapWindow

Function name:	moveMapWindow	
Input parameter	Type	Description
x	int	The new x-coord of the position of the window
y	int	The new y-coord of the position of the window
Result	Type	Description
Description:		
Moves the map-window to position (x,y).		

8.2.3.3 resizeMapWindow

Function name:	resizeMapWindow	
Input parameter	Type	Description
w		The width of the window.
h		The height of the window.
Result	Type	Description
Description:		
Resize the map-window to dimensions (w,h).		

8.2.3.4 minimizeMapWindow

Function name:	minimizeMapWindow	
Input parameter	Type	Description
Result	Type	Description
Description:		
Minimizes the map-window.		

8.2.3.5 maximizeMapWindow

Function name:	maximizeMapWindow	
Input parameter	Type	Description
Result	Type	Description
Description:		
Maximizes the map-window.		

8.2.3.6 restoreMapWindow

Function name:	restoreMapWindow	
Input parameter	Type	Description
Result	Type	Description
Description:		
Restores the map-window.		

8.2.4 Layer functionality

8.2.4.1 showMapRLSLayer

Function name:	showMapRLSLayer	
Input parameter	Type	Description
name	String	Name of the layer
Result	Type	Description
Description:		
Show the recording locations on the map.		

8.2.4.2 hideMapRLSLayer

Function name:	hideMapRLSLayer	
Input parameter	Type	Description
Result	Type	Description
Description:		
Hide the recording locations on the map.		

8.2.4.3 setMapRLSLayerDateRange

Function name:	setMapRLSLayerDateRange	
Input parameter	Type	Description
dateFrom	String	An ISO 8601 DateTime string indicating all recordings newer than specified date
dateTo	String	An ISO 8601 DateTime string indicating all recordings older than specified date Optional parameter. If left empty all recordings newer then `dateFrom` are displayed
Result	Type	Description
Description:		
Set the date range of the map recording location service layer. The `dateFrom` parameter represents the earliest date from which to show recording locations, up to the optional `dateTo` parameter. When no value of the `dateTo` parameter entered, the current system date/time is used.		

8.2.4.4 showMapEntityLayer

Function name:	showMapEntityLayer	
Input parameter	Type	Description
name	String	Name of the layer
minZoomLevel	uint	The zoom-level the layer will be first shown on
Result	Type	Description
Description:		
Show the entity-layer on the map.		

8.2.4.5 hideMapEntityLayer

Function name:	hideMapEntityLayer	
Input parameter	Type	Description
Result	Type	Description
Description:		
Hide the entity-layer of the map.		

8.2.4.6 addMapWFSLayer

Function name:	addMapWFSLayer	
Input parameter	Type	Description
name	String	Name of the layer
url	String	URL of the WFS service used to draw feature data
typeName	String	Name(s) of the features to be drawn on the layer Multiple types are comma seperated.
version	String	The WFS version to be used
color	uint	The color used for drawing the vector data A 24-bits RGB value is expected.
minZoomLevel	uint	The zoom-level the layer will be first shown on
useProxy	Boolean	Route WFS data through a CycloMedia proxy server (optional). Default is false.
Result	Type	Description

Description:		
Add a new WFS layer to the map. Due to security restrictions in the flash player, it is sometimes required to route data via the CycloMeda proxy server. This restriction occurs when the domain on which the WFS service is host does not explicitly allow a flash-client from another domain to use its data.		

8.2.4.7 addMapWMSLayer

Function name:	addMapWMSLayer	
Input parameter	Type	Description
name	String	Name of the layer
url	String	URL of the WMS service used to draw feature data
layer	String	Layer name on the server
version	String	The WMS version to be used
transparent	Boolean	Whether or not to display transparent pixels (if false, otherwise transparent pixels will be white)
tiled	Boolean	Whether or not the layer is tiled.
bgcolor	uint	The background color of the layer if it is not transparent (optional). Default is 0xFFFFFF.
useProxy	Boolean	Route WMS data through a CycloMedia proxy server (optional). Default is false.
Result	Type	Description
Description:		
Add a new WMS layer to the map. Due to security restrictions in the flash player, it is sometimes required to route data via the CycloMeda proxy server. This restriction occurs when the domain on which the WFS service is host does not explicitly allow a flash-client from another domain to use its data.		

8.2.4.8 addMapGMLLayer

Function name:	addMapGMLLayer	
Input parameter	Type	Description
name	String	Name of the layer to be added
gml	String	GML data
color	uint	The color used for drawing the vector data A 24-bits RGB value is expected.
minZoomLevel	uint	The zoom-level the layer will be first shown on
Result	Type	Description
Description:		
Add GML as a new layer onto the map.		

8.2.4.9 addMapOSMLayer

Function name:	addMapOSMLayer	
Input parameter	Type	Description
name	String	Name of the layer to be added
url	String	The url of the OSM layer
useProxy	Boolean	Route OSM data through a CycloMedia proxy server (optional). Default is false.
Result	Type	Description
Description:		
Add GML as a new layer onto the map.		

8.2.4.10 removeMapLayer

Function name:	removeMapLayer	
Input parameter	Type	Description
layerID	uint	A reference to the layer to be removed
Result	Type	Description
Description:		
Remove a layer from the map.		

8.2.5 Saving & printing

8.2.5.1 getMapScreenshot

Function name:	getMapScreenshot	
Input parameter	Type	Description
Result	Type	Description
screenshot	Object	An object containing screenshot data
Description:		
Gets a screenshot of the map. The screenshot is returned as an object, containing a width, height and data property. The data property consists of the base64 encoded pixel values of the image, where each pixel is represented by a uint. Each uint holds an RGB value of which the two most significant bytes reference the color red.		

8.2.5.2 showMapSaveDialog

Function name:	showMapSaveDialog	
Input parameter	Type	Description
Result	Type	Description
Description:		
Shows the used a save-dialog for the map, in order to save the image of the map to disk.		

8.2.5.3 showMapPrintDialog

Function name:	showMapPrintDialog	
Input parameter	Type	Description
Result	Type	Description
Description:		
Shows the user a print-dialog for the map, in order to print the image of the map.		

9 Examples

9.1 HTML

The following examples illustrates the most basic way to integrate a panorama window into an HTML page

```
<html>
        <body>
              <object>
                    <embed
                  src="https://www.globespotter.nl/v2/api/bapi/viewer_bapi.swf"
                              width="400" height="400"
                                    type="application/x-shockwave-flash"
                                    flashvars="&APIKey= --key--
                                    &MapSRSName=EPSG:28992&address=5555AA 1">
                        </embed>
              </object>
        </body>
</html>
```

A more sophisticated way of integrating a panorama window into an HTML page is shown in the following example:

```
<html>
    <head>
        <script language="javascript" src="AC_OETags.js"></script>
    </head>
    <body>
        <script language="JavaScript" type="text/javascript">

        // Version check for the Flash Player that has the ability to start Player
        //   Product Install (6.0r65)
        var hasProductInstall = DetectFlashVer(6, 0, 65);

        // Version check based upon the values defined in globals
        var hasRequestedVersion = DetectFlashVer(10, 0, 0);

        if ( hasProductInstall && !hasRequestedVersion ) {
            // DO NOT MODIFY THE FOLLOWING FOUR LINES
            // Location visited after installation is complete if
            // installation is required
            var MMPlayerType = (isIE == true) ? "ActiveX" : "PlugIn";
            var MMredirectURL = window.location;
            document.title = document.title.slice(0, 47) + " - Flash Player
                             Installation";
            var MMdoctitle = document.title;
```

```
            AC_FL_RunContent(
                "src", "playerProductInstall",
                "FlashVars",

              "MMredirectURL="+MMredirectURL+'&MMplayerType='+MMPlayerType+'&MMdoc
              title='+MMdoctitle+"",
                "width", "600",
                "height", "300",
                "align", "middle",
                "id", "viewer_bapi",
                "quality", "high",
                "bgcolor", "#ffffff",
                "name", "viewer_bapi",
                "allowScriptAccess","sameDomain",
                "type", "application/x-shockwave-flash",
                "pluginspage", "http://www.adobe.com/go/getflashplayer"
            );
        } else if (hasRequestedVersion) {

            // if we've detected an acceptable version
            // embed the Flash Content SWF when all tests are passed
            AC_FL_RunContent(
                    "src",
                   "https://www.globespotter.nl/v2/api/bapi/viewer_bapi.swf",
                    "width", "600",
                    "height", "300",
                    "align", "middle",
                    "id", "viewer_bapi",
                    "quality", "high",
                    "bgcolor", "#ffffff",
                    "name", "viewer_bapi",
                    "allowScriptAccess","sameDomain",
                    "type", "application/x-shockwave-flash",
                    "pluginspage", "http://www.adobe.com/go/getflashplayer",
                    "allowFullScreen", "true",
                    "FlashVars", " APIKey= --key--
                                  &MapSRSName=EPSG:28992&address=5555AA 1"
            );
          } else {  // flash is too old or we can't detect the plugin
            var alternateContent = 'Alternate HTML content should be placed here.
'
            + 'This content requires the Adobe Flash Player. '
            + '<a href=http://www.adobe.com/go/getflash/>Get Flash</a>';
            document.write(alternateContent);  // insert non-flash content
          }
        </script>
    </body>
</html>
```

This code performs validation on the installed flash player version and navigates the user to the adobe website when the player doesn't match the required version.

More examples can be found in the distributed API package or on the CycloMedia website. Contact CycloMedia for more information.

9.2 JavaScript

The HTML example can be extended with additional JavaScript functions that are called from the core API:

```
function hst_apiReady(apiState)
{
   if(apiState == true)
   {
      alert("API ready. Bapi major version: " +
       (viewer_bapi).getMajorVersion());
       (viewer_bapi).addRecordingLocations();
   }
}

// Indicates to the host that the SWF is ready for use.
// No parameters.
function hst_componentReady()
{
   try
   {
      // set srs
      (viewer_bapi).setApplicationParameter("EPSG:4326", 4);

      // set API key
      (viewer_bapi).setApplicationParameter
                    ("abcdefghijklmnopqrstuvwxyz0123456789", 5);


   }
   catch(error)
   {
      alert(error);
   }
}
```

More examples can be found in the distributed API package or on the CycloMedia website. Contact CycloMedia for more information.

9.3 C# Microsoft .Net

C# Microsoft .Net examples can be found in the distributed API package or on the CycloMedia website. Contact CycloMedia for more information.