Wireless Internet Platform for Interoperability 2.0.1

Part 5. Optional Specification of API

September 2004

Korea Wireless Internet Standardization Forum

1. Dividing, Adding, and Executing the Dynamic API

The divide, add, and execute functions of the dynamic API can be provided selectively. Therefore, the following specifications should be supported:

1.1. C API

- **MC_knlGetExecMName**

Prototype

M_Int32 MC_knlGetExecMName(M_Char* execName, M_Char* moduleName,
M_Char* rtnBuf, M_Int32 rtnBufSize)

Description

Obtains a name for the execution of the program installed on the program. The obtained name will be used as a parameter of MC_knlMExecute() and MC_knlMLoad().

Parameters

[in]	execName	Name obtained from MC_knlGetExecNames()
[in]	moduleName	Name of a module to be included in a jar and executed
[out]	rtnBuf	String that ends with a null value
[in]	rtnBufSize	Size of rtnBuf

Return Value

Pass

0

Fail

M_E_SHORTBUF When buf is too small to return all relevant names

Side Effect

None

Reference Item

None

- **MC_knlMExecute**

Prototype

M_Int32 MC_knlMExecute(char* symName, int parmCnt, ...)

Description

Executes a program installed on the program

When a program within a program is executed, the two have identical security level. When the program is large, this function divides the program into several small programs and executes partial loading; thus increasing program speed and executing a larger program than the heap supported by the platform in overlay form. Other operations are identical to those of the MC_knlExecute() function.

Parameters

[in]	symName	Symbolic name given by the developer when the program was developed
[in]	parmCnt	The number of parameters to be transmitted in a row after this parameter

Return Value

Pass

Library ID loaded

Fail

M_E_ACCESS	When the program has expired, or one has no access right
M_E_NOMEMORY	When there is insufficient memory
M_E_INVALID	When an invalid parameter is transmitted

Side Effect

None

Reference Item

None

- **MC_knlMLoad**

Prototype

M_Int32 MC_knlMLoad(char* symName, int parmCnt, ...)

Description

Loads the dynamic loading library installed on the program

As a library existing within a program, this library cannot be shared with other programs. When the program is large, this function divides the program into several small programs and executes partial loading; thus increasing program speed. Other operations are identical to those of the MC_knlLoad() function.

Parameters

[in]	symName	Symbolic name given by the developer when the program was developed
[in]	parmCnt	The number of parameters to be transmitted in a row after this parameter

Return Value

Pass

Program ID created

Fail

M_E_ACCESS	When the program has expired, or one has no access right
M_E_NOMEMORY	When there is insufficient memory
M_E_INVALID	When an invalid parameter is transmitted

Side Effect

None

Reference Item

None

1.2. Java API

- **getExecMName**

public static String getExecMName(java.lang.String execName, java.lang.String moduleName)

Obtains a name for the execution of the program installed on the program

The obtained name will be used as a parameter of mExecute() and mLoad().

Example:

```
String[] rtn;
String mName;
rtn = getExecNames(“mygame,” null, null);
mName = getExecMName(rtn[0], “stage1.bin”);
mExecute(mName, null);
(*Refer to a C example.)
```

Parameters

execName	Name obtained from getExecNames()
moduleName	Name of a module to be included in a jar and executed

Return Value

When the function succeeds, the name of a module being used in the platform is returned; when it fails, however, a null value is returned.

- **mExecute**

public static int mExecute(java.lang.String symName, java.lang.String[] args)

Executes a program installed on the program. When a program within a program is executed, the two have identical security level. When the program is large, this function divides the program into several small programs and executes partial loading; thus increasing program speed and executing a larger program than the heap supported by the platform in overlay form. Other operations are identical to those of the execute() function.

Parameters

symName	Symbolic name given by the developer when the program was

developed

args Parameter to be transmitted to the main method()

Return Value

When the function succeeds, the program ID of the executed program is returned; when it fails, however, a negative number is returned.

- **mLoad**

public static int mLoad(java.lang.String symName, java.lang.String[] args)

Loads the dynamic loading library installed on the program

As a library existing within a program, this library cannot be shared with other programs. When the program is large, this function divides the program into several small programs and executes partial loading; thus increasing program speed. Other operations are identical to those of the load() function.

Parameters

symName Symbolic name given by the developer when the program was developed

args Parameter to be transmitted to the main method()

Return Value

When the function succeeds, the program ID of the loaded program is returned; when it fails, however, a negative number is returned.

2. VGI-related API

2.1. Overview of VGI

- **Scope of API**

  The VGI (Vector Graphic Image) standard is a single API specification for processing VGI-related media in WIPI 2.0.1. The media currently supported through this API include Digital Aria's Mobile Flash and NeoMtel's SIS3.

  Depending on the VGI Open API specification defined, vector graphic-related solution companies can provide libraries to support their media in the future.

- **Common API**

  To establish a standard API on VGI in WIPI 2.0.1, the required part is designated as a common API based on Digital Aria's Mobile Flash Open API and NeoMtel's SIS3 API.

- **Naming Rule**

  A VGI common API of WIPI 2.0.1 should be defined such that it would have a prefix of MC_vgiXXX in the case of a function and MC_VgiXXX in the case of a structure.

2.2. Operation of the VGI Player

This is a function package related to a player processing VGIs.

VGI content data stored in a file or memory are abstracted into VGI instances to be played in a VGI player.

- **Creation and Extinction of the VGI Application Context**

  To use the VGI interface function, an application context should first be created using the MC_vgiInitialize function. One application can have only one application context, which is valid until the MC_vgiFinalize function is called.

- **Creation and Extinction of a VGI Instance**

  To play VGI content, a VGI instance should first be created using the MC_vgiCreateInstance function. When creating VGI instances, the application context is used as an argument. Play control functions used subsequently will be called using VGI instances as the first argument. Several VGI instances can be created and played for one application context, although it will consume too much memory.

- **Callback Function and Status Event**

Errors occurring during and at the end of the playing of the VGI content and unique operations carried out by content during playing are transmitted to the application using a callback function registered during MC_vgiCreateInstance or an event.

- **Defining the Play Area Related to VGI**

The area related to the playing of VGI content can be defined as shown in Figure 2-2-1. First, the area defined by the MC_vgiInitialize function corresponds to the "application" area in Figure 2-2-1. The area can be identical to the entire application LCD area, although it is defined here in anticipation of a possibility that only a part of it is actually used. The area defined by the MC_vgiPlay function is used to specify the size of the content to be played. Finally, the area defined by MC_vgiSetViewport is used to set the clipping area where the actual screen will be displayed within the application area. For example, if the area is set as shown in Figure 2-2-1, the actual screen where the VGI content is played will be limited to the area surrounded by the blue line. In contrast, MC_vgiSetViewPosition can be used to change the play location during the playing of the VGI content or to change the positions of (x, y). By defining the area, the usage of functions related to area setting can be provided to VGI library providers to enable them to set the play area as specified in Figure 2-2-1.

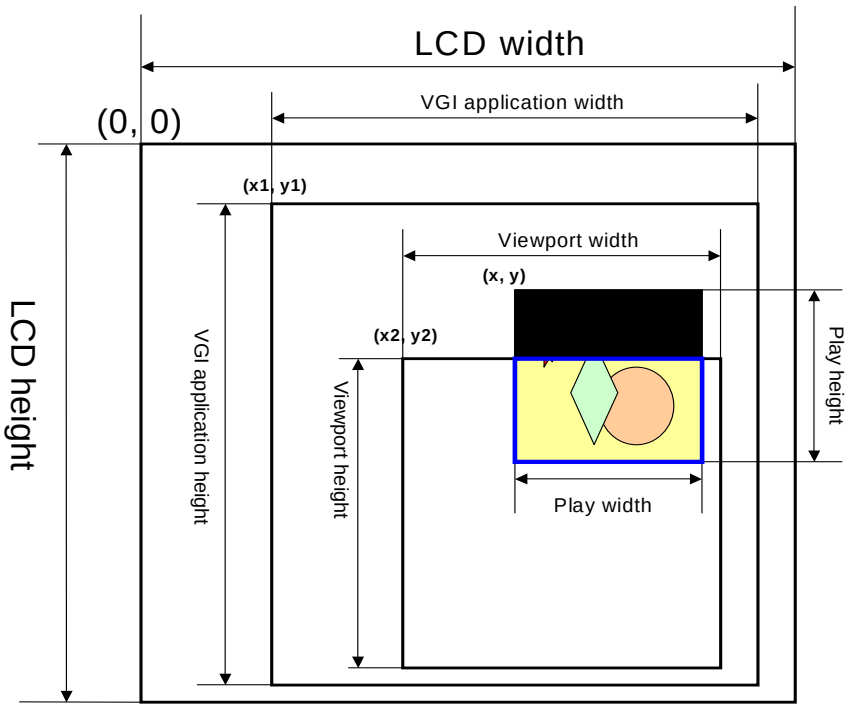

Figure 2-2-1. Example of play area setting.

2.3. VGI Message and Type Definition

- **MC_VGI_ZOOMIN**

Prototype

#define MC_VGI_ZOOMIN 0

Description

As a constant used in the MC_vgiZoom function, this function is used to zoom in the content and display.

- **MC_VGI_ZOOMOUT**

Prototype

#define MC_VGI_ZOOMOUT 1

Description

As a constant used in the MC_vgiZoom function, this function is used to zoom out the content and display.

- **MC_VGI_ZOOM100**

Prototype

#define MC_VGI_ZOOM100 2

Description

As a constant used in the MC_vgiZoom function, this function is used to display content in original ratio.

- **MC_VGI_PANLEFT**

Prototype

#define MC_VGI_PANLEFT 0

Description

A constant used in the MC_vgiPan function

- **MC_VGI_PANRIGHT**

Prototype

#define MC_VGI_PANRIGHT 1

Description

A constant used in the MC_vgiPan function

- **MC_VGI_PANUP**

Prototype

#define MC_VGI_PANUP 2

Description

A constant used in the MC_vgiPan function

- **MC_VGI_PANDOWN**

Prototype

#define MC_VGI_PANDOWN 3

Description

A constant used in the MC_vgiPan function

- **MC_VGI_EVENT**

Prototype

#define MC_VGI_EVENT MV_VGI_EVENT

Description

A message used in the callback function

- **MC_VGI_NOTIFY**

Prototype

#define MC_VGI_NOTIFY MV_VGI_NOTIFY

Description

Status handling

- **MC_VgiAppContext**

Prototype

typedef void* MC_VgiAppContext

Description

The application context for the use of the VGI interface

- **MC_VgiInstance**

Prototype

typedef void* MC_VgiInstance

Description

VGI instance handling

- **MC_VGIEvent**

Prototype

typedef struct _MC_VGIEvent{ M_Int32 notify; M_Char* nData;} MC_VGIEvent;

Description

In setting the cbevent_enable argument as TRUE when calling MC_vgiCreateInstance, the status change of the player can be transmitted to the application using the MV_VGI_EVENT and MV_VGI_NOTIFY events.

The first parameter of the event is MC_VgiInstance; the second parameter transmits a structure pointer having a MC_VGIEvent-type.

(*MC_VgiCallback) -- Since this event has some of the same functions as the callback function, choose whether you will use the event or callback function depending on the use.

```
MC_grpPostEvent(MC_knlGetCurProgramID(), MV_VGI_EVENT, MC_VgiInstance
hVgi, MC_VGIEvent* pVgiEvent)
```

Event	Factor		Description
	notify	nData	
MV_VGI_EVENT	M_E_NOMEMORY	x	Insufficient memory when playing VGI content
	M_E_INVALID	x	Invalid format for VGI content
	M_E_ERROR	x	Error due to other causes
	MC_VGI_PLAYEND	x	Occurs when the playing of the VGI content normally ends
MV_VGI_NOTIFY	MC_VGI_CALL	M_Char* phone	Alerts the user of a phone call from the VGI content at the phone number
	MC_VGI_SMS	M_Char* phone	Alerts the user of a message transmission from the VGI content to the phone number
	MC_VGI_URL	M_Char* url	Alerts the user of a message transmission of movement from the VGI content to url using the browser
	MC_VGI_TICK	x	Occurs when renderings are made for each frame of the VGI content

2.4. C API

- **MC_VgiCallback**

Prototype

typedef void (*MC_vgiCallback)(MC_VgiInstance hVgi, M_Int32 type, M_Int32 param1, M_Int32 param2)

Description

The callback function that is called until the status of the VGI library is changed.

Since this function has some of the same functions as the MC_VGIEvent event, choose whether you will use the event or callback function depending on the use.

Parameters

hVgi VGI instance

type Type of messages (MV_VGI_EVENT or MV_VGI_NOTIFY)

param1

param2

Type	Argument		Description
	Param1	Param2	
MC_VGI_EVENT	M_E_NOMEMORY	x	Insufficient memory when playing VGI content
	M_E_INVALID	x	Invalid format for VGI content
	M_E_ERROR	x	Error due to other causes
	MC_VGI_PLAYEND	x	Occurs when the playing of the VGI content normally ends
MC_VGI_NOTIFY	MC_VGI_CALL	M_Char* phone	Alerts the user of a phone call from the VGI content to the phone number
	MC_VGI_SMS	M_Char* phone	Alerts the user of a message transmission from the VGI content to the phone number
	MC_VGI_URL	M_Char* url	Alerts the user of a message transmission of movement from the VGI content to url using the browser

	MC_VGI_TICK	x	Occurs when renderings are made for each frame of the VGI content

Side Effect

None

Reference Item

MC_vgiCreateInstance, MC_vgiCreateInstanceByFile

- **MC_vgiGetContentInfoByFile**

Prototype

M_Int32 MC_vgiGetContentInfoByFile(M_Int32 fd, M_Byte* command, M_Byte* rtnBuf, M_Int32 bufsize)

Description

Reads content information directly from the file.

When the value to be returned is an integer, it will be converted into a decimal string before it is returned through the buffer. One way of identifying the content type easily is to use the signature of the file.

Parameters

[in] fd File identifier

[in] command Key value of the information to be read

Command	Remarks
"TYPE"	Distinguishes the content type before returning it "VIS" is returned in the case of VIS; "DMF" is returned in the case of Mobile Flash.
"WIDTH"	Width
"HEIGHT"	Height
"VERSION"	Version
"TOTAL_FRAME"	Total number of frames

[out] rtnBuf Buffer to which return information is returned

[in] bufSize Size of the buffer where the return value will be stored

Return Value

Pass

0

Fail

M_E_SHORTBUF Occurs when the size of the buffer transmitted is smaller than the returned string

M_E_INVALID An invalid parameter was transmitted.

Side Effect

None

Reference Item

MC_vgiGetContentInfo

- **MC_vgiGetContentInfo**

Prototype

M_Int32 MC_vgiGetContentInfo (MC_VgiInstance hVgi, M_Byte* command,
M_Byte* rtnBuf, M_Int32 bufsize)

Description

Reads content information; when the value to be returned is an integer, it will be converted into a decimal string before it is returned through the buffer.

Parameters

[in] hVgi VGI instance

[in] command Key value of the information to be read

Command	Remarks
"TYPE"	Distinguishes the content type before returning it "VIS" is returned in the case of VIS; "DMF" is returned in the case of Mobile Flash.
"WIDTH"	Width; the unit is in pixel
"HEIGHT"	Height; the unit is in pixel
"VERSION"	Version
"TOTAL_FRAME"	Total number of frames

[out] rtnBuf Buffer to which return information is returned

[in] bufSize Size of the buffer where the return value will be stored

Return Value

Pass

0

Fail

M_E_SHORTBUF Occurs when the size of the buffer transmitted is smaller than the returned string

M_E_INVALID An invalid parameter was transmitted.

Side Effect

None

Reference Item

None

- **MC_vgiSetContentInfo**

Prototype

M_Int32 MC_vgiSetContentInfo (MC_VgiInstance hVgi, M_Byte* command,
void* value)

Description

Changes the diverse information of the content on memory

Parameters

[in]	hVgi	VGI instance
[in]	command	Name of information to be changed
[in]	value	String representing the information value corresponding to the cmd

Command	Value
"LOOP"	When the value is "ON," the content is played continuously. When the value is "OFF," the content is played once.

Otherwise, different vendors have different commands.

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

None

- **MC_vgiInitialize**

Prototype

MC_VgiAppContext MC_vgiInitialize (M_Int32 x, M_Int32 y,M_Int32 width, M_Int32 height)

Description

Creates the context for the application program; the specified area denotes the LCD display area of the VGI application

Parameters

[in]	x	x starting point of the display area to be used by the VGI application
[in]	y	y starting point of the display area to be used by the VGI application
[in]	width	Width of the display area to be used by the VGI application
[in]	height	Height of the display area to be used by the VGI application

Return Value

Pass

Context of the VGI application program created

Fail

0

Side Effect

Has a different meaning from the values of x, y, width, and height used in MC_vgiPlay

Reference Item

MC_vgiPlay

- **MC_vgiFinalize**

Prototype

M_Int32 MC_vgiFinalize (MC_VgiAppContext VgiAc)

Description

Disables the context of the VGI application program created

Parameters

[in] VgiAc Context of the application program

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

None

- **MC_vgiCreateInstance**

Prototype

MC_VgiInstance MC_vgiCreateInstance(MC_VgiAppContext VgiAc, M_Byte* buf, M_int32 bufsize, MC_vgiCallback cbproc, M_Boolean cbevent_enable, void* extra);

Description

Creates a VGI instance from the content data stored in the memory

Parameters

[in]	VgiAc	Context of the application program
[in]	buf	Memory pointer where the content data is stored
[in]	bufsize	Size of the buffer where the content data is stored
[in]	cbevent_enable	TRUE when the receipt of the change of playing status as an event is preferred; otherwise, it is FALSE
[in]	extra	Used for fine control with additional information; varies depending on the vendor

Example:

“loop=1”	When the content is played continuously without limitation
“loop=0”	When the content is repeated only once
“bksound_off=1”	When the background sound of the content is not played
“bksound_off=0”	When the background sound of the content is played
“mem_limit=400”	A case wherein the memory limit is 400 KB; the VGI instance created will play the content within this memory limit
“performance=300”	Limit on the CPU use time; specifies the amount of CPU use in millisecond ; the VGI decoder releases the CPU occupation at this time interval for other tasks

Return Value

Pass

VGI instance created

Fail

0

Side Effect

None

Reference Item

MC_VGIEvent, MC_vgiCreateInstanceByFile

- **MC_vgiCreateInstanceByFile**

Prototype

MC_VgiInstance MC_vgiCreateInstanceByFile(MC_VgiAppContext VgiAc, M_Byte* filename, M_Int32 aMode, MC_vgiCallback cbproc, M_Boolean cbevent_enable void* extra);

Description

Creates a VGI instance from the content file

Parameters

[in]	VgiAc	Context of the application program
[in]	filename	File path/name of the content
	aMode	Access to the MC_DIR_PRIVATE_ACCESS private directory Access to the MC_DIR_SHARED_ACCESS shared directory Access to the MC_DIR_SYSTEM_ACCESS system directory
[in]	cbproc	Registers the callback function that is called when the playing ended or stopped due to an error. When a null value is registered, the callback function is not used.
[in]	cbevent_enable	TRUE when the receipt of the change of play status as an event is preferred; otherwise, it is FALSE
[in]	extra	Used for fine control with additional information; varies depending on the vendor
Example:		
“loop=1”		When the content is played continuously without limitation
“loop=0”		When the content is repeated only once
“bksound_off=1”		When the background sound of the content is not played
“bksound_off=0”		When the background sound of the content is played
“mem_limit=400”		A case wherein the memory limit is 400 KB; the VGI instance created will play the content within this memory limit
“performance=300”		Limit on the CPU use time; specifies the amount of CPU use in millisecond; the VGI decoder releases the CPU occupation at this time interval for other tasks

Return Value

Pass

VGI instance created

Fail

0

Side Effect

None

Reference Item

MC_VGIEvent, MC_vgiCreateInstance

- **MC_vgiPlay**

Prototype

M_Int32 MC_vgiPlay (MC_VgiInstance hVgi, M_Int32 x, M_Int32 y, M_Int32 width, M_int32 height)

Description

Starts playing the content; to stop playing the content, call the MC_vgiStop function

The area specified by this function denotes the play area of the content on the LCD display.

Parameters

[in]	hVgi	VGI instance
[in]	x	X coordinate of the area to which the VGI content will be outputted
[in]	y	Y coordinate of the area to which the VGI content will be outputted
[in]	width	Width of the area to which the VGI content will be outputted; when specifying 0, the default size of the content will be used
[in]	height	Height of the area to which the VGI content will be outputted; when specifying 0, the default size of the content

will be used

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

Has a different meaning from the values of x, y, width, and height used in MC_vgiInitialize

Reference Item

None

- **MC_vgiStop**

Prototype

M_Int32 MC_vgiStop (MC_VgiInstance hVgi)

Description

The MC_vgiStop function is called when you want to stop playing the content (called through the MC_vgiPlay function). When calling this function, it releases all the resources used for playing and informs the user that the content has stopped using the callback function or an event.

Parameters

[in][out]	hVgi	VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiPlay

- **MC_vgiPause**

Prototype

M_Int32 MC_vgiPause (MC_VgiInstance hVgi)

Description

Playing of content can be temporarily suspended by calling the MC_vgiPause function. To resume playing, call the MC_vgiResume function.

Parameters

[in] hVgi VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiPlay, MC_vgiResume

- **MC_vgiResume**

Prototype

M_Int32 MC_vgiResume (MC_VgiInstance hVgi)

Description

When the playing of content is temporarily suspended by calling the MC_vgiPause function, it can be resumed by calling the MC_vgiResume function.

Parameters

[in] hVgi VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiPause

- **MC_vgiReplay**

Prototype

M_Int32 MC_vgiReplay (MC_VgiInstance hVgi)

Description

When calling the MC_vgiReplay function while the content is being played (called through the MC_vgiPlay function), the content can be replayed from the start.

Parameters

[in] hVgi VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiPlay

- **MC_vgiLoopOn**

Prototype

M_Int32 MC_vgiLoopOn (MC_VgiInstance hVgi)

Description

This function is used to decide whether the content will be played continuously. Although continuous play option is set through an argument of MC_vgiCreateInstance, it can be changed while the content is being played using this function. Nonetheless, this function cannot be used when the playing has ended.

Parameters

[in] hVgi VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiLoopOff

- **MC_vgiLoopOff**

Prototype

M_Int32 MC_vgiLoopOff (MC_VgiInstance hVgi)

Description

This function is used to disable the continuous play option set through the MC_vgiLoopOn function. Although the continuous play option is set through an argument of MC_vgiCreateInstance, it can be changed while the content is being played (called through this function). Nonetheless, this function cannot be used when the playing has ended.

Parameters

[in] hVgi VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP The function is not provided.

M_E_ERROR Failed

Side Effect

None

Reference Item

MC_vgiLoopOn

- **MC_vgiSoundOn**

Prototype

M_Int32 MC_vgiSoundOn (MC_VgiInstance hVgi)

Description

When the sound of the content is turned off by calling the MC_vgiSoundOff function, it can be turned on again by calling the MC_vgiSoundOn function.

Parameters

[in] hVgi VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed
M_E_INVALIDSTATUS	The sound is not available.

Side Effect

None

Reference Item

MC_vgiSoundOff

- **MC_vgiSoundOff**

Prototype

M_Int32 MC_vgiSoundOff (MC_VgiInstance hVgi)

Description

The sound of the content can be turned off by calling the MC_vgiSoundOff function. The sound can be turned on again by calling the MC_vgiSoundOn function.

Parameters

[in] hVgi VGI instance

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiSoundOn

- **MC_vgiSetVolume**

Prototype

M_Int32 MC_vgiSetVolume (MC_VgiInstance hVgi, M_Int32 volume)

Description

The sound volume of the content being played using MC_vgiSetVolume can be changed into an absolute value given by calling MC_vgiPlay.

Parameters

[in]	hVgi	VGI instance
[in]	volume	Inputs an absolute value between 0 (Minimum) and 100 (Maximum) to set the sound volume

Return Value

Pass

0

Fail

M_E_NOTSUP	The set function is not provided.
M_E_ERROR	Setting failed

Side Effect

None

Reference Item

MC_vgiGetVolume

- **MC_vgiGetVolume**

Prototype

M_Int32 MC_vgiGetVolume (MC_VgiInstance hVgi, M_Int32 volume)

Description

MC_vgiPlay is used to return the sound volume of the content being played by calling the MC_vgiSetVolume function.

Parameters

[in] hVgi VGI instance

Return Value

Pass

Volume level (0 ~ 100)

Fail

M_E_NOTSUP	The corresponding function is not provided.
M_E_ERROR	Setting failed
M_E_ INVALIDSTATUS	The sound is not available.

Side Effect

None

- **MC_vgiZoom**

Prototype

M_Int32 MC_vgiZoom (MC_VgiInstance hVgi, M_Int32 mode)

Description

While the content is being played using MC_vgiPlay, the screen is enlarged or reduced every time the MC_vgiZoomIn function is called. The screen can also be restored to its original size. In addition, it can be enlarged up to 3 times. It increases in stages, i.e., by 100%, 200%, and 300%.

Parameters

[in]	hVgi	VGI instance	
[in]	mode	MC_VGI_ZOOMIN	ZOOMIN
		MC_VGI_ZOOMOUT	ZOOMOUT
		MC_VGI_ZOOM100	ZOOM100, Original size

Return Value

Pass

0

Fail

M_E_NOTSUP	The set function is not provided.
M_E_ERROR	Setting failed

Side Effect

None

Reference Item

MC_vgiPan

- **MC_vgiPan**

Prototype

M_Int32 MC_vgiPan (MC_VgiInstance hVgi, M_Int32 mode)

Description

When playing the content using MC_vgiPlay or in an enlarged state using MC_vgiZoomIn, the playing screen can be panned upward, downward, to the left, or to the right by calling the MC_vgiPan function. Panning can be repeated until the screen reaches the borders of the content by repeatedly calling the function.

Parameters

[in]	hVgi	VGI instance	
[in]	mode	MC_VGI_PANLEFT	PANLEFT
		MC_VGI_PANRIGHT	PANRIGHT
		MC_VGI_PANUP	PANUP
		MC_VGI_PANDOWN	PANDOWN

Return Value

Pass

0

Fail

M_E_NOTSUP	The set function is not provided.
M_E_ERROR	Setting failed.

Side Effect

None

Reference Item

MC_vgiZoom

- **MC_vgiHandleKeyEvent**

Prototype

M_Int32 MC_vgiHandleKeyEvent (MC_VgiInstance hVgi, M_Int32 eventcode,
M_Int32 keycode)

Description

When there is a key event from the user, and if the key does not have a key value that should be especially handled by the terminal, the key value should be transmitted to the VGI decoder. To transmit the key value, the MC_vgiHandleKeyEvent function is called. For the factor to be transmitted, MV_KEY_PRESS_EVENT, MV_KEY_RELEASE_EVENT and MV_KEY_REPEAT_EVENT are used as the value for the event code, and the KEY code in the MH_keyCode, for the key code.

Parameters

[in]	hVgi	VGI instance
[in]	eventcode	Event code inputted
[in]	keycode	Key code inputted

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

None

- **MC_vgiSetViewPosition**

Prototype

M_Int32 MC_vgiSetViewPosition (MC_VgiInstance hVgi, M_Int32 x, M_Int32 y)

Description

Sets the view position of the content

The view position can be changed anytime during playing after the VGI instance is created using this function.

Parameters

[in]	hVgi	VGI instance
[in]	x	X coordinate of the area to which the VGI content will be outputted
[in]	y	Y coordinate of the area to which the VGI content will be outputted

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiPlay, MC_vgiRedraw, MC_vgiSetViewport

- **MC_vgiRedraw**

Prototype

M_Int32 MC_vgiRedraw (MC_VgiInstance hVgi,M_Int32 x, M_Int32 y, M_Int32 width, M_Int32 height)

Description

This function is used when there is a need to redraw the content being played. The area to be redrawn is one that is determined through the argument inputted.

Parameters

[in]	hVgi	VGI instance
[in]	x	X starting point of the area to be redrawn
[in]	y	Y starting point of the area to be redrawn
[in]	width	Width of the area to be redrawn
[in]	height	Height of the area to be redrawn

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiPlay

- **MC_vgiSetViewport**

Prototype

M_Int32 MC_vgiSetViewport (MC_VgiInstance hVgi, M_Int32 x,M_Int32 y, M_Int32 width, M_Int32 height)

Description

Specifies the square clipping area

The content is not drawn outside of the clipping area. The starting point of the square starts at (x, y), and the function has separate parameters for the width and height.

Parameters

[in]	hVgi	VGI instance
[in]	x	X starting point of the clipping area
[in]	y	Y starting point of the clipping area
[in]	width	Width of the clipping area
[in]	height	Height of the clipping area

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

None

- **MC_vgiGetFrameOne[DigitalAria]**

Prototype

M_Int32 MC_vgiGetFrameOne (MC_VgiInstance hVgi, MC_GrpFrameBuffer fb, M_Int32 sx, M_Int32 sy, M_Int32 frame_number)

Description

Only one frame located in a particular position is decoded and stored in the frame buffer. This function is used when trying to store the content of a frame in a particular position directly in the buffer without calling the timer. It decodes as much as the width and the height of the frame buffer from the sx and sy of the frame image to be decoded. If the size of the frame image to be decoded is larger, it decodes as much as the size of the frame buffer. Similarly, if the size of the frame buffer is larger, it decodes as much as the size of the frame image.

The MC_VgiInstance instance type transmitted as an argument should be used after MC_vgiCreateInstance is called anew and assigned.

Parameters

[in]	hVgi	VGI instance
[out]	fb	Frame buffer
[in]	sx	X coordinate of the frame area
[in]	sy	Y coordinate of the frame area
[in]	frame_number	Frame number to be obtained

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

None

- **MC_vgiGetFrameOne[NeoMTel]**

Prototype

M_Int32 MC_vgiGetFrameOne (MC_VgiInstance hVgi, M_Byte* buf, M_Int32 width, M_Int32 height, M_Int32 frame_number)

Description

Only one frame located in a particular position is decoded and stored in the buffer. This function is used when trying to store the content of a frame in a particular position directly in the buffer without calling the timer. The MC_VgiInstance instance type transmitted as an argument should be used after MC_vgiCreateInstance is called anew and assigned.

Parameters

[in]	hVgi	VGI instance
[out]	buf	Frame buffer
[in]	width	Width of the frame buffer
[in]	height	Height of the frame buffer
[in]	frame_number	Frame number to be obtained

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

None

- **MC_vgiGetFrameMulti**

Prototype

M_Int32 MC_vgiGetFrameMulti(C_VgiInstance hVgi, M_Byte* buf, M_Int32 width, M_Int32 height, M_Int32 frame_number, M_Int32 next_frame)

Description

Only a number of frames located in a particular position are decoded and stored in the buffer. This function is used when trying to store the content of a frame in a particular position directly in the buffer without calling the timer. The MC_VgiInstance instance type transmitted as an argument should be used after MC_vgiCreateInstance is called anew and assigned.

Parameters

[in]	hVgi	VGI instance
[out]	buf	Frame buffer
[in]	width	Width of the frame buffer
[in]	height	Height of the frame buffer
[in]	frame_number	Frame number to be obtained
[in]	next_frame	When calling MC_vgiGetFrameMulti() continuously to obtain the next frame, set to 1; otherwise, set to 0.

Return Value

Pass

0

Fail

M_E_NOTSUP	The function is not provided.
M_E_ERROR	Failed

Side Effect

None

Reference Item

MC_vgiGetFrameOne

2.5. Java API (org.kwis.msp.vgi)

Interface VgiEventListener

public interface **VgiEventListener**

This is an interface on which the status of the VGI library alerts the user of the changes.

Detailed Description of the Field

- **VGI_NOMEMORY**

public static final int VGI_NOMEMORY

A constant that alerts the user of memory shortage during the playing of VGI content

- **VGI_INVALID**

public static final int VGI_INVALID

A constant that alerts the user of the invalid format of the VGI content

- **VGI_ERROR**

public static final int VG _ERROR

A constant that alerts the user of an error that occurred due to other causes

- **VGI_PLAYEND**

public static final int VGI_PLAYEND

A constant that notifies the user that the playing of the VGI content ended normally

- **VGI_STARTED**

public static final int VGI_STARTED

A constant that notifies the user that the playing of the VGI content started

- **VGI_STOPPED**

public static final int VGI_STOPPED

A constant that notifies the user that the playing of the VGI content stopped

- **VGI_PAUSED**

public static final int VGI_PAUSED

A constant that notifies the user that the playing of the VGI content was paused

- **VGI_RESUMED**

public static final int VGI_RESUMED

A constant that notifies the user that the playing of the VGI content resumed

- **VGI_REPLAYED**

public static final int VGI_REPLAYED

A constant that notifies the user of the replay of the VGI content from the start

- **VGI_CALL**

public static final int VGI_CALL

A constant that alerts the user of a phone call from the VGI content to the phone number

- **VGI_SMS**

public static final int VGI_SMS

A constant that alerts the user of a message transmission from the VGI content to the phone number

- **VGI_URL**

public static final int VGI_URL

A constant that notifies the user of the movement from the VGI content to url using the browser

- **VGI_TICK**

public static final int VGI_TICK

A constant that alerts the user of the renderings made for each frame of the VGI content

Detailed Description of the Method

- **notifyVgiEvent**

public void notifyVgiEvent(VgiClip clip, int event, char[] param)

A method that is called when there is a change in status during the playing of the VGI clip

Parameters

Clip	A clip in case of a status change
event	Status value
param	Used when there is an additional value to be transmitted for each

event

Event	Param	Description
VGI_CALL	char[] phone	Alerts the user of a phone call from the VGI content to the phone number
VGI_SMS	char[] phone	Alerts the user of a message transmission from the VGI content to the phone number
VGI_URL	char[] url	Alerts the user of a message transmission to move from the VGI content to url using the browser

Reference Item:

VgiClip.VgiClip (VgiBaseClip vBaseClip, byte[] buf, int bufsize, Boolean loop, Boolean bksound_off, VgiEventListener cbproc, Boolean cbevent_enable, byte[] extra), VgiClip.setListener (VgiClip hVgi, vgiEventListener listener)

Class VgiBaseClip

java.lang.Object
|
+--org.kwis.msp.vgi.**VgiBaseClip**

public class **VgiBaseClip** extends java.lang.Object

This is the uppermost class used to implement a media device.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detailed Description of the Field

Detailed Description of the Generator

- **VgiBaseClip**

public VgiBaseClip(int x, int y, int width, int height)

Parameters

x	X starting point of the display area to be used by the VGI application
y	Y starting point of the display area to be used by the VGI application
width	Width of the display area to be used by the VGI application
height	Height of the display area to be used by the VGI application Has a different meaning from the values of x, y, width, and height used in VgiClip

Detailed Description of the Method

Class VgiClip

java.lang.Object
|
+--org.kwis.msp.vgi.**VgiClip**

public class **VgiClip** extends java.lang.Object

This class implements clips played by VgiPlayer.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detailed Description of the Field

Detailed Description of the Generator

- **VgiClip**

public VgiClip(VgiBaseClip vBaseClip, byte[] buf, int bufsize, VgiEventListener cbproc, Boolean cbevent_enable, byte[] extra)

Parameters

vBaseClip	vgi base clip
buf	Memory pointer where the content data is stored
bufsize	Size of the buffer where the content data is stored
cbproc	Registers the callback function that is called when the playing ended or stopped due to an error; registers a null value; when a null value is registered, the callback function is not used
cbevent_enable	TRUE when the receipt of the change of play status as an event is preferred; otherwise, it is FALSE
extra	Used for fine control with additional information; varies depending on the vendor
Example:	
“loop=1”	When the content is played continuously without limitation
“loop=0”	When the content is repeated only once
“bksound_off=1”	When the background sound of the content is not played
“bksound_off=0”	When the background sound of the content is played
“mem_limit=400”	A case wherein the memory limit is 400 KB; the VGI instance created will play the content within this memory

	limit
“performance=300”	Limit on the CPU use time; specifies the amount of CPU use in millisecond; the VGI decoder releases the CPU occupation at this time interval for other tasks

- **VgiClip**

public VgiClip(VgiBaseClip vBaseClip, byte[] filename, int aMode, VgiEventListener cbproc, Boolean cbevent_enable, byte[] extra)

Parameters

vBaseClip	Vgi base clip	
filename	File path/name of the content	
aMode	MC_DIR_PRIVATE_ACCESS	Access to private directory
	MC_DIR_SHARED_ACCESS	Access to shared directory
	MC_DIR_SYSTEM_ACCESS	Access to system directory
loop	TRUE when playing the content continuously without limitation; otherwise, it is FALSE	
bksound_off	TRUE when the background sound of the content is not played; otherwise, it is FALSE	
cbproc	Registers the Listener function that is called when the playing ended or stopped due to an error; when a null value is registered, the listener function is not used	
cbevent_enable	TRUE when the receipt of the change of play status as an event is preferred; otherwise, it is FALSE	
extra	Used for fine control with additional information; varies depending on the vendor	

Example:

“loop=1”	When the content is played continuously without limitation
“loop=0”	When the content is repeated only once
“bksound_off=1”	When the background sound of the content is not played
“bksound_off=0”	When the background sound of the content is played
“mem_limit=400”	Memory limit; the VGI instance created will play the content within this memory limit
“performance=300”	Limit on the CPU use time; specifies the amount of CPU use in millisecond; the VGI decoder releases the CPU occupation at this time interval for other tasks

Detailed Description of the Method

Class VgiPlayer

java.lang.Object
|
+--org.kwis.msp.vgi.**VgiPlayer**

public class **VgiPlayer** extends java.lang.Object

This class includes methods used to play the VGI media.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detailed Description of the Field

- **E_SHORTBUF**

public static final int E_SHORTBUF

An error constant that is displayed when the buffer of the data to be returned according to the command is small

- **E_INVALID**

public static final int E_INVALID

An error constant that is displayed when the command is invalid

- **E_NOTSUP**

public static final int E_NOTSUP

An error constant that is displayed when the function is unsupported

- **E_ERROR**

public static final int E_ERROR

An error constant

- **VGI_ZOOMIN**

public static final int VGI_ZOOMIN

A vgiZoom constant that is used to zoom in the content and display

- **VGI_ZOOMOUT**

public static final int VGI_ZOOMOUT

A vgiZoom constant that is used to zoom out the content and display

- **VGI_ZOOM100**

public static final int VGI_ZOOM100

A vgiZoom constant that is used to display the content in original ratio

- **VGI_PANLEFT**

public static final int VGI_PANLEFT

A vgiPan constant

- **VGI_PANRIGHT**

public static final int VGI_PANRIGHT

A vgiPan constant

- **VGI_PANUP**

public static final int VGI_PANUP

A vgiPan constant

- **VGI_PANDOWN**

public static final int VGI_PANDOWN

A vgiPan constant

Detailed Description of the Generator

Detailed Description of the Method

- **notifyEvent**

public boolean notifyEvent(VgiClip hVgi, int event, char[] param)

Alerts the user of the change of status during the playing of a clip; the event transmitted is the same as VgiEventListener.notifyVgiEvent()

Parameters

hVgi	vgi clip
event	Status value
param	Used when there is an additional value to be transmitted for each event

- **setListener**

public int setListener(VgiClip hVgi, VgiEventListener listener)

Registers the Listener that will alert the user of the status change during the playing of a clip

Parameters

hVgi	vgi clip
listener	A new Listener; In case of a null value, the method removes the existing Listener.

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **getContentInfo**

public int getContentInfo(VgiClip hVgi, byte[] command, byte[] rtnBuf, int bufsize)

Reads content information; when the value to be returned is an integer, it will be converted into a decimal string before it is returned through the buffer.

Parameters

hVgi vgi clip

[in] command Key value of the information to be read

Command	Remarks
"TYPE"	Distinguishes the content type before it is returned "VIS" is returned in the case of VIS; "DMF" is returned in the case of Mobile Flash.
"WIDTH"	Width
"HEIGHT"	Height
"VERSION"	Version
"TOTAL_FRAME"	Total number of frames

[out] rtnBuf Buffer to which the information to be returned is returned

[in] bufSize Size of the buffer where the return value will be stored

Return Value

Pass

0

Fail

E_SHORTBUF Occurs when the size of the buffer transmitted is smaller than the returned string

E_INVALID An invalid parameter was transmitted.

- **SetContentInfo**

public int SetContentInfo (VgiClip hVgi, byte[] command, byte[] value)

Changes the diverse information of the content being played on memory

Parameters

	hVgi	vgi clip
[in]	command	Information name to be changed
[in]	value	A string representing the information value corresponding to the cmd

Command	Value
"LOOP"	When "ON," the method plays the content repetitively. When "OFF," the method plays the content once.

Return Value

Pass

0

Fail

E_SHORTBUF	Occurs when the size of the buffer transmitted is smaller than the returned string
E_INVALID	An invalid parameter was transmitted.

- **play**

public int play(VgiClip hVgi, int x, int y, int width, int height)

Plays the clip data. When this function is called to begin media processing, the start status will be transmitted to the event listener function registered in the clip. When the user tries to replay the clip being played, this function does not play any role. When the playing ends, the VGI_PLAYEND status will be transmitted to the event listener function.

Parameters

hVgi	Clip to be played
repeat	One time playing when false; continuous playing when true

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **stop**

public int stop(VgiClip hvgi)

Ends media playing

When this function is called, and the handling of the media is completed, the stop status is transmitted to the Event Listener function registered in the clip. When the user calls this function again from the stopped handler, this function does not play any role.

Parameters

hvgi Clip to be ended

Return Value

Pass

0

Fail

M_E_NOTSUP The function is not provided.

M_E_ERROR Failed

- **pause**

public int pause(VgiClip hvgi)

Temporarily suspends the playing of the media

When this function is called, and the handling of the media is paused, the pause status is transmitted to the Event Listener function registered in the clip. When calling this function again from the temporarily suspended or stopped handler, this function does not play any role.

Parameters

hvgi Clip to be suspended temporarily

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **resume**

public int resume(VgiClip hvgi)

Resumes the playing of the media suspended temporarily

When this function is called, and the handling of the media is resumed, the resume status is transmitted to the Event Listener function registered in the clip. When calling this function again from the handler handling the media, this function does not play any role.

Parameters

hvgi Clip to be resumed

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **replay**

public int replay(VgiClip hvgi)

When calling the vgiReplay method while the content is being played (called through the vgiPlay method), the content will be replayed from the beginning.

Parameters

hVgi vgi clip

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **loopOn**

public int loopOn(VgiClip hvgi)

This function is used to determine whether the content will be played continuously. Although the continuous play option is set through an argument during the creation of VgiClip, it can be changed with this function while the content is being played. Nonetheless, this function cannot be used when the playing of the content has ended.

Parameters

hVgi vgi clip

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **loopOff**

public int loopOff(VgiClip hvgi)

This function is used to disable the continuous play option of the content set by the MC_vgiLoopOn function. Although the continuous play option is set through an argument of MC_vgiCreateInstance, it can be changed with this function while the content is being played. Nonetheless, this function cannot be used when the playing of the content has ended.

Parameters

hVgi vgi clip

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **soundOn**

public int soundOn(VgiClip hVgi)

This function is used to turn on the sound again when the sound of the content being played was turned off (called through the SoundOff function).

Parameters

hVgi vgi clip

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed
E_INVALIDSTATUS	The sound is not available.

- **soundOff**

public int soundOff(VgiClip hVgi)

This function is used to turn off the sound of the content. The sound can be turned on again by calling the soundOn function.

Parameters

hVgi vgi clip

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **setVolume**

public int setVolume(VgiClip hvgi)

Changes the sound volume of the content

Parameters

hVgi vgi clip

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **zoom**

public int zoom(VgiClip hvgi, int mode)

While the content is being played using play, the screen is enlarged or reduced every time the zoom function is called. The screen can also be restored to its original size. In addition, it can be enlarged up to 3 times, with minimum reduction ratio of 100%. It increases in stages, i.e., by 100%, 200%, and 300%.

Parameters

hVgi vgi clip

mode

VGI_ZOOMIN	ZOOMIN
VGI_ZOOMOUT	ZOOMOUT
VGI_ZOOM100	ZOOM100, Original size

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **pan**

public int pan(VgiClip hvgi, int mode)

When playing the content using play or in an enlarged state using the zoom function, the playing screen can be panned upward, downward, to the left, or to the right by calling the pan function. Panning can be repeated until the screen reaches the borders of the content by calling this function repeatedly.

Parameters

hVgi	vgi clip	
mode		
	VGI_PANLEFT	PANLEFT
	VGI_PANRIGHT	PANRIGHT
	VGI_PANUP	PANUP
	VGI_PANDOWN	PANDOWN

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **handleKeyEvent**

public int handleKeyEvent(VgiClip hvgi, int eventcode, int keycode)

When there is a key event from the user, and if the key does not have a key value that should be especially handled by the terminal, the key value should be transmitted to the VGI decoder. To transmit the key value, the handleKeyEvent function is called. For the factor to be transmitted, MV_KEY_XXX is used as the value for event code, and MC_KEY_XXX, for key code.

Parameters

hVgi	vgi clip
eventcode	Event code
keycode	Key code inputted

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **setViewPosition**

public void setViewPosition(VgiClip hvgi, int x, int y)

Sets the view position of the content; the view position can be changed anytime during the playing after a VGI instance is created using this function

Parameters

hVgi vgi clip

x X coordinate of the area to which the VGI content will be outputted

y Y coordinate of the area to which the VGI content will be outputted

Return Value

Pass

0

Fail

E_NOTSUP The function is not provided.

E_ERROR Failed

- **reDraw**

public int reDraw(VgiClip hvgi, int x, int y, int width, int height)

This function is used when there is a need to redraw the content being played. The area to be redrawn is one that is determined through the argument inputted.

Parameters

hVgi vgi clip

x X starting point of the area to be redrawn

y Y starting point of the area to be redrawn

width	Width of the area to be redrawn
height	Height of the area to be redrawn

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **setViewport**

public int setViewPort(VgiClip hvgi, int x, int y, int width, int height)

Specifies the square clipping area; the content is not drawn outside of the clipping area

Parameters

hVgi	vgi clip
x	X starting point of the clipping area
y	Y starting point of the clipping area
width	Width of the clipping area
height	Height of the clipping area

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **getFrameOne**

public int getFrameOne(VgiClip hvgi, byte[] buf, int width, int height, int frame_number)

Only one frame located in a particular position is decoded and stored in the frame buffer. This function is used when trying to store the content of a frame in a particular position

directly in the buffer without calling the timer. The VgiClip instance type transmitted as an argument should be used after it is assigned anew.

Parameters

hVgi	vgi clip
buf	Frame buffer
sx	X coordinate of the frame area
sy	Y coordinate of the frame area
frame_number	Frame number to be obtained

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed

- **getFrameMulti**

public int getFrameMulti(VgiClip hvgi, byte[] buf, int width, int height, int frame_number, int next_frame)

Only a number of frames located in a particular position are decoded and stored in the frame buffer. This function is used when trying to store the content of a frame in a particular position directly in the buffer without calling the timer. The VgiClip instance type transmitted as an argument should be used after it is assigned anew.

Parameters

hVgi	vgi clip
buf	Frame buffer
width	Width of the frame buffer
height	Height of the frame buffer
frame_number	Frame number to be obtained

next_frame	When calling getFrameMulti() continuously to obtain the next frame, set to 1; otherwise, set to 0.

Return Value

Pass

0

Fail

E_NOTSUP	The function is not provided.
E_ERROR	Failed