in

COLLABORATORS			
	TITLE : in		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

in

1.1 commodityclass.guide

```
 Search
TABLE OF CONTENTS

commodityclass/--background--
commodityclass/CM_ADDHOTKEY
commodityclass/CM_DISABLEBROKER
commodityclass/CM_DISABLEHOTKEY
commodityclass/CM_ENABLEBROKER
commodityclass/CM_ENABLEHOTKEY
commodityclass/CM_MSGINFO
commodityclass/CM_REMHOTKEY
commodityclass/COMM_Description
commodityclass/COMM_ErrorCode
commodityclass/COMM_Name
commodityclass/COMM_Notify
commodityclass/COMM_Priority
commodityclass/COMM_ShowHide
commodityclass/COMM_SigMask
commodityclass/COMM_Title
commodityclass/COMM_Unique
```

1.2 commodityclass/--background--

```
    NAME
  Class:    commodityclass
  Superclass: ROOTCLASS
  Include File: <bgui/bgui_cx.h>

    FUNCTION
  To  provide  a  BOOPSI based interface to the commodities.library.
  This  class  will allow you to setup and maintain a simple broker with
  simple hotkeys.  The attached hotkeys can also be disabled  separately
  and enabled.
```

1.3 commodityclass/CM_ADDHOTKEY

```
  NAME
CM_ADDHOTKEY -- Add a hotkey to the broker.

  SYNOPSIS
err = DoMethod( obj, CM_ADDHOTKEY, desc, id, flags );

ULONG   err;
STRPTR    desc;
ULONG   id;
ULONG   flags;

  FUNCTION
This  method  must  be  used  to  attach  hotkeys  to the broker.  The
hotkey is defined with an input description string just like that used
by the amiga.lib HotKey() function.

  INPUTS
desc  - This  must  point  to  a  string  in  which  the  hotkey  is
  described.  The input description string is the same string as
  you  would pass the HotKey()  routine  from the amiga.lib link
  library.

id  - This  must be an  integer  which  represents  the  ID of the
  key.  This  must  be a unique value as this ID will be used to
  identify the key. Please note  that  you  should  keep  the ID
  between 1 and 65535 (I.E. In  the  lower  16 bits.) if you use
  the V40 style of message handling.

flags - This  field  may have  any of the following flags set:

  CAHF_DISABLED - When set the key is added to the broker but it
    is disabled from usage.

  RESULT
err - TRUE for success, FALSE for failure.  When FALSE is returned
  you can use the COMM_ErrorCode attribute to find  out  what it
  was that failed.

  EXAMPLE
Object  *broker;
#define ID_SF1    1

/*
 *  Add shift+f1 as a hotkey to the
 *  broker object.
 */
DoMethod( broker, CM_ADDHOTKEY, "shift f1", ID_SF1, 0L );

  SEE ALSO
CM_REMHOTKEY, CM_MSGINFO, COMM_ErrorCode, amiga.lib/HotKey()
```

1.4 commodityclass/CM_DISABLEBROKER

```
  NAME
CM_DISABLEBROKER -- Disable the commodity broker.

  SYNOPSIS
DoMethod( obj, CM_DISABLEBROKER );

  FUNCTION
To disable the commodity broker.

  INPUTS

  RESULT

  EXAMPLE
Object    *broker;

/*
 *  Disable commodity broker.
 */
DoMethod( broker, CM_DISABLEBROKER );

 SEE ALSO
CM_ENABLEBROKER
```

1.5 commodityclass/CM_DISABLEHOTKEY

```
  NAME
CM_DISABLEHOTKEY -- Disable a hotkey.

  SYNOPSIS
err = DoMethod( obj, CM_DISABLEHOTKEY, id );

ULONG   err;
ULONG   id;

  FUNCTION
To disable a hotkey. It is not necessary to disable the  broker  to do
this. This method will take care of this for you.

  INPUTS
id  - The ID of the key to disable.

  RESULT
err - True for succes, FALSE for failure.

  EXAMPLE
Object    *broker;
#define         ID_SF1    1

/*
 *  Disable the hotkey with this ID.
 */
```

```
DoMethod( broker, CM_DISABLEHOTKEY, ID_SF1 );

  SEE ALSO
CM_ENABLEHOTKEY
```

1.6 commodityclass/CM_ENABLEBROKER

```
  NAME
CM_ENABLEBROKER -- Enable the commodity broker.

  SYNOPSIS
DoMethod( obj, CM_ENABLEBROKER );

  FUNCTION
To enable the commodity broker.

  INPUTS

  RESULT

  EXAMPLE
Object    *broker;

/*
 *  Enable commodity broker.
 */
DoMethod( broker, CM_ENABLEBROKER );

 SEE ALSO
CM_DISABLEBROKER
```

1.7 commodityclass/CM_ENABLEHOTKEY

```
  NAME
CM_ENABLEHOTKEY -- Enable a hotkey.

  SYNOPSIS
err = DoMethod( obj, CM_ENABLEHOTKEY, id );

ULONG   err;
ULONG   id;

  FUNCTION
To enable a hotkey. It is not necessary to disable the  broker  to  do
this. This method will take care of this for you.

  INPUTS
id  - The ID of the key to enable.

  RESULT
err - True for success, FALSE for failure.
```

```
  EXAMPLE
Object    *broker;
#define         ID_SF1    1

/*
 *  Enable the hotkey with this ID.
 */
DoMethod( broker, CM_ENABLEHOTKEY, ID_SF1 );

  SEE ALSO
CM_DISABLEHOTKEY
```

1.8 commodityclass/CM_MSGINFO

```
  NAME
CM_MSGINFO -- Examine commodity broker messages.

  SYNOPSIS
cont = DoMethod( obj, CM_MSGINFO, type, id, data );

ULONG    cont;
ULONG   *type;
ULONG   *id;
ULONG   *data;

  FUNCTION
To get the pending commodity messages and extract  the  necessary data
from them. You supply storage space for the data you need to know.

  INPUTS
type  - When non-NULL the  result of  CxMsgType()  on the  commodity
  message is stored here.

id  - When non-NULL  the  result of  CxMsgID()  on  the  commodity
  message is stored here.

data  - When non-NULL the  result of  CxMsgData()  on the  commodity
  message is stored here.

  RESULT
CMMI_NOMORE  when  all  messages  have been processed.   You must keep
calling this method until CMMI_NOMORE is returned.

Since V40 the class can also return any of the following return codes:

  CMMI_KILL -- Uppon  receiving  this  return  code  you  should
    remove yourself from the system and terminate.

  CMMI_DISABLE -- This return code tell's you that the broker is
    disabled. At the time you get this message  the broker
    has already been disabled.

  CMMI_ENABLE -- When this is returned the broker is enabled. At
    the time you get this message the broker  has  already
    been enabled.
```

```
  CMMI_UNIQUE -- When  this  is  returned  another commodity has
    been  attemted  to  run  using your broker name.   You
    should let the user know this happened. Usually you do
    this by popping up your GUI.

  CMMI_APPEAR -- This is returned when you are requested to open
    up the GUI of the commodity.

  CMMI_DISAPPEAR -- This will be returned when you are requested
    to close down the GUI of the commodity.

Any return code other than these is the  ID of  a  CXM_IEVENT message.
This normally is the ID of an attached hotkey.

  EXAMPLE:
Object    *com_obj;
ULONG    mask = 0, type, id, data, rc;
BOOL     running = TRUE;

GetAttr( COMM_SigMask, com_obj, &mask );

do {
  Wait( mask );
  while (( rc = DoMethod( com_obj,
        CM_MSGINFO,
        &type,
        &id,
        &data )) != CMMI_NOMORE ) {
    switch ( type ) {
      ...
    }
  }
} while ( running );

/* Or (V40 style) */

do {
  Wait( mask );
  while (( rc = DoMethod( com_obj,
        CM_MSGINFO,
        NULL,
        NULL,
        NULL )) != CMMI_NOMORE ) {
    switch ( rc ) {
      ...
    }
  }
} while ( running );

  SEE ALSO
CM_ADDHOTKEY
```

1.9 commodityclass/CM_REMHOTKEY

```
  NAME
CM_REMHOTKEY -- Remove a hotkey from the broker.

  SYNOPSIS
err = DoMethod( obj, CM_REMHOTKEY, id );

ULONG   err;
ULONG   id;

  FUNCTION
To remove a hotkey previously added with CM_ADDHOTKEY from the broker.
It is not necessary to disable the key or the broker to do this.  This
method will take care of this for you.

  INPUTS
id  - The ID of the key to remove.

  RESULT
err - True for succes, FALSE for failure.

  EXAMPLE
Object    *broker;
#define         ID_SF1    1

/*
 *  Remove the hotkey with this
 *  ID from the broker.
 */
DoMethod( broker, CM_REMHOTKEY, ID_SF1 );

  SEE ALSO
CM_ADDHOTKEY
```

1.10 commodityclass/COMM_Description

```
  NAME
COMM_Description -- ( STRPTR )

  FUNCTION
To specify a short description of the commodity functionality.

  DEFAULT
NULL.

  APPLICABILITY
(I).
```

1.11 commodityclass/COMM_ErrorCode

```
  NAME
COMM_ErrorCode -- ( ULONG )
```

```
  FUNCTION
To find out exactly what went wrong when adding a hotkey to the broker
failed. Getting this attribute can result in the following error
codes:

CMERR_OK -- OK. No problems.
CMERR_NO_MEMORY -- Out of memory.
CMERR_KEYID_IN_USE -- Key ID was already used.
CMERR_KEY_CREATION -- Key creation failed.
CMERR_CXOBJERROR -- CxObjError() reported failure.

  APPLICABILITY
(G).

  SEE ALSO
CM_ADDHOTKEY
```

1.12 commodityclass/COMM_Name

```
  NAME
COMM_Name -- ( STRPTR )

  FUNCTION
To specify the name the commodity will have.

  DEFAULT
NULL.

  APPLICABILITY
(I).
```

1.13 commodityclass/COMM_Notify

```
  NAME
COMM_Notify -- ( BOOL )

  FUNCTION
When set to TRUE your broker will be notified when it has been
attempted to run a commodity with the same name.

  DEFAULT
TRUE.

  APPLICABILITY
(I).

  SEE ALSO
COMM_Unique
```

1.14 commodityclass/COMM_Priority

```
  NAME
COMM_Priority -- ( BYTE )

  FUNCTION
To specify the priority of the commodity (not the task priority, but
the value of the CX_PRIORITY tooltype)

  DEFAULT
0.

  APPLICABILITY
(I).
```

1.15 commodityclass/COMM_ShowHide

```
  NAME
COMM_ShowHide -- ( BOOL )

  FUNCTION
When set to TRUE your broker will get notified when it is supposed  to
open or close it's window(s).

  DEFAULT
FALSE.

  APPLICABILITY
(I).
```

1.16 commodityclass/COMM_SigMask

```
  NAME
COMM_SigMask -- ( ULONG )

  FUNCTION
To obtain the broker signal  mask  which  can  be  used  to  wait  for
commodity events to occur.

  EXAMPLE

Object         *CO_Com;
ULONG   mask, sigrec;

GetAttr( COMM_SigMask, CO_Com, &mask );

do {
  sigrec = Wait( mask );
  ...
} while ( ... );

  APPLICABILITY
```

(G).

1.17 commodityclass/COMM_Title

NAME
COMM_Title -- (STRPTR)

FUNCTION
To specify the title of the commodity.

DEFAULT
NULL.

APPLICABILITY
(I).

1.18 commodityclass/COMM_Unique

NAME
COMM_Unique -- (BOOL)

FUNCTION
When set to TRUE it will not be possible to run commodities with the same name as this one.

DEFAULT
TRUE.

APPLICABILITY
(I).

SEE ALSO
COMM_Name, COMM_Notify