honduras
DSM-CC Download Client
API Reference Manual

Bionic Buffalo Corporation
2003.08.13

Contents

Document Information and Copyright

Document Information and Copyright

Document Context

This document is a portion of the documentation for Bionic Buffalo's honduras product. Information about the product and other, related documentation can be found at *http://www.tatanka.com/prod/honduras.html*

Bionic Buffalo offers a family of products implementing DSM-CC and related protocols. For more information, see
http://www.tatanka.com/prod/dsmcc.html

Copyright, License, and Trademarks

Author and Publisher

Bionic Buffalo Corporation
502 North Division Street
Carson City, Nevada 89703
USA

telephone +1 775 882 1842
fax +1 775 882 6047
e-mail *query@tatanka.com*
web *http://www.tatanka.com*

PGP/GnuPG key fingerprint:

```
a836 e7b0 24ad 3259 7c38 b384 8804 5520 2c74 1e5a
```

Document History

Project name: *honduras*
File name: *hnd_api_reference*
Formal revision date: Wednesday 2003.08.13
Last modified: 2003-09-04-07:57

For the latest version of this document, or for other forms of this document, see
http://www.tatanka.com/doc/man/honduras/index.html

Updates and revisions of this document are announced on Bionic Buffalo's DSM-CC e-mail announcement list. For more information, see
http://www.tatanka.com/bbc/lists.html

Introduction

Introduction

Bionic Buffalo's honduras product implements DSM-CC download clients as specified in *Extensions for DSM-CC* (ISO/IEC 13818-6), one of the MPEG-2 family of specifications.

The honduras product includes a library of software which can be used to implement download client applications. It also includes a client application (dnlmount) which can be used without additional programming to mount download server contents onto the client's directory, so they appear as if they were local files.

This document describes the API available with the honduras library. For an overview of the entire product, refer to *DSM-CC Download Client Overview*. For information on dnlmount, refer to *DSM-CC Download dnlmount User's Guide*.

Before reading the honduras manuals, it is suggested that you first read the *DSM-CC Download Common Introduction*, which is part of the angola product documentation. That document provides an overview of the download protocol and how it is used in applications.

Using the Library

The library is provided in shrouded C source code (“portable object code”) format, with a Makefile and documentation. Instructions for installation are included with the distribution.

Library routines may be called as functions from a C program, or from a program in other languages which support the same calling sequences.

An application using the honduras library calls hnd_install_download_client_mgr(), which returns a reference to an object of class Honduras::ClientManager. This ClientManager object is then used to create one or more Guatemala::Client objects.

All of the Client objects created from a single ClientManager object share the same i/o addresses. Otherwise, they operate independently. If necessary, an application may create multiple ClientManager objects to allow operation of clients on multiple addresses.

The `ClientManager` and `Client` objects in a running application can be called in two ways:

- the application can call them directly, as subroutines
- another application (in the same machine, or over the network) can call them using the CORBA IIOP (the internet inter-ORB protocol)

The second method allows complete control of the download client from a remote location.

When the application requires asynchronous notification of certain events, it must create an object of class `Honduras::Application` which is registered with the `ClientManager`. By using an object for this purpose (instead of a callback address), the `ClientManager` can notify remote applications.

Interface and Operation Descriptions

The rest of this document consists of descriptions of the interfaces and operations available from the honduras library. The format is traditional. Because of the two ways of accessing `ClientManager` and `Client` objects, the API descriptions in this document specify the IDL interface as well as the C interface to each class.

`hnd_install_download_client_mgr`

Name

`hnd_install_download_client_mgr` -- create an object of class `Honduras::ClientManager`

Synopsis - C

```
#include <honduras.h>

void    hnd_install_download_client_mgr
                ( hnd_client_parameters_T     * parameters,
                Honduras_ClientManager        * client_manager,
                CORBA_Environment             * ev ) ;
```

Arguments

`parameters` -- *(in/out)* configuration parameters
`server_manager` -- *(out)* created `Honduras::ClientManager` object
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This function creates an object of class `Honduras::ClientManager` using the specified **`parameters`**.

Notes

1. This is the first `honduras` function called by an application.

Example

Availability

honduras (Bionic Buffalo)

See also

hnd_client_parameters_T

Reference

`hnd_client_parameters_T`

Name

`hnd_client_parameters_T` -- structure used to specify parameters for the installation of a download client manager

Synopsis - C

```
#include <honduras.h>

typedef struct hnd_client_parameters_T
        {
        CORBA_ORB                       orb ;
        PortableServer_POA              poa ;
        CosNaming_NamingContext         naming_context ;
        unsigned char                 * client_manager_name ;
        mar_netlink_T                   channels [ 4 ] ;
        int                             sockets [ 4 ] ;
        size_t                          maximum_message_sizes [ 4 ] ;
        struct timespec                 timer_granularity ;
        unsigned long                   open_file_limit ;
        }
        hnd_client_parameters_T ;
```

Description

The `hnd_client_parameters_T` structure us used to pass arguments to, and return results from, an invocation of the `hnd_install_download_client_mgr ()` operation.

Notes

1. The **`orb`** and **`poa`** are used to create and register the `Honduras::ClientManager` object.

2. If a **`naming_context`** and **`server_manager`**_name are defined, then the new `ClientManager` will be named accordingly in the designated context.

3. The subscript for **`channels`**, **`sockets`**, and **`maximum_message_sizes`** is one of the values of `Angola::Channel`. There may be one physical channel, but each of the four logical channel must be described separately. When it is necessary for the library to determine if two logical channels are mapped to the same physical channel, the software compares the values of the corresponding **`channels`** elements with each other.

4. If non-zero values are specified for **sockets**, then the sockets will be presumed open and configured. Otherwise, sockets will be created and configured according to **channels**, and returned to the caller for (optional) additional configuration. Regardless of whether or not the **sockets** are pre-opened, the **channels** must be specified.

5. The **maximum_message_sizes** represents the largest messages which might be sent or received on each of the four channels. It may be left zero for a default value.

6. **timer_granularity** may be left zero for a default value.

7. **open_file_limit** is the greatest number of files which may be open simultaneously. It may be left zero for a default value.

Example

Availability

honduras (Bionic Buffalo)

See also

hnd_install_download_client_mgr

Reference

`Honduras::Application`

Name

`Honduras::Application` -- interface used by object belonging to an application

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
        {
        interface               Application ;
        } ;
```

Synopsis - C

```
#include <honduras.h>

typedef CORBA_Object            Honduras_Application ;
```

Description

An application creates a `Honduras::Application` object when it wants to be notified asynchronously of events in the download client. The application registers the object by modifying the `Honduras::ClientManager::application` attribute.

Notes

1. The `Application` interface has operations `complete` and `response_indication`.

Example

Availability

`honduras` (Bionic Buffalo)

See also

`Honduras::Application::complete`
`Honduras::Application::response_indication`

Honduras::ClientManager::application

Reference

`Honduras::Application::complete`

Name

`Honduras::Application::complete` -- notifies an application that an operation is complete or was cancelled

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Application
        {
        void    complete
                    ( in Client                     client,
                    in unsigned long                cookie,
                    in Angola::CancelReason         reason,
                    in Angola::Statistics           statistics,
                    in Angola::ModuleSynopsisList   modules ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

void    Honduras_Application_complete
                ( Honduras_Application          o,
                Honduras_Client                 client,
                CORBA_unsigned_long             cookie,
                Angola_CancelReason             reason,
                Angola_Statistics             * statistics,
                Angola_ModuleSynopsisList     * modules,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the application to be notified
`client` -- *(in)* the `Client` conducting the operation
`cookie` -- *(in)* the cookie passed to the `gather_modules` or `gather_image` operation
`reason` -- *(in)* if zero, indicates that the operation completed normally; if nonzero, the reason for the operation cancellation

statistics -- *(in)* summary of operation activity
modules -- *(in)* description of modules sought in attempt
ev -- *(in/out)* CORBA environment

Returns

void

Exceptions

Description

This operation notifies an application that a download operation has completed, or was cancelled. It is sent as the ultimate result of a gather_image or gather_modules operation, to notify the application that the asynchronous activity has terminated.

Notes

1. The **cookie** was passed to **client**, in a previous gather_image or gather_modules operation.

Example

Availability

honduras (Bionic Buffalo)

See also

Honduras::Application
Honduras::Client::gather_image
Honduras::Client::gather_modules

Reference

`Honduras::Application::response_indication`

Name

`Honduras::Application::response_indication` -- notifies an application that a `DownloadInfoIndication` or `DownloadInfoResponse` message was received

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Application
        {
        void    response_indication
                    ( in Client                     client,
                    in DSM::CompatibilityDescriptor compat_descr,
                    in CORBA::OctetSeq              private_data,
                    in Angola::ModuleSynopsisList   modules,
                    out Angola::CancelReason        reason ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

void    Honduras_Application_response_indication
                ( Honduras_Application          o,
                Honduras_Client                 client,
                DSM_CompatibilityDescriptor   * compat_descr,
                CORBA_sequence_octet          * private_data,
                Angola_ModuleSynopsisList     * modules,
                Angola_CancelReason           * reason,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the application to be notified
`client` -- *(in)* the `Client` which received the message
`compat_descr` -- *(in)* the message's compatibility descriptor
`private_data` -- *(in)* the message's private data
`modules` -- *(in)* description of modules advertised in the message

reason -- *(out)* if nonzero, a reason to cancel the operation
ev -- *(in/out)* CORBA environment

Returns

void

Exceptions

Description

This operation notifies an application that a DownloadInfoResponse or DownloadInfoIndication message was received by **client**. The application may decide (based on the **compat_descr** and **private_data**) if the server image or modules are appropriate, and may take appropriate action, or may (optionally) cancel the operation.

Notes

Example

Availability

honduras (Bionic Buffalo)

See also

Honduras::Application

Reference

`Honduras::Client`

Name

`Honduras::Client` -- class of objects implementing a Download Client

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
        {
        interface               Client ;
        } ;
```

Synopsis - C

```
#include <honduras.h>

typedef CORBA_Object            Honduras_Client ;
```

Description

A `Honduras::Client` object implements the Download Client defined by the DSM-CC specification. Each `Client` object has a separate state, and implements a separate state machine.

Notes

1. `Client` objects are created by `ClientManager` objects, using the `create_client` operation.

2. The `Client` interface has the `session_parameters` attribute. Operations are `activate`, `deactivate`, `fetch_module`, `get_directory`, and `destroy`.

Example

Availability

`honduras` (Bionic Buffalo)

See also

Honduras::ClientManager::create_client
Honduras::Client::activate
Honduras::Client::deactivate
Honduras::Client::destroy
Honduras::Client::fetch_module
Honduras::Client::get_directory
Honduras::Client::session_parameters

Reference

`Honduras::Client::activate`

Name

`Honduras::Client::activate` -- start a download client

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
        {
        interface Client
                {
                void    activate ( ) ;
                } ;
        } ;
```

Synopsis - C

```
#include <honduras.h>

void    Honduras_Client_activate
                ( Honduras_Server               o,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client to be activated
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation activates a `Honduras::Client`.

Notes

1. Before invoking this operation, an application should set the `Client::session_parameters` attribute.

2. Once a `Client` is activated, its initial action will depend on the `scenario` specified in the `session_parameters`. In the flow-controlled scenario, the `Client` will initiate operation by sending a `DownloadInfoRequest` message to the server. In the non-flow-controlled and carousel scenarios, the `Client` will listen for incoming `DownloadInfoResponse` or `DownloadInfoIndication` messages from the server.

Example

Availability

`honduras` (Bionic Buffalo)

See also

`Honduras::Client::session_parameters`

Reference

`Honduras::Client::compatibility_descriptor`

Name

`Honduras::Client::compatibility_descriptor` -- the `DSM::CompatibilityDescriptor` used in `DownloadInfoRequest` messages

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Client
        {
        attribute DSM::CompatibilityDescriptor  compatibility_descriptor ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

DSM_CompatibilityDescriptor
      * Honduras_Client__get_compatibility_descriptor
                ( Honduras_Client               o,
                CORBA_Environment             * ev ) ;
void    Honduras_Client__set_compatibility_descriptor
                ( Honduras_Client               o,
                DSM_CompatibilityDescriptor   * compatibility_descriptor,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client to which the attribute applies
`compatibility_descriptor` -- *(`set` only) (in)* new or initial value for the compatibility descriptor
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) current value of the compatibility descriptor

Exceptions

Description

This attribute specifies the compatibility descriptor sent in `DownloadInfoRequest` messages.

Notes

1. This is *not* the compatibility descriptor received from servers in `DownloadInfoResponse` and `DownloadInfoIndication` messages. That value is passed to the `Application` by means of the `Application::response_indication` operation.

Example

Availability

`honduras` (Bionic Buffalo)

See also

Reference

`Honduras::Client::deactivate`

Name

`Honduras::Client::deactivate` -- terminate a download client

Synopsis - IDL

```
#include <guatemala.idl>

module Honduras
        {
        interface Client
                {
                void    deactivate ( ) ;
                } ;
        } ;
```

Synopsis - C

```
#include <honduras.h>

void    Honduras_Client_deactivate
                ( Honduras_Client               o,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client to be deactivated
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation terminates the operation of a `Honduras::Client`. The `Client` may be reactivated again. The `Client` remains in existence until it is destroyed.

Notes

Example

Availability

honduras (Bionic Buffalo)

See also

Honduras::Client::activate
Honduras::Client::destroy

Reference

`Honduras::Client::destroy`

Name

`Honduras::Client::destroy` -- destroy a download client

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
        {
        interface Client
                {
                void    destroy ( ) ;
                } ;
        } ;
```

Synopsis - C

```
#include <honduras.h>

void    Honduras_Client_destroy
                ( Honduras_Client               o,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client to be destroyed
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation destroys a `Honduras::Client` object. References to the `Client` are no longer valid.

Notes

Example

Availability

honduras (Bionic Buffalo)

See also

Reference

`Honduras::Client::fetch_module`

Name

`Honduras::Client::fetch_module` -- download a module into memory

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Client
        {
        unsigned short fetch_module
            ( in unsigned short             id,
            out CORBA::OctetSeq             data ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

CORBA_unsigned_short
        Honduras_Client_fetch_module
                ( Honduras_Client               o,
                CORBA_unsigned_short            id,
                CORBA_OctetSeq                ** data,
                CORBA_Environment              * ev ) ;
```

Arguments

`o` -- *(in)* the client which is to fetch the designated module
`id` -- *(in)* the moduleId of the module to be fetched
`data` -- *(out)* the module contents
`ev` -- *(in/out)* CORBA environment

Returns

the status value describing the outcome of the request

Exceptions

Description

This operation downloads a fresh copy of the module with `moduleId` value **`id`**, and returns a memory buffer **`data`** with the contents of the module.

Notes

1. The specified module is loaded directly into the memory buffer. It is not cached or assembled on disk.

Example

Availability

`honduras` (Bionic Buffalo)

See also

Reference

`Honduras::Client::gather_image`

Name

`Honduras::Client::gather_image` -- asynchronously download an image from the server

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Client
        {
        unsigned short gather_image
            ( in Angola::ModuleSynopsisList     modules,
            in string                           base_directory,
            in unsigned long                    cookie ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

CORBA_unsigned_short
        Honduras_Client_gather_image
                ( Honduras_Client               o,
                Angola_ModuleSynopsisList     * modules,
                CORBA_string                    base_directory,
                CORBA_unsigned_long             cookie,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client which is to download the image
`modules` -- *(in)* the modules of the image
`base_directory` -- *(in)* for modules with a reference address given as a path name, the base directory to be used in specifying the complete path
`cookie` -- *(in)* a value to be returned to the user in an Application::complete operation
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation causes the **`client`** to initiate an asynchronous download of an image from the server in a flow-controlled or non-flow-controlled scenario. When the entire image is downloaded, or perhaps when the operation is cancelled or fails, the **`client`** will call the application's `complete` operation to notify the application of the termination.

Notes

1. The `reference` for each module in the **`modules`** list specifies its destination (whether it is to be downloaded to memory, file, or object.) Modules do not need to use a uniform type of destination. (For example, some modules may be loaded into memory, and some into files.)

2. The `reference` values of **`modules`** are relative to the location of the **`client`** object, not to the location of the application.

3. For a module, if `reference . _d` is `Tibet_DATASET_REF_PATH_NAME`, then the module's contents wll be loaded into the file specified by **`base_directory`** and `reference . _u . path_name`. If the file already exists, its contents will be destroyed.

4. For a module, if `reference . _d` is `Tibet_DATASET_REF_OCTET_SEQ`, then the module's contents will be loaded into a memory buffer in the caller's address space. In such a case, r`eference . _u . octet_sequence` will not be used, and should indicate an empty buffer or `NULL` address. The address of the new buffer which will contain the data will be returned to the application using the `Application::complete` operation's `modules` parameter.

5. `Application::complete` will be called when the entire image is transferred (or the operation fails or is cancelled), not for each module separately. **`cookie`** will be passed as an argument, allowing the application to correlate that invocation with this invocation.

6. This operation is inappropriate for carousel scenarios.The `gather_modules` operation should be used for carousels.

Example

Availability

honduras (Bionic Buffalo)

See also

Reference

`Honduras::Client::gather_modules`

Name

`Honduras::Client::gather_modules` -- asynchronously download one or more modules from a carousel

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Client
        {
        unsigned short gather_modules
            ( in Angola::ModuleSynopsisList     modules,
            in string                           base_directory,
            in unsigned long                    cookie ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

CORBA_unsigned_short
        Honduras_Client_gather_modules
                ( Honduras_Client               o,
                Angola_ModuleSynopsisList     * modules,
                CORBA_string                    base_directory,
                CORBA_unsigned_long             cookie,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client which is to download the image
`modules` -- *(in)* the modules of the image
`base_directory` -- *(in)* for modules with a reference address given as a path name, the base directory to be used in specifying the complete path
`cookie` -- *(in)* a value to be returned to the user in an Application::complete operation
`ev` -- *(in/out)* CORBA environment

Returns

`void`

Exceptions

Description

This operation causes the **`client`** to initiate an asynchronous download of one or more modules from the server in a carousel scenario. When each module is downloaded, or perhaps when the operation is cancelled or fails, the **`client`** will call the application's `complete` operation to notify the application of the termination.

Notes

1. The `reference` for each module in the **`modules`** list specifies its destination (whether it is to be downloaded to memory, file, or object.) Modules do not need to use a uniform type of destination. (For example, some modules may be loaded into memory, and some into files.)

2. The `reference` values of **`modules`** are relative to the location of the **`client`** object, not to the location of the application.

3. For a module, if `reference . _d` is `Tibet_DATASET_REF_PATH_NAME`, then the module's contents wll be loaded into the file specified by **`base_directory`** and `reference . _u . path_name`. If the file already exists, its contents will be destroyed.

4. For a module, if `reference . _d` is `Tibet_DATASET_REF_OCTET_SEQ`, then the module's contents will be loaded into a memory buffer in the caller's address space. In such a case, r`eference . _u . octet_sequence` will not be used, and should indicate an empty buffer or `NULL` address. The address of the new buffer which will contain the data will be returned to the application using the `Application::complete` operation's `modules` parameter.

5. `Application::complete` will be called when each module is transferred (or the operation fails or is cancelled). There will be as many calls to `complete` as there are modules in the list. **`cookie`** will be passed as an argument, allowing the application to correlate that invocation with this invocation. The application must distinguish among calls to `complete` by examing `complete`'s `modules` argument.

6. This operation is inappropriate for flow-controlled and non-flow-controlled download

scenarios.The `gather_image` operation should be used for those scenarios.

Example

Availability

`honduras` (Bionic Buffalo)

See also

Reference

`Honduras::Client::get_directory`

Name

`Honduras::Client::get_directory` -- return a copy of the download directory

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Client
        {
        unsigned short get_directory
            ( out Angola::ModuleSynopsisList    directory ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

CORBA_unsigned_short
        Honduras_Client_get_directory
                ( Honduras_Client               o,
                Angola_ModuleSynopsisList    ** directory,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client which is to return the directory
`directory` -- *(out)* the module contents
`ev` -- *(in/out)* CORBA environment

Returns

the status value describing the outcome of the request

Exceptions

Description

This operation returns a copy of the current download directory. The directory contents indicate which modules are available for download.

Notes

1. The cached copy of the directory is returned.

Example

Availability

honduras (Bionic Buffalo)

See also

Reference

`Honduras::Client::private_data`

Name

`Honduras::Client::private_data` -- the private data used in `DownloadInfoRequest` messages

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Client
        {
        attribute CORBA::OctetSeq           private_data ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

CORBA::OctetSeq
      * Honduras_Client__get_private_data
                ( Honduras_Client               o,
                CORBA_Environment             * ev ) ;
void    Honduras_Client__set_private_data
                ( Honduras_Client               o,
                CORBA_OctetSeq                * private_data,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the client to which the attribute applies
`private_data` -- *(`set` only) (in)* new or initial value for the private data
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) current value of the private data

Exceptions

Description

This attribute specifies the private data sent in `DownloadInfoRequest` messages.

Notes

1. This is *not* the private data received from servers in `DownloadInfoResponse` and `DownloadInfoIndication` messages. That value is passed to the `Application` by means of the `Application::response_indication` operation.

Example

Availability

`honduras` (Bionic Buffalo)

See also

Reference

`Honduras::Client::session_parameters`

Name

`Honduras::Client::session_parameters` -- parameters for a download session

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface Client
        {
        attribute Angola::SessionParameters session_parameters ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

Angola_SessionParameters
        Honduras_Client__get_session_parameters
                ( Honduras_Client           o,
                CORBA_Environment         * ev ) ;
void    Honduras_Client__set_session_parameters
                ( Honduras_Client           o,
                Angola_SessionParameters  * session_parameters,
                CORBA_Environment         * ev ) ;
```

Arguments

`o` -- *(in)* the client which the attribute applies
`session_parameters` -- *(`set` only) (in)* new or initial parameters for the download session
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) current download session parameters

Exceptions

Description

This attribute specifies the scenario (flow-controlled download, non-flow-controlled download, or carousel), window size, block size, and timing parameters for the Session.

Notes

1. Not all size and timing parameters are used for all scenarios. Applicability is indicated in the following table:

Parameter	*Flow controlled download*	*Non flow controlled download*	*Carousel*
`blockSize`	yes	yes	yes
`windowSize`	yes	no	no
`ackPeriod`	yes	no	no
`tCDownloadWindow`	yes	no	no

2. In the flow-controlled scenario, the parameters specified for the server may not be the parameters used in the session. The server negotiates a set of session parameters based on its own limits, and on the limits specified by the client in the client's `DownloadInfoRequest` message.

Example

Availability

`honduras` (Bionic Buffalo)

See also

Reference

`Honduras::ClientManager`

Name

`Honduras::ClientManager` -- class of objects which can create `Client` objects

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
        {
        interface               ClientManager ;
        } ;
```

Synopsis - C

```
#include <honduras.h>

typedef CORBA_Object            Honduras_ClientManager ;
```

Description

A `Honduras::ClientManager` creates `Honduras::Client` objects. The `Client` objects created by a `ClientManager` share the same i/o addresses. The `ClientManager` interface includes the `application` attribute, and the `create_client` operation.

Notes

1. A `ServerManager` object is returned by the function `hnd_install_download_client_mgr`.

2. Each `ClientManager` object may be associated with zero or one `Honduras::Application` objects. The association is specified by the `ClientManager::application` attribute.

Example

Availability

`honduras` (Bionic Buffalo)

See also

hnd_install_download_client_mgr
Honduras::ClientManager::application
Honduras::ClientManager::create_client

Reference

`Honduras::ClientManager::application`

Name

`Honduras::ClientManager::application` -- the `Honduras::Application` object associated with a `ClientManager`

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface ClientManager
        {
        attribute Application    application ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

Honduras_Application
        Honduras_ClientManager__get_application
                ( Honduras_ClientManager           o,
                CORBA_Environment                * ev ) ;
void    Honduras_ClientManager__set_application
                ( Honduras_ClientManager           o,
                Honduras_Application               application,
                CORBA_Environment                * ev ) ;
```

Arguments

`o` -- *(in)* the `ClientManager` with which the `Application` is associated
`application` -- *(`set` only) (in)* the `Application` to associate with the `ClientManager`
`ev` -- *(in/out)* CORBA environment

Returns

(`get` only) the `Application` associated with the `ClientManager`

Exceptions

Description

This attribute specifies the `Honduras::Application` associated with the `ClientManager`.

Notes

1. The `Application` object is created by the application which uses the `honduras` library, `ClientManager`, or `Client` objects.

Example

Availability

`honduras` (Bionic Buffalo)

See also

`Honduras::Application`

Reference

`Honduras::ClientManager::create_client`

Name

`Honduras::ClientManager::create_client` -- create a download client

Synopsis - IDL

```
#include <honduras.idl>

module Honduras
    {
    interface ClientManager
        {
        Client   create_client ( ) ;
        } ;
    } ;
```

Synopsis - C

```
#include <honduras.h>

Honduras_Client
....    Honduras_ClientManager_create_client
                ( Honduras_ClientManager        o,
                CORBA_Environment             * ev ) ;
```

Arguments

`o` -- *(in)* the `ClientManager` to create the `Client`
`ev` -- *(in/out)* CORBA environment

Returns

the new `Client` object

Exceptions

Description

This operation creates a new `Honduras::Client` object.

Notes

Example

Availability

honduras (Bionic Buffalo)

See also

Honduras::Client

Reference