honduras
DSM-CC Download Client Introduction

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_introduction*
Formal revision date: Wednesday 2003.08.13
Last modified: 2003-09-04-07:55

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

Overview

Overview

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 is an overview of the honduras product and its use. For a description of the library API, refer to *DSM-CC Download Client API Reference Manual*. 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.

Interfaces and Operations

Interfaces and Operations

guatemala defines three interfaces (object classes): Honduras::Application, Honduras::ClientManager, and Honduras::Client. This chapter describes each interface and its operations.

Honduras::Application

A Honduras::Application object is created by the application using the honduras library. It is registered with the ClientManager, and its operations provide a way for ClientManager and Client objects to call the application asynchronously. In traditional terms, it is a call-back mechanism.

The following operations are defined on the Application interface:

> **complete ()** -- informs the application that a scenario has completed, normally or otherwise, giving the reason for the termination
>
> **response_indication ()** -- informs the application that the server has sent a DownloadInfoResponse nessage, or a DownloadInfoIndication message, allowing the application to examine the compatibility descriptor and private data, to decide if the information is applicable to it, and to take other appropriate action

---

Honduras::ClientManager

A Honduras::ClientManager object creates Client objects, and handles their network connections. (All Client objects created by a single ClientManager must have the same network addresses.) The ClientManager also deals with some attributes and parameters which are common to the Clients under its control.

The following operations and attributes are defined on the ClientManager interface:

application -- this attribute notifies the ClientManager regarding the identity of the Application object

create_client () -- instantiates a Client object

Honduras::Client

A Honduras::Client object is create by a ClientManager. Each Client represents an instance of a "Download Client" as defined by the specification, and includes a state machine distinct from that of other Client objects.

The following operations and attributes are defined on the Client interface:

activate () -- enables operation of the Client

compatibility_descriptor -- the default compatibility descriptor, as sent in the DownloadInfoRequest message

deactivate () -- disables operation of the Client

destroy () -- destroys a Client

fetch_module () -- synchronously download one module

gather_image () -- asychronously download image from server

gather_modules () -- asynchronously download selected modules

get_directory () -- return the most recent directory from the carousel server

private_data -- the value of the private data portion of the DownloadInfoRequest message

session_parameters -- defines operating parameters for the session

Example Procedure: Simple Data Carousel Download

Example Procedure: Simple Data Carousel Download

This procedure downloads a single module from a server using the data carousel scenario. It has limitations, but it requires a minimal amount of coding to implement. It may be useful to test connections or other aspects of an application or system.

Limitations

This procedure has limitations. The most significant are:

- Neither compatibility descriptors, nor private data, are supported. There is no option to receive varying content depending on client requirements.
- Configurability is limited. However, additional operations can be added to the basic sequence of operations to adjust the configuration.

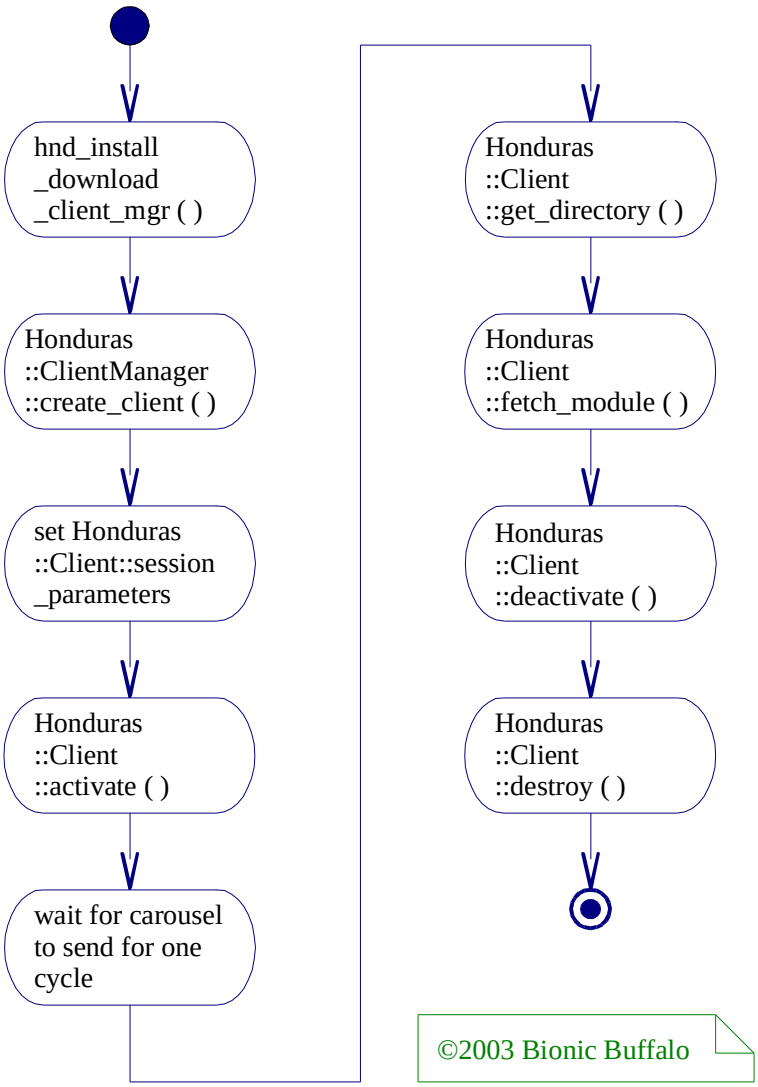
Description

1. Install `honduras` using `hnd_install_download_client_mgr()`. This returns a reference to the `ClientManager` object.
2. Use the `ClientManager` to create a `Client`.
3. Set the `Client::session_parameters`.
4. Activate the client. The Client will listen for messages, but will only save `DownloadInfoIndication` messages from the server. (It will not, by default, save any carousel data.)
5. Wait for the carousel to make at least one revolution, so the directory (in the server's `DownloadInfoIndication` message) has been sent. Use `Client::get_directory` to retrieve the directory of modules available in the download session.

6. Select a module from the directory. Use `Client::fetch_module` to grab a copy of the module the next time it is sent. (Repeat this step to fetch additional modules.)

7. Deactivate, then destroy, the `Client`.

Sequence of Operations



Example Procedure: Dynamically-Selected Flow-Controlled Download

Example Procedure: Dynamically-Selected Flow-Controlled Download

This procedure demonstrates the capability of the client to request specific content from the server, and to verify that the server's offering is appropriate, based on compatibility descriptors and private data.

Description

1. Install `honduras` using `hnd_install_download_client_mgr()`. This returns a reference to the `ClientManager` object.

2. Use the `ClientManager` to create a `Client`. Create an `Application` object using standard CORBA mechanisms. Register the `Application` with the `ClientManager`.

3. Set the `Client::session_parameters`

4. Set the `Client::compatibility_descriptor` and `Client::private_data` attributes to reflect the `Client`'s configuration and requirements

5. Activate the `Client`. The `Client` will send a `DownloadInfoRequest` to the `Server`.

6. When a `Client` receives a `DownloadInfoResponse`, then the `Client` will invoke the `Application::response_indication` operation, letting the `Application` know the contents of the request's private data and compatibility descriptor.

7. Based on the details of the `DownloadInfoResponse` (as reported by the `response_indication` operation), the `Application` decides if the server's contents are appropriate, and indicates by return value if the `Client` ought to continue with the download operation.

8. When the `Application` returns from the `response_indication` operation with an affirmative response to the question of continuing, the `Client` prepares for the download operation (by allocating necessary buffers, *et caetera*) and then sends the initial

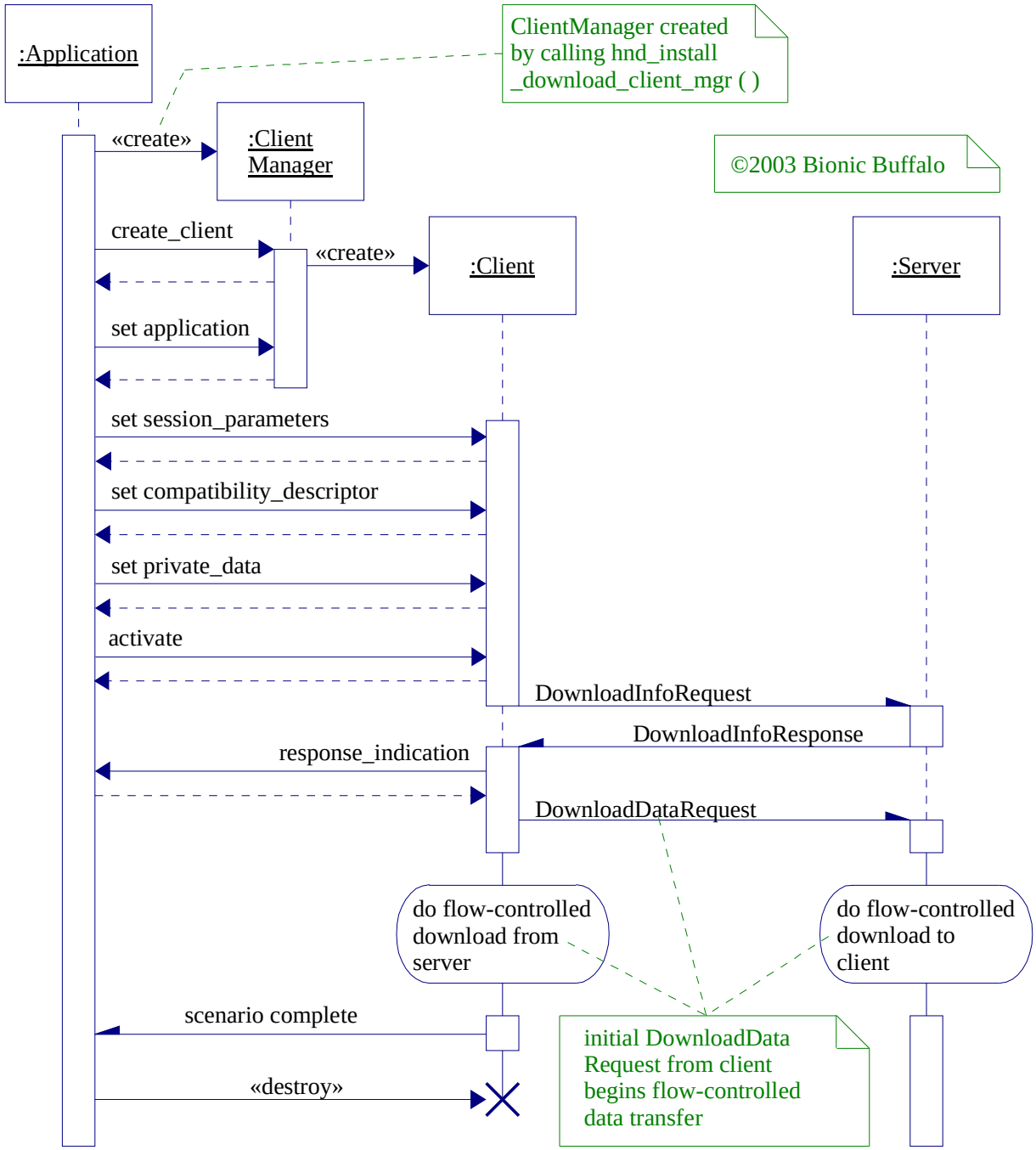
`DownloadDataRequest` message.

9. The `Server` downloads the content to the `Client`.

10. Using the `complete` operation, the `Client` notifies the `Application` that the scenario is finished.

11. The `Application` destroys the `Client`.

Sequence of Operations



Example Procedure: Asynchronous Carousel Download

Example Procedure: Asynchronous Carousel Download

This procedure explains the steps needed to download, asynchronously, modules from a carousel. In this example, there are multiple DownloadInfoIndication messages sent by the carousel, corresponding to multiple content sets. The client begins downloading one or more modules after being notified of their availability.

Description

1. Install `honduras` using `hnd_install_download_client_mgr()`. This returns a reference to the `ClientManager` object.

2. Use the `ClientManager` to create a `Client`. Create an `Application` object using standard CORBA mechanisms. Register the `Application` with the `ClientManager`.

3. The `Application` sets the `Client::session_parameters`

4. Activate the `Client`. The `Client` will wait for `DownloadInfoIndication` messages from the `Server`.

5. When a `Client` receives a `DownloadInfoIndication`, then the `Client` will invoke the `Application::response_indication` operation, letting the `Application` know the contents of the request's private data and compatibility descriptor.

6. Based on the details of the `DownloadInfoResponse` (as reported by the `response_indication` operation), the `Application` decides if the server's contents are appropriate for the `Application`. If they are not appropriate, then the `Application` returns to the previous step, waiting for another `response_indication` invocation from the `Client`. If they are appropriate, then the `Client` advances to the next step.

7. The `Application` decides which modules it wants to download, then it invokes `Client::gather_modules` to cause the `Client` to accumulate the blocks belonging to the chosen modules.

8. The `Client` downloads the requested modules (by listening to the carousel).

9. Using the `complete` operation, the `Client` notifies the `Application` that the requested modules are available.

10. The `Application` destroys the `Client`.

Sequence of Operations