angola
DSM-CC Download Common Introduction

Bionic Buffalo Corporation
2003.09.23

Contents

Document Information and Copyright

Document Information and Copyright

Document Context

This document is a portion of the documentation for Bionic Buffalo's angola product. Information about the product and other, related documentation can be found at *http://www.tatanka.com/prod/angola.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: *angola*
File name: *ago_introduction*
Formal revision date: Tuesday 2003.09.23
Last modified: 2003-09-24-14:15

For the latest version of this document, or for other forms of this document, see
http://www.tatanka.com/doc/man/angola/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 guatemala, honduras, and angola products implement the DSM-CC download protocol as specified in *Extensions for DSM-CC* (ISO/IEC 13818-6), one of the MPEG-2 family of specifications. The server is implemented by guatemala, and the client by honduras. The angola library is used by both for certain commmon functions, and is is available only as an adjunct to the other two products.

This document is an introduction to both server and client aspects of the download protocol and its use. This is the place to start when learning about download.

Specifications and Background

Specifications and Background

This chapter describes the formal specification of DSM-CC, and explains a little of where and how it is used.

Download and Its Relationship to DSM-CC

DSM-CC (an acronym for Digital Storage Media Command and Control) is defined by ISO/IEC 13818-6, one of a family of related specifications. These specifications are commonly known as the MPEG-2 specifications, and include:

- ISO/IEC 13818-1: Systems. Specifies the format of content streams, and how they are multiplexed and synchronized. Specifies the transport of content over unreliable channels. These formats, with variations, are used for satellite, cable TV, DVD, and HDTV. (The older, corresponding MPEG-1 specification was ISO/IEC 11172-1.)

- ISO/IEC 13818-2: MPEG-2 Video Compression. Compression of video, both interlaced and non-interlaced. A superset of the older MPEG-1 video compression standard (ISO/IEC 11172-2).

- ISO/IEC 13818-3. MPEG-2 Audio Compression. Compression of audio. An extension of MPEG-1 audio (ISO/IEC 11172-3). Allows multiple channels. (Note that the popular MP3 format is a form of MPEG-1, layer 3 audio.)

- ISO/IEC 13818-4 Compliance testing procedures.

- ISO/IEC 13818-5 Software simulation of systems, audio, and video.

- ISO/IEC 13818-6. Digital storage media command and control (DSM-CC). Protocols and interfaces for controlling content selection and delivery, connection establishment, session and resource management, and related activities.

- ISO/IEC 13818-7. Advanced audio coding.

- ISO/IEC 13818-9. Real-time interfaces.

- ISO/IEC 13818-10. Conformance extensions and testing for DSM-CC.

Although DSM-CC is usually associated with video delivery (via satellite or terrestrially) and with interactive content, it is also used among audio servers and clients. The architecture describes three main parts of the system: the client, the server, and the session and resource manager (SRM). The server provides content and other services to the client, and both are "clients" of the SRM. The SRM allocates and manages network resources (such as channels, bandwidth, and network addresses.) By combining server and client components together onto the same platforms, peer-to-peer content access and delivery systems can be constructed.

These specifications include numerous implementation options. For example, MPEG-2 video can be encoded in different ways, and a DSM-CC system can be constructed to include or exclude certain features and interfaces. Normally, an outside specification will define a profile of specific options, allowing systems built using common profiles to interoperate.

DSM-CC defines or extends five distinct protocols:

- Download. The download protocol is the topic of this document, and is discussed in detail below.

- User-User. Allows remote access by the client to objects on the server. The User-User specification goes beyond the definition of specific server object classes to define classes local to the client, as well as some of the interaction with other parts of the system. The distributed object model is based on CORBA. Objects are accessed using the internet inter-ORB protocol (IIOP), with some optional extensions. Two subsets, "core" and "extended", are defined. In the model, some clients may also load content onto the server.

- User-Network. There are two parts to this protocol: Session and Resource. This protocol us used between the client and SRM, and between the server and SRM. The U-N Session protocol is used to establish sessions with the network, associated with resources with are allocated and released using the U-N Resource protocol.

- MPEG transport profiles. The specification provides profiles to the standard MPEG transport protocol (defined by ISO/IEC 13818-1) to allow transmission of event, synchronization, download, and other information in the MPEG transport stream.

- Switched Digital Broadcast-Channel Change Protocol (SDB/CCP). Enables a client to remotely switch from channel to channel in a broadcast environment. Used to attach a client to a continuous-feed session (CFS) or other broadcast feed. Sometimes used in pay-per-view.

An implementation does not always need all of these protocols. Almost all implementations in the real world use a subset. The download protocol is probably more often than any of the others

included in such a subset.

Download in Non-DSM-CC Environments

The most common uses of the DSM-CC download protocol are found in systems which don't use much else from DSM-CC:

1. DVB (an acronym for Digital Video Broadcasting) uses, as one of its protocols, a scheme adapted from the DSM-CC download protocol. The two download versions are almost completely compatible, and the same base software can be used for both. DVB has been adopted by ETSI (the European Telecommunications Standards Institute).

2. The ATSC (Advanced Television Systems Committee) Data Broadcast Standard (A/90) specifies the DSM-CC download protocol.

3. The OpenCable Common Download Specification is based on DSM-CC download.

4. Time Warner's Pegasus Interactive Services Architecture (ISA) includes use of a download protocol based on DSM-CC's download protocol.

The above standards are widely used in cable and satellite television systems. The download protocol is widely used for transfer of data of many kinds from server to client. Data types include software (usually binary executables and Java bytecode programs), program guides, images, video and audio content, directories of files, and other types of content and data.

Basic Concepts

Basic Concepts

This chapter describes some of the basic concepts needed to understand and use DSM-CC download.

Blocks, Modules, and Images

All download operations involve clients and servers. In all cases, data are transferred from server to client, although sometimes there may be control messages going the other way.

For a given session, the data are grouped into *modules*. Modules are of arbitrary size, as defined by the application. The server gives each module a number, its `moduleId`.

The protocol divides each module into *blocks*. Every block in a session is the same size, except that the last block of each module might be shorter than the others if the module's size isn't an integer multiple of the block size. Block sizes are chosen based on reasons of efficiency, network characteristics, performance, and implementation considerations; they don't need to be related to the module sizes.

In some situations, all of the modules are considered to be components of an *image*. In other situations, modules aren't directly related to one another. When the modules constitute an image, they are all downloaded by the client. When there is no concept of an image, the client may download a subset of the modules.

Each block transmitted by the server is accompanied by the applicable `moduleId` and relative block number within the module. This allows a client to recognize and select specific blocks and modules during the download operation.

Scenarios

Every download operation must conform to one of three different *scenarios*:

- In *flow-controlled download*, a complete image is transferred from one server to one client.

The client controls the flow of data using a sliding-window protocol with positive and negative acknowledgements to the server.

- In *non-flow-controlled download*, a complete image is transferred from one server to one or more clients. Although the client does not control the flow of data, there may be agreements regarding the transfer parameters. Multicast or broadcast may be used for the transfer.

- In the *data carousel* scenario, the server repeatedly transfers selected modules to whatever clients are listening. A given client may choose to receive none or any part or all of the modules in the carousel. Multicast or broadcast may be used for the transfer.

In these three scenarios, the DSM-CC specification places no restriction on the format of the data being transferred. However, in an object carousel, which is a specialization of the data carousel, a structure is defined to allow four specific classes of object: `DSM::Directory`, `DSM::File`, `DSM::Stream`, and `DSM::ServiceGateway`.

When other specifications adopt the DSM-CC protocol, one of the most common changes is to specialize or define the format of data transferred. This does not affect the core protocol, only how the contents are interpreted.

Message Types and Channels

The protocol uses seven message types, divided into two categories. They are:

Message Type	***Category***	***Direction***	***Function***
`Download ServerInitiate`	control	server to client	advertise or identify a server
`Download InfoRequest`	control	client to server	*(flow controlled scenario, optionally in non flow controlled scenario)* inform the server regarding capabilities and limitations of client *(not used in carousel)*
`Download InfoResponse`	control	server to client	*(in response to DownloadInfoRequest)* to inform client regarding download parameters and information regarding modules to be downloaded
`Download InfoIndication`	control	server to client	*(in absence of DownloadInfoRequest, for the same purpose as DownloadInfoResponse)*

Message Type	*Category*	*Direction*	*Function*
`Download DataRequest`	data	client to server	*(flow controlled scenario only)* to control the flow of data, to acknowledge negatively or positively, or to terminate complete the scenario
`Download DataBlock`	data	server to client	*(all scenarios)* transfer data from server to client
`Download Cancel`	control	either direction	*(all scenarios)* prematurely terminate a scenario in progress

Even when a message is possible in a scenario, it may not be used in some applications. For example, the carousel scenario is most frequently used without a control channel from client to server, and in those instances no `DownloadCancel` messages will be sent to from the client to the server.

With two message categories (control and data) and two directions (“up”, from client to server; and “down”, from server to client) there are four possible channels: control-up, control-down, data-up, and data-down. The specification allows each of the four channels to be distinct or combined in any way, an important consideration in most cable and satellite television environments where the available bandwith upstream (to the head-end or server) is typically much less than the bandwidth downstream (to the client).

In each scenario, the server sends at least on `DownloadInfoResponse` or `DownloadInfoIndication` message. Each of these messages contains a list of modules with their `moduleId`s and additional information. These lists are the way the server learns of the specific modules to be downloaded, or available for download.

Example Configurations

As an example of typical configurations, the following are presented. Many other configurations are possible.

Satellite Broadcast

When satellites broadcast program guides and software updates, they usually employ data carousels. The control down and data down channels are typically multiplexed into a single link from the satellite. There are no control up or data up channels.

Cable Program Guide Dissemination

Cable systems often use carousels for program guides. The control down and data down channels are typically combined, into the MPEG stream or into a downstream IP link. There are no control up or data up channels.

Set-Top Box Configuration

Some cable systems configure and update software on set-top boxes using flow controlled download. The download protocol us used to initialize a new client on the network, including transfer of the initial software. A low speed IP link is used for control up and data up. A higher speed digital link is used for control down and data down.

Mechanisms for Specialized Content

Mechanisms for Specialized Content

In a download environment, there may be a choice of data offerings available to the client. These offerings may be available serially on the same channels, or simultaneously on different channels. The download protocol provides several mechanisms for the identification of specialized content. Some of these also provide for out-of-band information, and others allow the server to tailor the content to the client, or guide the client in chosing from among multiple server data offerings. This chapter reviews those mechanisms.

Compatibility Descriptors

`DSM::CompatibilityDescriptor` is a data structure which conveys information about a piece of hardware or software. A `CompatibilityDescriptor` can be used to convey an inventory of the software or hardware available on a client, or which might be available, or which might be required.

A `CompatibilityDescriptor` includes a sequence of `DSM::InterfaceDescriptor`s. Each `InterfaceDescriptor` iincludes a software or hardware model and version, as well as a sequence of `DSM::Subdescriptor`s for additional information. The `Subdescriptor`s contain arbitrarily sequences of octets, up to 255 octets in length. `Subdescriptor` use is implementor defined.

Servers may include `CompatibilityDescriptor`s in `DownloadServerInitiate`, `DownloadInfoResponse`, and `DownloadInfoIndication` messages. Clients receiving these messages can examine the `CompatibilityDescriptor`s to decide if the messages are appropriate for them. This enables the clients to select from among multiple available sets of content.

Clients may include a `CompatibilityDescriptor` in a `DownloadInfoRequest` message. This enables the client to inform the server of its requirements. The server may examine the `CompatibilityDescriptor` and adjust the content accordingly.

Private Data

Private data is is an unstructured sequence of up to 65535 octets. Private data may be included in any of the messages which allow compatibility descriptors. Private data may also be included in `DownloadCancel` messages. The syntax and semantics of private data are defined by the implementation. (Some specifications use private data to extend or to specialize the download protocol as defined by DSM-CC.)

Uses for private data include giving a reason for the request, distinguishing among possible data offerings, redirecting to alternative download addresses, conveying information about the client or server, indicating preferences, and extending other fields. When used in `DownloadCancel` messages, private data may be used to convey information about the cause or reason for the cancellation.

Adaptation Headers

Data messages (`DownloadDataBlock` and `DownloadDataRequest`) may contain adaptation headers. An adaptation header acts as an out-of-band channel associated with data messages. It consists of an `adaptationType`, and up to 255 octets of additional information. The specification defines two `adaptationType` values, and allows the implementor to define others. The specification says that these headers are intended for purposes specific to the network.

The two adaptationType values defined by the specification are one for conditional access (encryption and scrambling) support, and one to identify the client or server.

Advertising and Redirection

A server may send `DownloadServerInitiate` messages. Each `DownloadServerInitiate` message may contain a server address, a compatibility descriptor, and private data. The server addresses in different `DownloadServerInitiate` messages may be different from one another, and they may be different from the address of the server sending the `DownloadServerInitiate` message. This has the effect of advertising the server or servers described by the messages, and possibly of redirecting the client to a

different address.

If the server sends `DownloadServerInitiate` messages with different compatibility descriptors or with different private data, then a client may choose the one suitable for its configuration.

The advertising phase is optional, and is not described in the specification as a part of any of the three scenarios.

Operation Sequences

Operation Sequences

This chapter summarizes the sequences of operations involved in the download scenarios.

Flow-Controlled Download Scenario

In the flow-controlled download scenario, an entire image is transferred from one server to one client.

The scenario begins with a `DownloadInfoRequest` from the client. This conveys the client's `maximumBlockSize` and `bufferSize`. The client might also include a compatibility descriptor and private data in the request. The server replies with a `DownloadInfoResponse`, which includes a download identifier, list of modules, and specific protocol parameters within the constraints set by the client.

During the subsequent transfer phase, the server uses a sliding window protocol to send the modules to the client. The modules are sent in order, and the blocks within each module are sent in order. The scenario completes when all modules have been sent and acknowledged positively.

When the operation is conducted over a reliable transport, the protocol omits `DownloadDataRequest` messages from the client, letting the transport mechanism handle flow control and retransmissions.

Non-Flow-Controlled Download Scenario

In the non-flow-controlled download scenario, an entire image is transferred from one server to one or more clients.

The scenario may begin in one of two ways, depending on configuration:

1. The client sends a `DownloadInfoRequest` to the server, indicating the client's maximum buffer size and maximum block size. The request may also include a compatibility descriptor and private data. The server responds with a `DownloadInfoResponse`, listing all of the

modules in the image, as well as the protocol parameters.

2. The server begins by sending a `DownloadInfoIndication`, listing the modules of the image, and the protocol parameters.

Next, the server sends the image to the client or clients. The modules and blocks of the image may be sent in any sequence. Clients do not acknowledge blocks or control transmission. The server usually repeats the transmission, again and again, allowing the clients to receive any blocks which might have been missed or corrupted in transmission.

For the client, the operation completes when all of the image has been received successfully. For the server, the operation completes when it is configured to do so.

Data Carousel Scenario

In the data carousel scenario, the server sends one or more modules to one or more clients. Each of the clients may elect to receive any, all, or none of the modules.

The server begins by sending one or more `DownloadInfoIndication` messages. Each of these may be different. Each client uses the compatibility descriptors and private data to select which of the `DownloadInfoIndication` messages is applicable to that client. The module list in each message includes a (proper or improper) subset of all of the modules transmitted by the carousel.

Each client uses the module list from the selected `DownloadInfoIndication` message to determine which modules to download.

The server repeatedly transmits all of the modules, and each client examines the incoming data blocks, accepting only those blocks which pertain to the modules which that client has chosen to download. The server may transmit the blocks and modules in any order.

For the client, the operation completes when all of the selected modules have been received successfully. For the server, the operation repeats indefinitely (based on configuration).