# DSM−CC Client Integration Guide

PRELIMINARY DRAFT
2002.06.06

**Bionic Buffalo Corporation**

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

## Copyright and License

Copyright 1997, 1998, 2000, 2002 Bionic Buffalo Corporation. All rights reserved, including moral rights.

*No−Charge License:* This document may be reproduced and distributed free of charge, only if all of the following conditions are met: (1) The document is reproduced only in its entirety, including all page headings and copyright notices; (2) No changes, deletions, additions or other alterations are made; (3) The document is reproduced only in the original PDF format, or on paper or other printed form; and (4) No charge is made for copies.

Other licenses are available from Bionic Buffalo Corporation.

## Author and Publisher

Bionic Buffalo Corporation
2533 North Carson Street, Suite 1884
Carson City, Nevada 89706−0147
United States of America

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

## Related Products

This document pertains to products licensed by Bionic Buffalo Corporation. For other documents and for additional information, see *http://www.tatanka.com/prod/info/dsmcc.html*, or send e−mail to *query@tatanka.com*.

## Document Revision History

Original release, 2002.06.06

Printed 20:24:14, Saturday, 29 June, 2002.

Updates and revisions of this document are announced on Bionic Buffalo s DSM CC e mail announcement list. See *http://www.tatanka.com/bbc/lists.html* for information.

## Additional Document Information

Project name: *sri lanka*.

Document file name: *srilanka_clntgr_20020606*.

Written and illustrated using StarOffice and OpenOffice.

# Contents

# Preface

This is a guide for those using Bionic Buffalo's software to implement systems acting as DSM−CC clients.

Bionic Buffalo DSM−CC and related software are structured into a number of separate component packages. This is the document that explains the integration of the separate packages into a client system.

The central features of this document are an explanation of various technical matters, and a Task List. The Task List is the first chapter, with page references to more detailed discussions within this document.

THIS IS A PRELIMINARY VERSION OF THE FIRST RELEASE OF THIS DOCUMENT. ERRORS ARE LIKELY!

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

# Task List

Throughout this document, tasks which must be performed when porting and hosting Bionic Buffalo DSM−CC software are noted in paragraphs surrounded by boxes, each with a task number.

This chapter lists the porting and hosting tasks described within this document, along with the numbers of the pages where they are found.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.
*http://www.tatanka.com*

# Overview

## Introduction

DSM−CC is a specification defining mechanisms for communication, control, and data transfer in networked multimedia applications. In such environments, the real−time content (audio or video) is transferred using some different mechanism (such as MPEG−2 transport), while DSM−CC is used to select content, control operations, manage resources, and perform other tasks. In general, DSM−CC is used for all of the functions except for the real−time content itself.

Naturally, there is a close relationship between DSM−CC and content transfer protocols. Some (maybe all) DSM−CC information may be carried along with the audio or video content inside the MPEG−2 transport or other content stream.

Multimedia clients using DSM−CC may implement functions not directly related to multimedia. For instance, it is common to incorporate a web browser into a multimedia client. DSM−CC provides for these additional applications, and may even share connections with them. (The internet also may be used for some or all of the DSM−CC protocols.)

This document describes the integration of a client complying with the DSM−CC specification. To the extent that client functionality is available in a server, this document also serves to describe server integration. This document covers the use of Bionic Buffalo's pre−written DSM−CC implementations, not the protocols themselves, nor the internals of the Bionic Buffalo software. It is a guide for someone who wants to use existing DSM−CC code, not for someone who wants to implement the specification.

Bionic Buffalo's implementations are modular, as described below. There are additional documents for each of the modules used.

## MPEG−2 Family of Specifications

DSM−CC (Digital Storage Media Command and Control) is a group of related protocols and interface definitions used for communication among multimedia clients and servers. The DSM−CC specification is the sixth of the MPEG−2 specifications, which 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. One example of such an umbrella specification is the DAVIC specification, which describes an overall architecture for networked multimedia content storage, distribution, and use. Because of the number of DSM−CC options possible, such an umbrella specification or profile definition is a necessary requirement for implementing a DSM−CC system.

## Protocols, Interfaces, and Portability

DSM−CC defines various protocols and interfaces in sufficient detail to build a client which will interoperate with a server, but does not provide complete definitions for use within the client itself.

Although there is an API defined for application−level programs, there is no equivalent for system−level functionality. This means that user applications may be portable, but functions such as configuration, resource management, connection establishment or tear−down, and system calls are not likely to be the same from one platform to another.

The standard application API is defined as part of the User−User interface. It arises naturally from the User−User definition, since that definition is based on CORBA IDL. (CORBA IDL has a well−defined mapping to APIs in various languages.) If an application uses only the User−User API, then it will be highly portable. If it uses any other API, then some porting will be required when moving to a new environment.

Bionic Buffalo has (of necessity) defined APIs for its software implementing the other parts of DSM−CC. The Bionic Buffalo APIs are also based on CORBA, since the infrastructure already must be included for the User−User implementation, and CORBA offers a number of other advantages as well.

## Five Distinct DSM−CC Protocols

DSM−CC defines or extends five distinct protocols:

- 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. (The Download protocol is sometimes considered a User−Network protocol, but it doesn’t involve the SRM, so we categorize it separately.)

- 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.

- Download. Several variations of this protocol allow transfer of content from server to client, either within the MPEG transport stream or on a separate (presumably high−speed) channel. Flow−controlled download allows the download operations to be negotiated and controlled by the client. A variation of download is an autonomous "data carousel" on the server which repeatedly downloads information; the download carousel client waits for the information without initiating the transfer. An extension to the data carousel is the "object carousel", which presents downloaded information as objects compatible with the objects defined by the User−User API. (The choice of download or IIOP protocols is embedded in the object’s IOR, so the means of access is transparent to the client application.)

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.
*http://www.tatanka.com*

- 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. For this reason, Bionic Buffalo provides client DSM−CC software in separate modules:

- `spain` user−user (U−U) client
- `italy` user−network (U−N) client
- `honduras` download and carousel client
- `costa rica` switched digital broadcast (SDB) channel change protocol (CCP) client and server
- `samoa` DSM−CC server and client subset

(Additionally, Bionic Buffalo offers server software, not covered in this document.) These modules communicate among themselves, and also with the other client system and application software. These communication mechanisms, and the way in which the modules are integrated into a client, are explained in this document. The details of each module’s interfaces (including API, common data structures and semantics) are to be found in the documents corresponding to each of the above products.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved. srilanka_clntgr_20020606
*http://www.tatanka.com* 2002−06−29 20:18

# Network Architecture

DSM−CC clients are deployed in a variety of network topologies, using different physical, link, network, and transport layer substrata. Common environments include cable, LAN, internet, telephone (synchronous and plesiochronous), satellite broadcst, and radio networks.

In a system employing DSM−CC, a client is likely connected to more than one server. This may be a division by function (*e.g.*, one server for session control, another for content) or a division by content type (*e.g.*, different servers for different content), or some other division. Server connections may be dedicated, switched, assigned by the network, or selected by the client. Connections may be made by passive, one−way receipt of broadcast data, or may be made by active log−in using resource allocation and configuration negotiations.

A client may have more than one physical network connection. For example, a set−top box may have a one−way high−speed downlink (satellite or cable) as well as a low−speed uplink or bidirectional channel. The five DSM−CC protocols can be combined in different ways over the available physical connections.

DAVIC terminology is used sometimes in this document. Although not every system is DAVIC−compliant, the model provides a useful point of reference.

## Information Flows

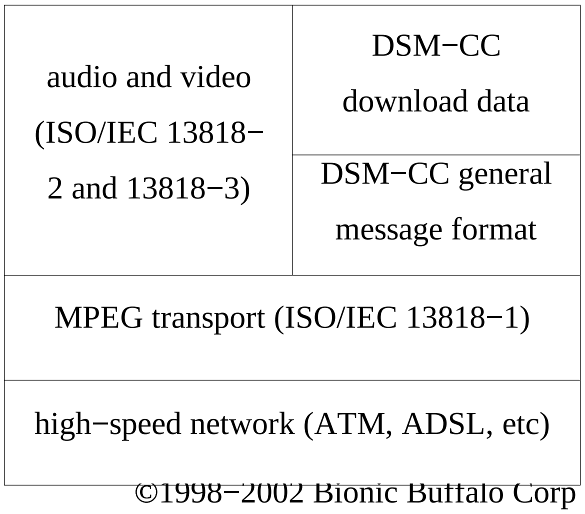
DAVIC allows various protocol stack configurations for the realization of a network. Information flows are classified into groups:

- S1 (content) flow: (MPEG audio and video, download data)
- S2 (application control) flow: protocols which alter the behaviour of the S1 flow (User−User, download control
- S3 (session control) flow: session control protocols (User−Network)
- S4 (network) flow: link control and signalling protocols (Q.2931)
- S5 (management): network management protocols

Although there are five flows, they may be combined for transmission over only one or two channels. For example, it is common to combine S2, S3, and S4 on a lower speed bi−directional channel, while placing the S1 flow on a high−speed, uni−directional channel. (Note that the download protocol is often split between two different channels, since it uses both S1 and S2 flows.)

## The S1 Flow: Content

The most common protocol stack configuration for the S1 (content) flow is shown here. Audio and video content share a single high−speed channel. Most often, the high−speed channel is one−directional, from the head−end or server to the client.


audio and video (ISO/IEC 13818−2 and 13818−3)
DSM−CC download data
DSM−CC general message format
MPEG transport (ISO/IEC 13818−1)
high−speed network (ATM, ADSL, etc)
©1998−2002 Bionic Buffalo Corp

Other S1 stacks are possible. For instance, download data can be carried using UDP/IP over the internet.

The DSM−CC general message format refers to an overall structure for DSM−CC messages. In the S1 flow, they are encapsulated within the private data section of the MPEG transport stream.

All DSM−CC messages, except for User−User messages, begin with a common DSM−CC message header. (The User−User messages are based on RPC, either IIOP or ONC RPC, or some variant of one of these.) The header contains a `dsmccType` field which specifies the type of DSM−CC message:

| ***dsmccType*** | ***Description*** |
|---|---|
| `0x00` | reserved to ISO/IEC 13818−6 |
| `0x01` | user−network configuration |
| `0x02` | user−network session |
| `0x03` | download |
| `0x04` | SDB channel change protocol |
| `0x05` | user−network pass−through |
| `0x06..0x7f` | reserved to ISO/IEC 13818−6 |
| `0x80..0xff` | defined by implementor or user |

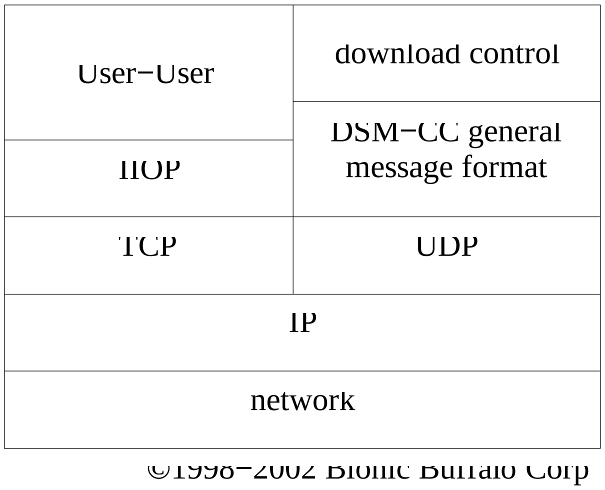
Although content can be multiplexed by encapsulating multiple program streams within a transport stream, the same result is sometimes achieved by using multiple transport connections on the same network. For example, in an ATM network, different virtual channels may be used to separate the programs.

For more information on MPEG streams, see the *MPEG Streams* section below.

## The S2 Flow: Application Control

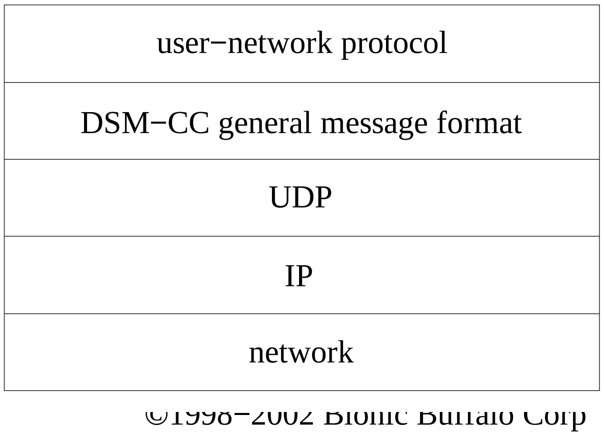
The S2 flow is used for protocols which control or alter the S1 flow. This includes the User−User protocol and the download control protocol. It is a bi−directional flow.

The S2 flow requires a reliable transport mechanism, which is usually TCP/IP. In this example, the reliability for download is provided by the download control protocol itself, which will retry for missing or error blocks.



One possibility for the network layer is to encapsulate IP in MPEG transport. Systems are sometimes built with asymmetrical IP connections: MPEG downstream, PPP upstream (for example). Other variations include different protocols from TCP/IP and UDP/IP for the transport layer.

## The S3 Flow: Session Control



The S3 flow, used for session−level protocols, generally follows the same approach as is used for download control. There is typically only one kind of content: the User−Network session and configuration messages.

A typical activity might be to create the S1 and S2 connections needed to view a movie or listen to audio.

DSM−CC envisions session establishment being initiated by the server as well as by the client. While, for most systems, this might not be needed in the context of content delivery, it may sometimes be useful for system management. (For example, a server might connect at its own convenience to upgrade a client’s software.)

### The S4 Flow: Network Connection Control

The S4 flow is used to establish connections with the network. Although DSM−CC doesn't define any S4 flows, it is aware of them because it understands the underlying network as a configurable resource.

An example of the type of protocol used in an S4 stack is Q.2931.

### The S5 Flow: Network and System Management

DSM−CC doesn't define any management protocols. The most commonly used protocols in the S5 flow are SNMP and CMIP.

## Reference Model

DAVIC also defines a reference model with five subsystems:

- The Content Provider System (CPS)
- The CPS−SPS Delivery System
- The Service Provider System (SPS)
- The SPS−SCS Delivery System
- The Service Consumer System (SCS)

In other words, a content provider, a service provider, and a consumer system, with two delivery systems in between. Of course, only the last part is the client in most cases. (DSM−CC also allows for "producer" clients, which upload content to servers.)

Between each of these subsystems, DAVIC defines a reference point. Reference points are designated A0, A1, A2, etc. A partial list of reference points is:

| ***Reference Point*** | ***Description*** |
|---|---|
| A0 | This point is usually internal to the client device. It is point between the network interface and the rest of the internal components. (Usually, the network interface is a distinct physical module within the client.) |
| A1* | If an enhanced in−house network is used, this is the point between the in−house network and the client. |
| A1 | SCS/delivery system boundary. The point between the delivery system and the client device. |
| A2 | Point between the network termination and the distribution network. |

| ***Reference Point*** | ***Description*** |
|---|---|
| A4 | Point between the core network and the access network. The SPS/SCS delivery system is divided into two parts: the core network (from the SPS) and the access network (to the consumer). |
| A9 | Delivery system/SPS boundary. |
| A10 | SPS/delivery system boundary. |
| A11 | Delivery system/CPS boundary. |

Most of these are of little concern to client integrators, but they are sometimes mentioned, and are listed here for convenience.

## MPEG Streams

The MPEG transport stream is defined by ISO/IEC 13818−1. It is a one−way stream, sent without acknowledgements or NAKs. (Transport streams can be saved as files on CD−ROMs or disks, and often are.)

### General Structure

MPEG transport streams are sent in 188−byte packets. (A single packet can fit into four 48−byte ATM cells.) If a payload is longer than will fit into a transport packet, it may be continued in the next packet. The `payload_unit_start_indicator` flag is set to indicate that a payload begins within the current transport packet.

A single transport stream can carry multiple programs, and each program can consist of multiple elementary streams. A program is a collection of elementary streams sharing a single time base. (*Example:* Each television channel in a cable system is a program.) An elementary stream is audio or video from a single source. (*Example:* A single television channel might consists of three elementary streams: the video, plus left and right audio.)

In addition to programs, transport streams also carry tables. There are five kinds of tables defined:

| ***Table*** | ***Description*** |
|---|---|
| Program Association Table (PAT) | Map program number to PID |
| Program Map Table (PMT) | Specifies PID values for the components of one program |
| Network Information Table (NIT) | Physical network parameters |

| ***Table*** | ***Description*** |
|---|---|
| Conditional Access Table (CAT) | Associates one or more Entitlement Management Messages (EMMs) with a unique PID value |
| private data tables | (various uses) |

Tables (except for the NITs) are organized into sections. Each section of a table may be transmitted separately. There is no requirement that the sections of a table be transmitted in any specific sequence. By segmentation of the table into sections, a portion (one section) of a table may be updated without transmitting the entire table.

In transport streams, the tables are collectively called Program Specific Information (PSI). The first byte of each transmitted section is a table identifier byte. Some values are:

| ***Table ID*** | ***Type of Data*** |
|---|---|
| `0x00` | program association section |
| `0x01` | conditional access section |
| `0x02` | program map section |
| `0x03..0x37` | (reserved to ISO/IEC 13818−1) |
| `0x38..0x39` | (reserved to ISO/IEC 13818−6) |
| `0x3a` | DSM−CC multiprotocol encapsulated data |
| `0x3b` | DSM−CC User−Network messages (except Download Data) |
| `0x3c` | DSM−CC Download Data |
| `0x3d` | DSM−CC Stream Descriptors |
| `0x3e` | DSM−CC Private Data |
| `0x3f` | (reserved to ISO/IEC 13818−6) |
| `0x40..0xfe` | defined by user or implementor |
| `0xff` | (forbidden value; used for fill) |

There is a packet identifier (PID) field in the header of each transport packet. The PID associates the packet with a specific table or program. Transport packets with a common PID value relate to the same program or table. PID values are allocated as follows:

| ***PID Value*** | ***Use*** |
|---|---|
| `0x0000` | Program Association Table (PAT) |
| `0x0001` | Conditional Access Table (CAT) |
| `0x0002..0x000f` | (reserved) |
| `0x0010..0x1ffe` | (dynamically assigned; used for programs and tables) |
| `0x1fff` | Null packet |

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

The Program Association Table (PAT) is used to find other tables, some of which in turn are used to convey information about the dynamically−assigned PIDs. The Conditional Access Table (CAT) conveys information relating to scrambling and encryption, and is not used by DSM−CC.

## Program Association Table

Each entry in the Program Association Table (PAT) is a pair of numbers: a program number and a PID.

PIDs cannot be used across a system to identify programs, because the PID values may change when streams are multiplexed, demultiplexed, or relayed. The PID assigned to a program by a server may not be the same PID seen by the client. Furthermore, each elementary stream or table in a program uses a separate PID, so a set of PIDs (rather than a single PID) would be needed to identify a program.

For these reasons,  a different identifier − the program number − is used for system−wide identification of a program. The User−Network protocol identifies programs by their program numbers, which the server has in common with the client.

The PID specified for a program number in the PAT designates which packets contain the program’s Program Map Table (PMT). The PMT specifies which additional PIDs are used for the various elementary streams or tables of a program.

The PAT allows translation from the User−Network mapping to PID, allowing selection of a specific program from a multiplexed stream.

## Program Map Table

The Program Map Table (PMT) identifies the elementary streams and tables belonging to a program. Each entry in the PMT has three parts: a stream type, the stream’s PID, and descriptors which provide additional information about the stream.

Stream types include the following:

| ***Stream Type*** | ***Description*** |
|---|---|
| 0x00 | (reserved) |
| 0x01 | MPEG−1 video (per ISO/IEC 11172) |
| 0x02 | MPEG−2 video (per ISO/IEC 13818−2) |
| 0x03 | MPEG−1 audio (per ISO/IEC 11172) |
| 0x04 | MPEG−2 audio (per ISO/IEC 13818−3) |
| 0x05 | private tables |
| 0x06 | PES packets containing private data |
| 0x07 | MHEG (per ISO/IEC 13522) |

| ***Stream Type*** | ***Description*** |
|---|---|
| `0x08` | DSM−CC (per ISO/IEC 13818−1) |
| `0x09` | auxiliary data (per ISO/IEC 13818−1/11172−1) |
| `0x0a` | DSM−CC multiprotocol encapsulation |
| `0x0b` | DSM−CC (User−Network Messages only) |
| `0x0c` | DSM−CC (Stream Descriptors only) |
| `0x0d` | DSM−CC (any type, including User−Network Messages or Stream Descriptors) |
| `0x0e..0x7f` | (reserved) |
| `0x80..0xff` | (defined by implementor or user) |

(Although ISO/IEC 13818−1 says that DSM−CC uses stream type `0x08`, this is contradicted by the DSM−CC specification itself, which calls for values `0x0b..0x0d`.)

The stream descriptors in the PMT provide additional information about the stream. (This includes things such as frame rates, language, clock information, copyright, and other data.)

## Stream Descriptors

Stream descriptors, whether used in PMTs or DSM−CC table sections, have a common format. They begin with a descriptor tag and a length field. Some assigned tag values are:

| ***Descriptor Tag*** | ***Descriptor Type*** |
|---|---|
| `0x00` | (reserved) |
| `0x01` | (reserved) |
| `0x02` | video stream descriptor |
| `0x03` | audio stream descriptor |
| `0x04` | hierarchy descriptor |
| `0x05` | registration descriptor |
| `0x06` | data stream alignment descriptor |
| `0x07` | target background grid descriptor |
| `0x08` | video window descriptor |
| `0x09` | conditional access (CA) descriptor |
| `0x0a` | ISO 639 language descriptor |
| `0x0b` | system clock descriptor |
| `0x0c` | multiplex buffer utilization descriptor |
| `0x0d` | copyright descriptor |
| `0x0e` | maximum bitrate descriptor |

| *Descriptor Tag* | *Descriptor Type* |
|---|---|
| `0x0f` | private data indicator descriptor |
| `0x10` | (reserved) |
| `0x11` | (reserved) |
| `0x12` | (reserved) |
| `0x13` | DSM−CC carousel identifier descriptor |
| `0x14` | DSM−CC association tag descriptor |
| `0x15` | DSM−CC deferred association tags descriptor |
| `0x16` | (reserved to DSM−CC) |
| `0x17` | DSM−CC normal play time (NPT) reference descriptor |
| `0x18` | DSM−CC normal play time (NPT) endpoint descriptor |
| `0x19` | DSM−CC stream mode descriptor |
| `0x1a` | DSM−CC stream event descriptor |
| `0x1b..0x3f` | (reserved) |
| `0x40..0xff` | (defined by user or implementor) |

## DSM−CC Sections

When DSM−CC information is carried within MPEG transport streams, it is carried within DSM−CC sections. These are table sections with a table identifier of `0x38..0x3f`. To allow them to be found easily, these sections are found in streams with types of `0x0a..0x0d`.

## Multiprotocol Encapsulated Data

Although defined by the DSM−CC specification, multiprotocol encapsulated data (table type `0x3a`, carried in stream type `0x0a` or `0x0d`) is in practice used not only by DSM−CC. When it is implemented, it is also used by applications and other system software.

Multiprotocol encapsulated data is used to carry other protocols within the MPEG transport stream. These messages must be encoded according to the ISO/IEC 8802−2 Logical Link Control (LLC) and ISO/IEC 8802−1a SubNetwork Attachment Point (SNAP) specifications. Most commonly, TCP/IP is carried within LLC, which in turn is carried by the MPEG transport stream.

If the MPEG transport stream is only downward to the client, bidirectional protocols such as TCP/IP must use an alternate upstream channel. The alternate channel of is a PPP link over a lower−speed telephone line.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

## Options, and a Warning

There is no single, correct way to do many things allowed by the DSM−CC specification. Implementors and systems integrators seem driven to find many incorrect ways, as well. So what is encountered in the field will not always comply with the specifications.

As an example of the choices available, consider the transport of User−Network messages. This can be done in several ways:

- they can be sent as DSM−CC sections in the MPEG transport stream
- they can be encapsulated in TCP/IP or UDP/IP, and sent as multiprotocol encapsulated data in the MPEG transport stream
- they can be encapsulated in TCP/IP or UDP/IP, and sent on a separate channel
- they can be sent directly over ATM

Each of these techniques is described by the specification. And for every correct possibility, there are many incorrect ones: improper encapsulation, incorrect addressing, bad resource identification, disallowed extensions, forbidden subsetting, excessively creative or stupid error handling, and misuse of the MPEG transport stream.

Builders of DSM−CC clients are somewhat at the mercy of those who construct the networks, servers, distribution systems, and head−end plant. Defensive design is the order of the day. A designer should not assume that the way an MPEG stream is put together is the only way it will ever be done.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

# Functional Components

This chapter describes the main functional components of clients using one or more of the DSM−CC protocols, as they relate to DSM−CC.

The relevant features are discussed, and the other features may be completely ignored. For example, in the discussion of MPEG demultiplexing, the main concern is extracting the streamdescriptors, and little attention is paid to the video content itself.

## General Considerations

This section describes some general considerations needed to understand the ensuing discussion.

### Simplified Descriptions

Many client systems are built in quantity, so the manufacturing costs become significant considerations. In such cases, manufacturing cost savings may justify engineering expenditures not reasonable in environments such as desktop PCs. For instance, many clients use special hardware (usually in the form of a chip) for MPEG demultiplexing and decoding. If only one copy of the system were built, the cost of integrating the MPEG demux/decode chips into the design wouldn’t be justified: it would be cheaper just to buy a faster main CPU.

For this reason, it is important not to read the accompanying diagrams simply as software or hardware component diagrams.

This also implies a level of complexity in an implementation which might not be obvious from the diagrams. Using the above example, an MPEG demux chip combined with a general purpose CPU might require buffering, multiported memory, sychronization logic and other features which won’t be shown on a high level diagram. If the demux function were implemented in software, then the equivalent software infrastructure would be required: semaphores, threads, locks, and other real−time programming constructs. In either case, a lot of necessary plumbing is omitted.

### Feature Set

Most DSM−CC clients rely on a small subset of the features described here. Low−end devices are designed and marketed based on low cost. Rather than describe several different designs, this document will use a single, feature−rich design, which can be pruned as needed for specific markets.

## Personalities

A personality is a collection of configured attributes for a client. It is not defined by the specifications. It is a feature of Bionic Buffalo’s implementations to make it easier for a single client to connect to multiple service providers.

Many of the attributes in a personality are also programmed separately. When no personality is specifically selected, the separately−programmed attributes are used to create a default personality.

Most of the personality attributes relate to an initial connection to a service provider. The most complex structure is the compatibility descriptor, described under the *User−Network* section, below.

*Task 1: Define personalities.*

## Security

Access to DSM−CC services and objects is subject to controls implemented using the `DSM::Access` and `DSM::Security` interfaces. However, the definitions of these leave a number of matters up to the implementors.

Furthermore, the specification does not discuss security on the client system itself.

### Local Enforcement Mechanism

On some hosts, there is a native security enforcement mechanism. For example, Unix−like systems have a user/group/world model to control access to files and other resources. If a Unix−like model is available, Bionic Buffalo’s software will work within that model.

*Task 1: Configure the system and DSM−CC for the local enforcement mechanism.*

### Authentication

The mechanism to authenticate a user to an object is undefined by the specification. It might vary from one object to the next. Possible mechansisms vary greatly in complexity: simple passwords, encrypted passwords, digital signatures, and more complex authentication sometimes involving smart cards.

The implementor must provide software to implement the necessary authentication mechanism.

*Task 2: Provide software to authenticate the user.*

The authentication process may involve use of certificates. This may not be a trivial task in a manufacturing environment.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

*Task 3: Provide certificates for authentication of the user.*

In addition to user authentication, it is sometimes necessary to authenticate the host. This is especially true when the user is asked to provide credit card numbers or other personal information for pay−per−view or other transactions.

Bionic Buffalo DSM−CC provides SSL encryption with accompanying authentication, but other mechanisms must be provided by the implementor.

*Task 4: Provide software to authenticate the host.*

In any case, host or root certificates must be provided for the host.

*Task 5: Provide certificates for authentication of the host.*

## Encryption

If an object’s allSecure bit is set, then messages to and from that object must be encrypted. Bionic Buffalo provides SSL for this purpose, but if another method is required, then software must be provided by the implementor.

*Task 6: Provide software for encrypted communication with objects.*

## Principal

CORBA and DSM−CC transactions require a `Principal`, which is an opaque array of bytes specifying the person or other entity requesting an operation.

Each service provider may expect something different for the Principal value. Software must be provided to set `Prinicipal` occasionally as needed.

*Task 7: Provide software to set the value of* `Prinicipal`.

## Local Object Security

Security mechanisms must be set up for local objects accessed remotely. SSL is provided with the DSM−CC software, but it must be configured.

*Task 8: Configure security for remote access of local objects.*

Any alternative software (other than SSL) must be provided by the implementor.

*Task 9: Provide software for secure access to local objects.*

Unless simple passwords are used, certificates are also required.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.
*http://www.tatanka.com*

*Task 10: Provide certificates for local object security.*

## Asynchronous Exception Processing

Applications calling DSM−CC library routines can handle exceptions as they arise. For the various daemons, autonomous processes and tasks, however, exception handling must be configured.

As exceptions occur, the DSM−CC software refers to an exception configuration table to determine what action to take. Possible actions include:

- ignore the error
- retry the operation (subject to configurable limitations)
- make entries in the system logs
- make entries in a private DSM−CC log
- pass messages to designated error handlers
- for certain exceptions, take specific actions
- restart certain processes or tasks
- reduce or limit use of resources
- disable specified functionality

More than one action may be taken for any given error. Actions may be defined for groups of errors. Some actions are built−in, others (such as running a program in response to an exception) may require software provided by the implementor.

*Task 11: Configure asynchronous exception handling.*

## MPEG Transport or Program Streams

Although it is conceivable to have a DSM−CC client without MPEG transport or program stream capability, in practice these mechanisms are usually used to deliver the content.

### Common Procedures

Messages received with detected errors (using CRCs or other indicators) should be discarded.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

### Transport Streams

DSM−CC data in transport streams is carried in the private data section. In addition, DSM−CC needs access to stream descriptors, program map tables (PMTs) and program association tables (PATs).

The transport stream demultiplexor must delivery the required information to DSM−CC as it arrives. There are library calls for this purpose. Each incoming stream descriptor, User−Network message, or table implies a call to deliver the information.

*Task 12: Deliver incoming mapping tables (PATs and PMTs).*

When sections encompass multiple transport packets, the implementor is responsible for reassembly of DSM−CC sections from the transport stream. The DSM−CC code will parse the section data.

*Task 13: Deliver incoming DSM−CC data sections from transport streams.*

Multiprotocol encapsulated data, even if it carries DSM−CC data, must be handled differently from the other DSM−CC sections. The DSM−CC software will not process the protocols. Messages must be handed of to an appropriate protocol stack.

For example, if the encapsulated messages are TCP/IP and UDP/IP messages, they must be passed to the internet stack for processing. The DSM−CC protocol will access encapsulated protocols using the Berkeley sockets interface (or some custom network interface, if you are using one).

*Task 14: Hand off multiprotocol encapsulated messages from the transport stream to suitable protocol stacks.*

### Program Streams

Stream descriptors may also be embedded within MPEG program streams, where they are carried as PES packets. As with transport streams, a library call should be made for each incoming stream descriptor.

*Task 15: Deliver incoming stream descriptors from program streams.*

## Network

The DSM−CC software will use standard networking (such as TCP/IP) if so configured. The Berkeley sockets interface is normally used, but the interface code can be modified for some alternative API.

Incoming traffic is handled in a straightforward fashion. Generally, DSM−CC doesn't care how or through what connection a message arrived. Outgoing messages, however, may require some decisions: which interface to use, whether to employ SSL, what MTU limits should be employed, and so forth. This can be complicated when ATM networks are used, or when the path to a host is split (incoming on one interface, outgoing on another).

DSM−CC uses additional network configuration to supplement the system's routing table and interface parameters. These can be changed at runtime, but in any case the parameters must be defined.

*Task 16: Define extended network configuration and access parameters.*

The software must know its own network address in many situations, but there are not standard mechanisms for this purpose. The type of address depends on the network type, and may range from quasi−physical values (such as MACs) to purely logical values (such as telephone numbers or ATM addresses). The implementor must tell the DSM−CC software what its addresses are.

*Task 17: Specify additional network addresses.*

For internet connections, listener tasks are provided for the reserved ports, to intercept DSM−CC messages. In some implementations, other kinds of network connections are used, so the listener must be replaced by an implementor−provided routine. In still other circumstances, the incoming messages share ports with other traffic, so the incoming traffic must be demultiplexed.

*Task 18: Deliver relevant incoming network messages to DSM−CC.*

## Local Object Services

Bionic Buffalo DSM−CC provides local directory and file objects with the same interfaces as server directories and files.

*Task 19: Specify local DSM::Directory root, and populate the tree.*

If no specific access permissions are required for access, then default values must be provided. These are the same values provided by the `DSM::Access` interface.

*Task 20: Provide default access permissions for local objects.*

Any object which uses some different access value must be configured specifically. This would apply especially to local files or directories accessible remotely.

Security for local objects is included in this requirement. For example, a password might be required for remotely accessible objects, and encrypted communication might be required.

*Task 21: Provide access permissions for specific local objects.*

## User−Network Client

There are a great deal of capabilities in the User−Network client. Practically no client needs all of them, if for no other reason than few clients have a dozen different network connections.

*Task 22: Define a functional subset of User−Network.*

The client's device address is a six−byte identifier unique on the network. Often, the MAC is used.

*Task 23: Set the device address.*

The client provides compatibility descriptors to the network at the time connections are established. The most important use by the network is server selection (certain clients might be assigned to specific servers). The servers use the compatibility descriptors to tailor their behaviour to specific classes of client.

The format of compatibility descriptors is generally described by the DSM−CC specification, but the contents are defined by organizations which implement systems and provide services.

Each personality, including the default personality, is associated with a set of compatibility descriptors.

*Task 24: Specify compatibility descriptors.*

## User−User Client

### Additional Service Context Data

Service Context contains additional "hidden" data sent with IIOP messages. Some service context information is defined by the specifications, and is included automatically. (Configuration is described elsewhere in this document, but "service context" may not be mentioned explicitly.)

Implementors may define additional service context data, or they may be required to provide such data by some service providers.

*Task 25: Define additional service context data.*

### Functional Subset

Many clients will use a subset of User−User capabilities. This reduces image size and may simplify end−user support.

*Task 26: Select a functional subset of User−User.*

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

### Resources

The User−User client is generally configurable to limit its use of resources. Buffering and caching can consume a lot of memory, which might not be available in the client. The use of memory and other resources should be adjusted to a value appropriate for the implementation.

*Task 27: Configure User−User resource limits.*

### `DSM::File`

The `DSM::File` implementation is configurable for the buffering strategy and resource limits. This configuration should be adjusted for the implementation.

*Task 28: Configure buffer strategy and limits for the* `DSM::File` *implementation.*

### `DSM::Session`

During session establishment, the capability descriptors are sent to the server. The server may download objects to the client to start the session. (The choice of objects may depend on the capability descriptors.) The server may also download the initial application.

Bionic Buffalo's DSM−CC software can handle the download, but passing the downloaded objects to the implementation, along with loading and starting the initial application, are inherently non−portable operations. To illustrate the process, a sample program which accepts and runs downloaded shell scripts in Unix−like environments is provided. However, the implementor must, in general, provide software to accept the downloaded objects and to run the application.

The objects are available as opaque sequences of octets. For certain kinds of objects defined by the DSM−CC specification, there is a well−defined interpretation of the opaque sequence. When such well−known objects are downloaded, the DSM−CC software can manage them as expected. (For example, the `DSM::File` operations are available for downloaded `DSM::File` objects.) When the object interfaces are not familiar to DSM−CC, then a simple memory buffer is handed to the application or implementation.

*Task 29: Accept downloaded objects at session establishment.*

Loading and starting applications can range in complexity from the trivial to the complex. Shell scripts and Java bytecodes are not too bad, but executables which require dynamic linking and relocation can be difficult. In any case, the implementor must provide the necessary code.

*Task 30: Load, link, and run initial applications.*

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.
*http://www.tatanka.com*

When sessions are suspended and restored, the context is saved at the client. If some mechanism other than a simple memory buffer or disk file is required, then the implementor must provide the software. In any case, the implementor must manage suspended sessions so they don’t overrun the client’s resources. (Perhaps a limit can be placed on the number of suspended sessions, or the amount of memory or disk space allocated to suspended sessions.)

*Task 31: Manage suspended sessions.*

### `DSM::Stream`

An implementation or application refers to a `DSM::Stream` object by opaque object reference. The specification provides no API to associate the `Stream` object with a specific program in an MPEG transport stream. Bionic Buffalo provides an extended interface for this purpose. In cases where there may be multiple incoming programs or transport streams, it is essential to determine which program corresponds to the selected stream object.

This is listed here because it is usually a system function, integrated with the implementation. Normally, this is not done by the application (although it can be, if wanted).

*Task 32: Acquire the stream object’s program number*

### `DSM::State`

The standard software saves object states in memory or on disk. It must be configured. Additionally, management of saved states (perhaps some garbage collection mechanism) is necessary to limit resource utilization.

*Task 33: Configure and manage saved states.*

## Download Client

Most download operations are managed transparently to the application. However, the download client must make decisions about buffer allocation, timing, and other parameters. The download client must be configured for default values of these parameters.

*Task 34: Configure the download client.*

Carousel operations are passive downloads: they operate without requests or flow control. Because the carousel tends to be used differently than other download functions, its parameters can be configured separately, even though they are equivalent to normal download parameters. For example, because the carousel file sizes might be different from on−demand download file sizes, the optimal amount of buffer space might differ.

*Task 35: Configure the carousel download parameters.*

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

For each object or data item in the carousel, the client can adopt one of two strategies:

- wait until the object is needed before acquiring it
- acquire it in anticipation of need

The first strategy minimizes memory, while the second minimizes latency. The carousel client provides an API for dynamic strategy selection. However, a default strategy is necessary until such time as the carousel management functions are invoked.

The default strategy may be selected on an object−by−object basis. Certain files might use an anticipatory strategy, while others might use a deferred policy.

*Task 36: Provide default carousel management strategies and parameters.*

A carousel sends occational `download_info_indication` messages, which list the modules (with lengths, versions, and other information) which will be downloaded. The carousel client can monitor these messages to look for new or updated modules. It can then call a user−defined procedure with notification of the new or updated information. The procedure can optionally change the policy to cause that new or updated information to be donwloaded.

*Task 37: Accept download change information.*

## Timeout Values

There are a variety of timeout values throughout the DSM−CC software. The software is configured with default values, which should be evaluated by the implementor. Factors such as network speed, memory size, processor speed, and other software influence expected times.

Most timeout values can be adjusted dynamically, but initial values must be reasonable.

*Task 38: Adjust default timeout values.*

## Startup Behaviour

As with other software, the DSM−CC software must be initialized and started before use. Most implementations require DSM−CC be running constantly almost from the time the system is powered up.

The first step is to specify which portions of DSM−CC (User−User, User−Network, Download, etc.) are included in the implementation. This is done in the DSM−CC master configuration file, which in turn is consulted by the Makefiles for the available components.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.
*http://www.tatanka.com*

*Task 39: Specify which components of DSM−CC are included.*

Depending on the host environment, DSM−CC will run (in a memory−mapped environment) as a collection of threaded or unthreaded processes, or it will run (in an unmapped RTOS) as a number of tasks or threads. The mechanism for starting the programs is inherently system−dependent.

In any case, a root process or thread is started, which will start the other processes.

*Task 40: Start the root DSM−CC process or thread.*

## Session Establishment

Sessions may be established by an application, or they may be established by the system. The Bionic Buffalo software allows multiple, concurrent sessions. In any case, session establishment involves the same procedure, using the `DSM::Session` interface.

*Task 41: Establish automatic, initial session.*

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.
*http://www.tatanka.com*

# Porting and Hosting

## Logical Processes

Each Bionic Buffalo software component, when implemented, exists as one or more logical processes.

A logical process differs from a Unix−style process in that it may exist in the same address space with another logical process. It is similar to a Unix−style process in that there is no assumption that it will share memory with another logical process. In a Unix−like environment, a logical process is implemented as a Unix−style process.

## Logical Threads

Logical threads are analagous to threads (in Unix−like environments) and to tasks (in RTOS environments). However, logical threads may be implemented in non−preemptive environments (such as DOS).

Logical threads may share data structures in memory with other logical threads. When they do, the logical threads belong to the same logical process.

In a multithreaded or multitasked environment, a logical thread is implemented as a thread or task. Newer Bionic Buffalo code uses the POSIX threads API in such environments.

In a single−thread environment, the threading is simulated in the code itself by adding a very simple scheduler and by making sure that no logical thread ever blocks execution by another logical thread. The mechanisms by which this is done are beyond the scope of this document. (Please refer to the *Work List Architecture™ Reference* for details.)

## Host Environments

Bionic Buffalo classifies host environments as follows:

- (Class S) one real execution thread in a single address space; no preemptive scheduling except perhaps for interrupt handlers (seen in very simple embedded systems, signal processors, and classical DOS)
- (Class E) traditional embedded system environment, with an RTOS supporting multiple threads or tasks, preemptively scheduled, without memory mapping
- (Class U) traditional Unix−style, multiple−process environment, where each process has one or more threads, memory is mapped (processes do not share address space among themselves)

- (Class M) multiprocessor environments supporting communication among the processes on different CPUs; the constituent CPUs may each be Class S, Class E, or Class U environments
- (Class C) multiprocessor environments supporting MPI

Each of the logical processes of a component is written for hosting on one or more of these environments. Except for `samoa` (which supports Class U only), Bionic Buffalo client DSM−CC software can be hosted in Class E, Class U, or Class M environments.

Bionic Buffalo intends to support certain reference platforms as host environments. Not all products have been ported to the reference platforms, but they will be supported them should a customer require it. (In other words, the port will be done if someone wants it.) The reference platforms are:

- Class S: FreeDOS
- Class E: RTEMS, eCOS
- Class U: Solaris, BSD, Linux, MinGW (Windows)
- Class M: various Class U platforms on a LAN
- Class C: SMP Linux

## Development Environments

Linux, Solaris, BSD, and (to a limited extent) Cygwin are supported development environments.

## Threads and Other Real−Time Extensions

If you are hosting on platforms which support POSIX threads (pthreads) and POSIX.4 (real−time POSIX), then there is little to do for this part. If you are using a subset as commonly found on the supported host platforms, then there is still little to do, because we use the same subsets.

*Task 42: Confirm that reasonable subsets of pthreads and POSIX.4 extensions are available.*

Without these interfaces, or some reasonable subset, you will probably find porting Bionic Buffalo's DSM−CC difficult.

## Interprocess Communications

The various logical processes of DSM−CC need a way to communicate among themselves. This includes implementor code and applications which make library calls: the library routines employ some form of interprocess communication (IPC) to talk to the DSM−CC logical processes.

The DSM−CC IPC code is isolated to a few modules. Modification or replacement may be needed, and should not be too difficult. The supported host platforms use several mechanisms:

- Berkeley sockets (Class U or Class M platforms)
- queues in shared memory (Class E platforms) (implemented using the POSIX.4 primitives)
- POSIX.4 message queues (Class U platforms)

If your host environment supports something similar to these, then you may be able to use the supported host software as a starting point.

*Task 43: Assure the availability of interprocess communications.*

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.
*http://www.tatanka.com*

# Index

## Alphabetical Index

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.

©1997, 1998, 2000, 2002 Bionic Buffalo Corporation; All Rights Reserved.