angola
DSM-CC Download Common API Reference Manual

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_api_reference*
Formal revision date: Tuesday 2003.09.23
Last modified: 2003-09-24-14:10

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

Introduction

Introduction

Bionic Buffalo's angola product implements portions of the DSM-CC download protocol as specified in *Extensions for DSM-CC* (ISO/IEC 13818-6), one of the MPEG-2 family of specifications. It is available only as an adjunct to the guatemala (download server) and honduras (download client) products. It is not released separately.

The angola product includes a library of software which can be used as part of an implementation of download server or client applications.

This document describes the API available with the angola library.

Before reading this document, 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.

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. The API descriptions in this document specify the IDL interface as well as the C interface to each class.

`Angola::CancelReason`

Name

`Angola::CancelReason` -- specifies the reason an operation was or is to be cancelled

Synopsis - IDL

```
#include <angola.idl>

module Angola
        {
        typedef unsigned short CancelReason ;
        } ;
```

Synopsis - C

```
#include <angola.h>

typedef unsigned short          Angola_CancelReason ;
```

Description

The `Angola::CancelReason` gives the reason an operation was or is to be cancelled.

Notes

1. Defined values are:

Value	*Sender*	*Name*	*Description*
0x0000			(reserved)
0x0001	S,C	SCENARIO_TIMEOUT	scenario timer expired
0x0002	S,C	INSUFFICIENT_MEMORY	not enough memory available
0x0003	S	AUTHORIZATION_DENIED	authorization denied
0x0004	S,C	FATAL_ERROR	unspecified fatal error
0x0005	S	INFO_REQUEST_ERROR	the download server cannot offer the client's required maximum block size or buffer size

Value	*Sender*	*Name*	*Description*
0x0006	S	COMPATIBILITY_ERROR	the server cannot find an appropriate image from the client, based on the client's compatibility descriptor
0x0007	S	UNRELIABLE_NETWORK	the client has requested a reliable network session, but the connection in use is unreliable
0x0008	C	INVALID_DATA	the client received data which did not match that specified in the DownloadInfoIndication or DownloadInfoResponse messages
0x0009	C	INVALID_BLOCK	a block with an invalid block number or module identifier was received
0x000a	C	INVALID_VERSION	a block with an unexpected version number was received
0x000b	C	ABORT	the client has decided for unspecified reasons to terminate the operation
0x000c	C,S	RETRANSMISSION_LIMIT	the maximum allowed number of retransmissions would be exceeded
0x000d	C	BAD_BLOCK_SIZE	unable to support the server's choice of block size
0x000e	C	BAD_WINDOW	unable to support the server's choice of window size
0x000f	C	BAD_ACK_PERIOD	unable to support the server's choice of acknowledgement period
0x0010	C	BAD_WINDOW_TIMER	unable to support the server's choice of window timeout
0x0011	C	BAD_SCENARIO_TIMER	unable to support the server's choice of scenario timeout
0x0012	C	BAD_CAPABILITIES	the client cannot parse the compatibility descriptor sent by the server
0x0013	C	BAD_MODULE_TABLE	the client cannot parse the module table sent by the server

Value	*Sender*	*Name*	*Description*
0x0014 to 0xffff			(reserved)

Example

Availability

angola (Bionic Buffalo)

See also

Reference

`Angola::ModuleSynopsis`

Name

`Angola::ModuleSynopsis` -- description of a download module

Synopsis - IDL

```
#include <angola.idl>

module Angola
    {

    struct ModuleSynopsis
            {
            unsigned long               downloadId ;
            unsigned short              blockSize ;
            unsigned short              moduleId ;
            unsigned long               moduleSize ;
            octet                       moduleVersion ;
            CORBA_OctetSeq              moduleInfo ;
            unsigned long               moduleTimeOut ;
            unsigned long               blockTimeOut ;
            unsigned long               minBlockTime ;
            DSM::ConnBinder             Taps ;
            CORBA_OctetSeq              userInfo ;
            Tibet::DatasetReference     reference ;
            unsigned short              status ;
            unsigned short              priority ;
            unsigned short              copy_limit ;
            unsigned long               flags ;
            } ;

    typedef sequence<ModuleSynopsis>    ModuleSynopsisList ;

    } ; /* end Angola */
```

Synopsis - C

```
#include <angola.h>

typedef struct Angola_ModuleSynopsis
        {
        CORBA_unsigned_long             downloadId ;
        CORBA_unsigned_short            blockSize ;
        CORBA_unsigned_short            moduleId ;
        CORBA_unsigned_long             moduleSize ;
        CORBA_octet                     moduleVersion ;
        CORBA_sequence_octet            moduleInfo ;
        CORBA_unsigned_long             moduleTimeOut ;
        CORBA_unsigned_long             blockTimeOut ;
        CORBA_unsigned_long             minBlockTime ;
        DSM_ConnBinder                  Taps ;
        CORBA_sequence_octet            userInfo ;
        Tibet_DatasetReference          reference ;
        CORBA_unsigned_short            status ;
        CORBA_unsigned_short            priority ;
        CORBA_unsigned_short            copy_limit ;
        CORBA_unsigned_long             flags ;
        }
        Angola_ModuleSynopsis ;

typedef struct CORBA_sequence_Angola_ModuleSynopsis
        {
        CORBA_unsigned_long             _maximum ;
        CORBA_unsigned_long             _length ;
        Angola_ModuleSynopsis         * _buffer ;
        }
        CORBA_sequence_Angola_ModuleSynopsis ;

typedef CORBA_sequence_Angola_ModuleSynopsis
                                Angola_ModuleSynopsisList ;
```

Description

The `Angola::ModuleSynopsis` structure describes a single download module. A `ModuleSynopsisList` is a sequence of `ModuleSynopsis` structures.

The `moduleId`, `moduleSize`, `moduleVersion`, and `moduleInfo` members are from information in the `DownloadInfoResponse` and `DownloadInfoIndicatio`n messages. The `moduleTimeOut`, `blockTimeOut`, `minBlockTime`, `Taps`, and `userInfo` members are passed from server to client, but only in object carousels. The other members are not exchanged between client and server, and are used only for communication between applications and the service objects implemented by the download libraries.

`ModuleSynopsis` structure contents are defined in the `Server`, with the help of the `Server`'s `Application`. Although they are not passed in that form to the client, most of the structure is reconstructed by the `Client` from information originating in the `Server`. Exceptions are the `reference`, `status`, `priority`, and `flags` members, which are local, and are used to communicate between the libraries and the `Applications`.

Notes

1. Except for the `reference`, `status`, `priority`, and `flags` members (and then only as allowed), a `Client`'s `Application` should not modify any members of a `ModuleSynopsis` structure received from the libraries.

2. The `downloadId` and `blockSize` match the `downloadId` and `blockSize` from the `DownloadInfoResponse` or `DownloadInfoIndication` message containing the module's description.

3. The `moduleId` is an identifier assigned by the server. It is unique within the scope of a single `downloadId`.

4. The `moduleSize` is the length of the module, in octets.

5. The `moduleVersion` is the version of the module, with respect to the `downloadId`. The server sets this field arbitrarily, and changes it when the module contents have changed. Each `downloadDataBlock` includes this field, allowing the client to detect when the version is changed during a download operation.

6. The `moduleInfo` is a sequence of from 0 to 255 octets. Its purpose is to allow the client to identify the module. Its contents are implementation-specific. They may include file name, content type, entry point address, version, or any other information required by an implementation.

7. For object carousels, the `moduleInfo` sequence contains the `BIOP::ModuleInfo` structure, which is constructed from the `moduleTimeOut`, `blockTimeOut`, `minBlockTime`, `Taps`, and `userInfo` members. The `moduleInfo` is an alternate, unparsed representation of this same information.

8. *(object carousels only)* The `moduleTimeOut` is the maximum number of microseconds allowed to acquire the module.

9. *(object carousels only)* The `blockTimeOut` is the the maximum number of microseconds

allowed to acquire the next block of a module, after another block has been acquired.

10. *(object carousels only)* The `minBlockTime` is the minimum period of time between blocks.

11. *(object carousels only)* The `Taps` describe the connection on which the module is broadcast.

12. *(object carousels only)* The contents of the `userInfo` field are specific to the implementation.

13. The `reference` describes the file, object, or data structure containing or to contain the module's contents. Not all values of the discriminator are valid for all operations.

14. The `status` describes the condition of the module transfer operation.

15. The `priority` is applicable to carousels. It sets a relative frequency with which the module is transmitted. The interpretation is based on the interleave mechanism.

16. The `copy_limit` is applicable to carousels. It specifies the number of copies of of the module which will be sent by the server before removing the module from the carousel. A value of zero indicates no limit: the module will be repeatedly sent, indefinitely, until it is explicity removed from the carousel by the application.

17. The `flags` allow for additional information to be communicated between application and service object.

18. Earlier versions of `Angola` defined a `modulePath` member, which is no longer defined. Its function has been subsumed by that of the `reference` member.

Example

Availability

`angola` (Bionic Buffalo)

See also

Reference

`Angola::Scenario`

Name

`Angola::Scenario` -- specifies the download scenario

Synopsis - IDL

```
#include <angola.idl>

module Angola
        {
        enum Scenario
                {
                UNSPECIFIED_SCENARIO,
                FLOW_CONTROLLED_DOWNLOAD,
                NON_FLOW_CONTROLLED_DOWNLOAD,
                DATA_CAROUSEL,
                OBJECT_CAROUSEL
                } ;
        } ;
```

Synopsis - C

```
#include <angola.h>

typedef unsigned long           Angola_Scenario ;

#define Angola_UNSPECIFIED_SCENARIO             0
#define Angola_FLOW_CONTROLLED_DOWNLOAD         1
#define Angola_NON_FLOW_CONTROLLED_DOWNLOAD     2
#define Angola_DATA_CAROUSEL                    3
#define Angola_OBJECT_CAROUSEL                  4
```

Description

The `Angola::Scenario` specifies which download scenario is used. The three scenarios are defined by the specification.

Notes

1. In the `FLOW_CONTROLLED_DOWNLOAD Scenario`, a complete set of data is transferred from the server to one client. Flow control, including windows, timeouts, and acknowledgements, is used.

2. In the NON_FLOW_CONTROLLED_DOWNLOAD Scenario, there is no flow control. One set of data is transferred from the server to one or more clients.

3. In the DATA_CAROUSEL Scenario, there is no flow control, and the server repeatedly sends the data to one or more clients. Multiple modules may be transmitted, and any client may choose to read or ignore any or all of the modules transmitted.

4. The OBJECT_CAROUSEL Scenario is a specialization of the DATA_CAROUSEL Scenario, with additional semantics defined by the specification.

Example

Availability

angola (Bionic Buffalo)

See also

Reference

`Angola::SessionParameters`

Name

`Angola::SessionParameters` -- the specified or negotiated protocol parameters for a download session

Synopsis - IDL

```
#include <angola.idl>

module Angola
        {
        struct SessionParameters
                {
                Angola::Scenario        scenario ;
                unsigned short          block_size ;
                unsigned short          window_size ;
                unsigned short          acknowledgement_period ;
                unsigned long           acknowledgement_timeout ;
                unsigned long           scenario_timeout ;
                unsigned long           download_id ;
                unsigned long           unacknowledged_limit ;
                unsigned long           block_size_limit ;
                unsigned long long      rate_limit ;
                } ;
        } ;
```

Synopsis - C

```
#include <angola.h>

typedef struct Angola_SessionParameters
        {
        Angola_Scenario             scenario ;
        CORBA_unsigned_short        block_size ;
        CORBA_unsigned_short        window_size ;
        CORBA_unsigned_short        acknowledgement_period ;
        CORBA_unsigned_long         acknowledgement_timeout ;
        CORBA_unsigned_long         scenario_timeout ;
        CORBA_unsigned_long         download_id ;
        CORBA_unsigned_long         unacknowledged_limit ;
        CORBA_unsigned_long         block_size_limit ;
        CORBA_unsigned_long_long    rate_limit ;
        }
        Angola_SessionParameters ;
```

Description

The `Angola::SessionParameters` structure describes the protocol employed or to be employed for a download session. Although the meaning of the parameters are unambiguous, sometimes the structure is used to represent constraints on a session, or a starting point for negotiation. The parameters in fact employed for a session are not always the same as requested.

Not all members are meaningful in all contexts. Especially, not all parameters are exchanged between client and server. Refer to the descriptions of specific operations for applicability and use.

Notes

1. The `scenario` is one of the three scenarios defined by the specification.

2. The download_id is the identifier for the scenario. It must be unique within the connection for flow-controlled and non-flow-controlled scenarios, and unique within the network for carousel scenarios. It is assigned by the server or server application.

3. The `block_size` is the length (in octets) of the data carried in each `DownloadDataBlock` messages. (The final block may be shorter than the others.)

4. The `window_size` is the number of blocks which may be outstanding without an acknowledgement. A value of zero indicates an infinite number; that is, no acknowledgement is ever required. The `window_size` is only meaningful for the flow-controlled download scenario.

5. The `acknowledgement_period` is the number of blocks the client must receive before sending a positive acknowledgement. This value should be less than or equal to the `window_size`. A negative acknowledgement may be sent at any time, and the last block must be acknowledged regardless of the number of blocks received. The `acknowledgement_period` is only applicable to the flow-controlled download scenario.

6. The `acknowledgement_timeout` is the timeout period for acknowledgements. It is specified in microseconds. It is meaningful only in the flow-controlled download scenario.

7. The `scenario_timeout` is the timeout period for the entire scenario. It is specified in microseconds.

8. The `download_id` is assigned by the server. It is used to distinguish among download

sessions. It should be unique within the connection for flow-controlled and non-flow-controlled scenarios. It should be unique within the network for carousels.

9. The `unacknowledged_limit` is the maximum number of octets which can be received by the client without acknowledgement. It is a factor in the server's calculation of block and window sizes. It is meaningful only in the flow-controlled download scenario.

10. The `block_size_limit` is the maximum allowable data block size.

11. The `rate_limit` is specified in octets/second. It is the maximum rate at which data may be transmitted.

Example

Availability

`angola` (Bionic Buffalo)

See also

Reference

`Angola::Statistics`

Name

`Angola::Statistics` -- conveys information regarding activity during a download session

Synopsis - IDL

```
#include <angola.idl>

module Angola
        {
        struct Angola_Statistics
                {
                TimeBase::TimeT         time_active ;
                unsigned long           total_modules ;
                unsigned long           total_blocks ;
                unsigned long long      total_octets ;
                unsigned long           modules_sent ;
                unsigned long           modules_received ;
                unsigned long           blocks_sent ;
                unsigned long           blocks_received ;
                unsigned long           acknowledgements_sent ;
                unsigned long           acknowledgements_received ;
                unsigned long long      octets_sent ;
                unsigned long long      octets_received ;
                unsigned long           carousel_cycles ;
                unsigned long           total_errors ;
                unsigned long           transmit_errors ;
                unsigned long           receive_errors ;
                unsigned long           other_errors ;
                } ;
        } ;
```

Synopsis - C

```
#include <angola.h>

typedef struct Angola_Statistics
        {
        TimeBase_TimeT              time_active ;
        CORBA_unsigned_long         total_modules ;
        CORBA_unsigned_long         total_blocks ;
        CORBA_unsigned_long_long    total_octets ;
        CORBA_unsigned_long         modules_sent ;
        CORBA_unsigned_long         modules_received ;
        CORBA_unsigned_long         blocks_sent ;
        CORBA_unsigned_long         blocks_received ;
        CORBA_unsigned_long         acknowledgements_sent ;
        CORBA_unsigned_long         acknowledgements_received ;
        CORBA_unsigned_long_long    octets_sent ;
        CORBA_unsigned_long_long    octets_received ;
        CORBA_unsigned_long         carousel_cycles ;
        CORBA_unsigned_long         total_errors ;
        CORBA_unsigned_long         transmit_errors ;
        CORBA_unsigned_long         receive_errors ;
        CORBA_unsigned_long         other_errors ;
        }
        Angola_Statistics ;
```

Description

The `Angola::Statistics` structure conveys information regarding activities during a download operation. Not all fields are meaningul in all contexts.

Notes

1. The `time_active` is the interval since the client or server was activated until now, or until it was deactivated or the operation cancelled.

2. `total_modules` is the number of modules to be transferred until the scenario ends.

3. `total_blocks` is the number of blocks to be transferred during the scenario.

4. `total_octets` is the number of octets of data to be transferred during the scenario. It excludes header and control information.

5. `modules_sent` is the number of complete modules which have been sent so far. This number is not meaningful for clients. It is reset to zero at the beginning of each carousel cycle.

6. `modules_received` is the number of complete modules which have been received so far. It is not meaningful for servers.

7. `blocks_sent` is the number of blocks which have been sent successfully. It is not meaningful for clients. It is reset to zero at the beginning of each carousel cycle.

8. `blocks_received` is the number of blocks which have been received successfully. It is not meaningful for servers.

9. `acknowledgements_sent` is the number of blocks which have been acknowledged so far. It is not meaningful for servers.

10. `acknowledgements_received` is the number of blocks which have been acknowledged so far. It is not meaninful for clients. It is reset to zero at the beginning of each carousel cycle.

11. `octets_sent` is the number of octets which have been sent successfully so far. It is not meaningful for clients. It is reset to zero at the beginning of each carousel cycle.

12. `octets_received` is the number of octets which have been received successfully so far. It is not meaningful for servers.

13. `carousel_cycles` is the number of complete carousel cycles which have transpired. It is meaningful only for servers in a carousel scenario.

14. `total_errors` is the total number of communication errors found so far.

15. `transmit_errors` is the number of transmit errors encountered so far.

16. `receive_errors` is the number of receive errors encountered so far.

17. `other_errors` is the number of communication errors, other than transmit or receive errors, encountered so far.

Example

Availability

`angola` (Bionic Buffalo)

See also

Reference