NAGRAVISION
KUDELSKI GROUP

CONFIDENTIAL

Conditional Access System

IRD Commands

Specification

V2.2.0

Save #02

Issued on September 02, 2008 for Taiwan Broadband Communications.

File Name:	CakSpeIrd020200.pub.doc
Last Modification:	28.08.2008 13:20
Printed on:	28.08.2008 13:20

Table Of Contents

List Of Figures

1 Introduction

1.1 Purpose

This document defines the general format of an IRD command as well as generic NagraVision commands, such as “Reset PIN Code” or “Force Tune”. It also defines the rules for defining manufacturer’s or operator’s specific commands.

1.2 How to Read this Document

Chapter 2 gives an overview of the concept of IRD Commands and a summary of existing commands.

Chapter 3 specifies the generic format of an IRD Command.

Chapter 4 specifies more in details the existing Nagravision IRD Commands.

1.3 Document History

2.2.0 12-Aug-2008, Jean-Luc Bussy
- Added PPP username and password to command “Set Callback Parameters”

2.1.0 21-Apr-2008, Jean-Luc Bussy
- Added command “Set Callback Parameters”

2.0.0 13-Sep-2007, Jean-Luc Bussy
- Change IRD command format to include the zone addressing mode

1.11.0 06-Aug-2007, Jean-Luc Bussy
- Added IPTV force tune commands

1.10.0 15-Jun-2007, Jean-Luc Bussy
- Added commands “Force Download by ID”, “Force Identification by Zone” and “Force Stand-By by Zone”

1.9.0 02-Apr-2007, Jean-Luc Bussy
- Added “Set Pairing Configuration” command

1.8.0 08-Feb-2007, Sébastien Robyr
- Replaced “enable/disable Basic Access smart cards” by “enable/disable BGA” commands

1.7.0 18-Jan-2007, Sébastien Robyr
- Adapted document to new template
- Added “enable/disable Basic Access smart cards” commands

1.6.0 18-Jul-2006, Jean-Luc Bussy
- Added reboot STB command

1.5.1 13-Jun-2006, Sébastien Robyr
- Added card-less related maximum IRD command size
- Corrected max IRD-CMD size for DNASP2 and DNASP3

1.5.0 16-May-2006, Jean-Luc Bussy

Added the following commands:
- Force Tune by Zone ID
- Cancel Zone ID
- Update Cohabitation Tables

1.4.0 13-Feb-2006, Jean-Luc Bussy
- Added EMM Wake-Up commands.

1.3.20 26-Jan-2006, Jean-Luc Bussy

- Added command "Set Zone ID"

1.3.19 11-Jan-2006, Marc Pighini

- Added Usage monitoring and Broadcast network operator lock commands.

1.3.18 09-Jan-2005, Jean-Luc Bussy

- Updated 'Change Usage ID' command.

1.3.17 20-May-2005, Jean-Luc Bussy

- Added command “Format Logical Disk”.

1.4 Used Acronyms

Abbreviation	Definition
BGA	**B**all **g**rid **a**rray – shortcut used to designate a Nagravision ICC that is embedded into the decoder (soldered on the Set-Top Box's PCB)
CA	**C**onditional **A**ccess
CAK	**C**onditional **a**ccess **k**ernel
CLESE-1	**C**ard-**l**ess **e**mbedded **s**ecurity **e**ngine, first generation
CRL	**C**ontent **r**evocation **l**ist
DNASP2	**D**igital **N**agra **s**ecurity **p**rocessor, second generation
DNASP3	**D**igital **N**agra **s**ecurity **p**rocessor, third generation (also known as Aladin)
DVB	**D**igital **v**ideo **b**roadcasting
ICC	**I**ntegrated **c**ircuit **c**ard – synonym of Smart card
IRD	**I**ntegrated **r**eceiver **d**ecoder
MKY	**M**ovie **k**e**y**
NVM	**N**on-**v**olatile **m**emory
PCB	**P**rinted **c**ircuit **b**oard
STB	**S**et-**T**op **B**ox

1.5 References

[1] Force Identification, Implementation Guidelines V1.0.0

[2] IRD Master/Slave, Solution Overview, Issue 1.0.0

[3] IRD Master/Slave, Implementation Guideline, Issue 0.0.3

[4] ANSI/STCE 41 2003, POD Copy Protection System

[5] NagraVision, Data Item Loader, Application Programming Interface, V 1.0.4 or higher.

[6] NagraVision, Conditional Access Kernel, EMM Wake-Up Specification, V1.0.1

1.6 Notational conventions

Hexadecimal values

Hexadecimal values are written in `courier new` police with a quote sign in the front of the number. For instance, the hexadecimal representation of the decimal value 685 is written `'2AD`.

Hexadecimal numbers may be represented with a space between each byte value, as for example: `'01 34 F2 B5`.

Sometimes, in the IRD command examples, a space may visually separate field values, as shown in Figure 1.

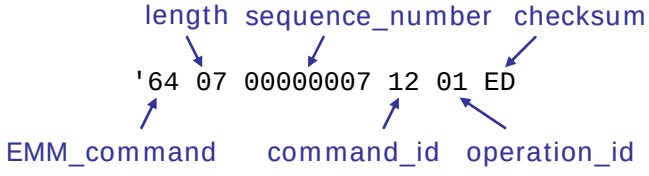

Figure 1 – PIN Code IRD Command representation

Generic Text

Expressions that appear in `courier writing style` indicate a precise descriptor field name as it appears in the corresponding specification document. All source code occurrences also appear in `courier writing style`.

Highlighted sentences in this document describe pending issues or items to be clarified.

Expressions that appear in *italic* introduce a new concept of naming convention.

All buffers presented in the figures are represented with their most significant Bytes on the left. This is just a representational convention. The figure here below illustrates two different ways of representing the constant `CST_EXAMPLE` of value `'DEADBEEF12345678:`

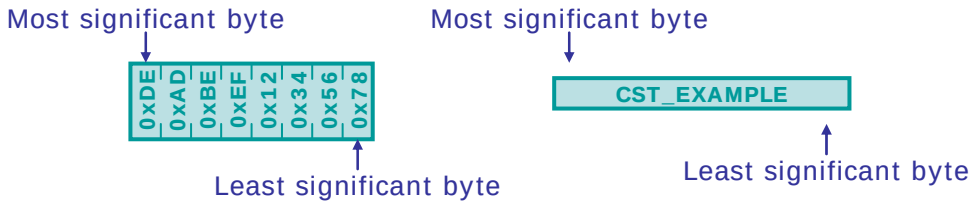

Figure 2 – Notational convention for buffers of Bytes

1.7 Trademarks

Any company's or product name(s) found herein may be the trademarks or registered trademarks of their respective companies.

2 Overview

IRD commands allow the head-end to send messages to the set-top box in a secured way. IRD commands are carried by EMMs. They can benefit from EMM addressing mode. It means that a message can be addressed either to one single set-top box or to all set-top boxes.

The CA Kernel embedded in the set-top box is not dependent at all on IRD commands. It gets the command from the smart card and forwards it to the set-top box application without additional processing. The set-top box application is completely responsible for IRD command management. Periodicity of commands (coming from the fact that commands are carried by EMMs) has to be managed by the set-top box application by means of the sequence number. If a command has to be split into several commands due to the EMM length limitation[1], it is also the responsibility of the set-top box application to re-build the original command[2].

NagraVision has defined a set of generic commands. The table below gives a synopsis of these commands along with the associated command identifier. Refer to Chapter 4 for a detailed description.

Name	command_id	operation
Reset PIN Code	0x12	0x01
Mail	0xC0	0x01
Force Tune	0xC1	0x01
Force Identification – Standard	0xC2	0x01
Force Identification – By Zone ID	0xC2	0x02
Set Macrovision CPS	0xC4	0x01
Configure STB	0xC5	0x01
Set Network ID	0xC6	0x01
Master/Slave Initialization	0xC7	0x01
Master/Slave Cancellation	0xC7	0x02
Master/Slave Single Shot	0xC7	0x03
Automatic Master/Slave	0xC7	0x04
Set PIN Code	0xC8	0x01..0xFF
Force Stand-by – Standard	0xC9	0x01
Force Stand-by – By Zone ID	0xC9	0x02
Configure Camlock	0xCA	0x00, 0x01
Copy Protection – Validate POD_ID/Host_ID	0xCB	0x00
Copy Protection – Revoke POD_ID/Host_ID	0xCB	0x01
Copy Protection – Force Authentication	0xCB	0x02
Copy Protection – Set Key Session Period	0xCB	0x03
Restore Factory Setting	0xCC	0x01

[1] The maximum size of the `command_body` that can be carried by one IRD-CMD is 70 Bytes for DNASP3, 54 Bytes for DNASP2 and 148 Bytes for CLESE1 (the complete IRD buffer returned by the ICC includes 3 more Bytes, the `EMM_command`, the `length` and the `checksum`).

[2] Some implementations at the Head End, as the current SMS interface, already limit the maximum size of the IRD-CMD to 48 Bytes (due to historical compliance with DNASP2 smart cards). The developer shall therefore be carefull when using IRD commands and verify possible implementation limitations.

Name	command_id	operation
Force Tune with Timeout	0xCD	0x01
Force Tune by Zone ID	0xCD	0x02
IPTV – Force Tune with Timeout	0xCD	0x03
IPTV – Force Tune by Zone ID	0xCD	0x04
Pop-up	0xCF	0x00..0x01
MovieKey	0xD0	0x00
Push-VOD – Content Configuration	0xD1	0x00
Push-VOD – Partition Formatting	0xD1	0x01
Push-VOD – Erase Asset	0xD1	0x02
Push-VOD – Erase Metadata File	0xD1	0x03
Push-VOD – Set Downloads Wake-Up	0xD1	0x04
Force Software Download - Standard	0xD2	0x00
Force Software Download – By Download ID	0xD2	0x01
Change Usage ID – Resident software	0xD3	0x00
Change Usage ID – Downloadable apps	0xD3	0x01
Set Community Type	0xD4	0x00
Format Logical Disk	0xD5	0x00
Usage Monitoring	0xD6	0x00
Broadcast Network Operator Lock	0xD7	0x00
Set Zone ID	0xD8	0x00
Cancel Zone ID	0xD8	0x01
EMM Wake-Up – Disabling the feature	0xD9	0x00
EMM Wake-Up – Setting new parameters	0xD9	0x01
Update Cohabitation Tables	0xDA	0x00
Reboot STB	0xDB	0x00
Disable BGA (embedded Smart card)	0xDC	0x00
Enable BGA (embedded Smart card)	0xDC	0x01
Set Pairing Configuration	0xDD	0x00
Set Callback Parameters	0xDE	0x00

Table 1 - Commands Summary

All commands required by a manufacturer or an operator that do not belong to this list may result in a specific command. Refer to §4.10 for a description of the procedure allowing the definition of a specific command.

3 IRD Command Format

Description

This section defines the generic format of an IRD command. A command is composed of a header, an optional addressing information field, an optional payload and a checksum.

Syntax

```
IRD_command(){
  header()
  addressing_info()
  payload()
  checksum                    8    uimsbf
}

header(){
  tag                         8    uimsbf    h64
  length                      8    uimsbf
  sequence_number             32   uimsbf
  command_id                  8    uimsbf
  addressing_mode(){
    by_zone                   1    bslbf
    reserved                  1    bslbf     b0
  }
  operation                   6    uimsbf
}

addressing_info(){
  if(by_zone){
    zone_id                   48   uimsbf
  }
}

payload(){
  for(i=0; i<N; i++){
    data                      8    uimsbf
  }
}
```

Semantics

header()

`tag` — This 1-byte field identifies an IRD command. It is always equal to h64.

`length` — This 1-byte field specifies the length in bytes of all the following fields, from `sequence_number` to `checksum` included. The maximum value of this field is listed below and depends on the type of EMM carrying the command:

DNASP2	55
Aladin (101)	71
CLESE1	149
Merlin (112)	83
Merlin (132)	103
Merlin (164)	128

The value in brackets corresponds to the EMM length.

sequence_number	This 4-byte field is incremented whenever a command is generated by the head-end. Since IRD commands are carried by EMMs, the set-top box application may be notified of the same command several times. It is the responsibility of the set-top box application to process the sequence number in order to avoid a command to be run several times. To do so the sequence number of the last *x* commands run by the application may be stored in NVM. The value *x* depends on the maximum number of different commands that could be broadcast at the same time on the network. It is operator dependent.
command_id	This 1-byte field specifies the command identifier.
addressing_mode	
.by_zone	When set to 1, this flag indicates that the command is addressed by zone. In such a case the addressing_info structure contains the identifier of the zone targeted by the command. The command shall be discarded if the zone identifier contained in the command differs from the one stored in the set-top box. If the set-top box has not been assigned any zone ID (see 4.23.1) or if its zone ID has been cancelled (see 4.23.2), the command shall be discarded as well.
[3]operation	This 6-bit field is used in conjunction with the command_id. The couple (command_id, operation) uniquely identifies a command. This field was defined on 8 bits in the previous format.

addressing_info()

zone_id	This 6-byte field serves as the zone identifier. The format of the zone ID is customer specific.

payload()

data	Additional data (optional)
checksum	Two's complement of the sum of all bytes from the command_id to the last data byte of the payload. The sum of all bytes from the command_id to the checksum must be equal to 0. For instance, the checksum of the reset PIN code command here after is equal to `ED (`12+01+ED=0) `64 07 00000007 **12 01** ED

[3] If the addressing mode is set to b00, the command syntax is backward compatible with the previous format having an operation field coded on 8 bits.

4 Generic IRD Commands

4.1 Reset PIN Code

Description

Forces the STB to clear the parental code. This may be required if the subscriber lost the PIN code, or when reclaiming the STB from the field.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    h12
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None.

Notes

1. If multiple PIN codes are available, then the operation field indicates the PIN code number.

Example

The following command reset the PIN code number 1.

```
IRD_command = `6407000000071201ED
```

4.2 Mail

Description

This command provides mail messages to the STB. The management of the messages is the STB responsibility.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC0
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()           0..n  uimsbf
  payload(){
    mail_id                     10  bslbf     Mail message number
    total_segment                6  bslbf     Total number of segments
    priority                     2  bslbf     0 normal priority
                                               1 high priority
                                               2 emergency
                                               3 reserved
    segment_number               6  bslbf
    for (i=0; i<N; i++){
      message                    8  bslbf     Mail message body
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

mail_id	Unique mail number
total_segment	Total number of segments required to carry the whole message. It's a 6-bit variable covering the range [1..63].
priority	Influences the STB behavior. For example, normal priority would not affect the display, while emergency mail would be displayed on the screen without manual intervention.
segment_number	Identifies the current segment. The first segment is equal to 0 and the last segment is equal to total_segment-1.

Notes

1. If the total length of a mail does not fit into an IRD command, then the mail is split in several segments, each having the same mail id and consecutive segment numbers.

4.3 Force Identification

4.3.1 Standard

Description

Forces the STB to display its Nagra S/N along with the UA of its smart card on the screen for a while.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC2
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()           0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

Note

See document [1] for more information about this command.

Example 1

Standard force identification:

```
IRD_command = `64 07 00001630 C2 01 3D
```

Example 2

Force identification of set-top boxes belonging to the zone 123:

```
IRD_command = `64 0D 00001630 C2 81 00000000007B 42
```

4.3.2 By Zone ID (deprecated)

Description

This command is deprecated. It is replaced by the standard “Force Identification” command with the zone addressing mode enabled (`addressing_mode.by_zone=1`).

From a legacy applications standpoint, this change consists in replacing the operation value from h02 to h81.

Format

```
IRD_command(){
  EMM_command                 8     uimsbf    0x64
  length                      8     uimsbf    0x0D
  command_body(){
    sequence_number           32    uimsbf
    command_id                8     uimsbf    0xC2
    operation                 8     uimsbf    0x02
    zone_id                   48    uimsbf    6-byte zone identifier
    checksum                  8     bslbf
  }
}
```

4.4 Set Macrovision CPS

Description

The Macrovision system uses a chip inside the set-top box that acts on the analog video output to prevent the recording, but not the viewing. The chip accepts configuration data and operational data. Configuration data define the different ways to mess up the signal: how long to rotate the colors, how high is the peak in the signal, and so on. This is a 136-bit string called CPS by Macrovision. Operational data tell us which way has to be applied: turn color stripe on, turn v sync off, and so on. This is an 8-bit string called the Mode byte by Macrovision.

The purpose of this IRD command is to provide the CPS string to a set-top box, in order to parameter its Macrovision chip. The Mode byte is not transmitted through this command and will be part of a private descriptor present in the EIT.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC4
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
    for (i=0; i<N; i++){
      cps                        8  uimsbf    CPS string
    }
  }
  checksum                     8  uimsbf
}
```

Payload Semantic

cps	CPS (Copy Protection Setup) string defined by Macrovision. The actual Macrovision chip expects this string to be 136 bits long (17 bytes). However, the current specification defines it of variable length in order to support future version. The length can be deduced from global "length" field of the IRD command.

4.5 Configure STB

Description

This command allows the head-end to enable or disable features in a set-top box. Each feature is associated to a single bit set to 1 when enabled and 0 when disabled. All features are disabled by default. The features configuration has to be stored in NVM so that no information is lost after power-cycling the set-top box.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC5
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
     compatible_mode             1  bslbf
     video                       1  bslbf
     audio                       1  bslbf
     smartcard_1                 1  bslbf
     smartcard_2                 1  bslbf
     harddisk                    1  bslbf
     dvd                         1  bslbf
     serial_port_1               1  bslbf
     serial_port_2               1  bslbf
     parallel_port               1  bslbf
     usb_port                    1  bslbf
     1394_port                   1  bslbf
     spare_port_1                1  bslbf
     spare_port_2                1  bslbf
     peripheral_1                1  bslbf
    peripheral_2                 1  bslbf
     for(i=0; i<N; i++){
       pattern                   8  bslbf     optional
     }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

compatible_mode	Usually set to 1 when the set-top box is fully DVB compliant and set to 0 when the set-top box usage is restricted to a specific network only. In case the set-top box is configured for a specific network but is connected to another network, the set-top box application shall display a proper message and all features shall be disabled. Operators willing to avoid rented set-top boxes to be used in other networks could use the compatible mode.
video	Video decoding shall be disabled when set to 0.
audio	Audio decoding shall be disabled when set to 0.
smartcard_1	Smart card reader 1 shall be disabled when set to 0.
smartcard_2	Smart card reader 2 shall be disabled when set to 0.
harddisk	Hard disk shall be disabled when set to 0.
dvd	Dvd shall be disabled when set to 0.

serial_port_1	Access to serial port 1 shall be disabled when set to 0.
serial_port_2	Access to serial port 2 shall be disabled when set to 0.
parallel_port	Access to parallel port shall be disabled when set to 0.
usb_port	Access to usb port shall be disabled when set to 0.
1394_port	Access to IEEE 1394 port shall be disabled when set to 0.
spare_port_1	Access to spare port 1 shall be disabled when set to 0.
spare_port_2	Access to spare port 2 shall be disabled when set to 0.
peripheral_1	Peripheral 1 shall be disabled when set to 0.
peripheral_2	Peripheral 2 shall be disabled when set to 0.
pattern	Optional additional bit fields. Their absence shall be interpreted as value 1 by the set-top box application.

4.6 Set Network ID

Description

This command sets the set-top box network ID to a specific value. This allows the set-top box to retrieve the Network Information Table (NIT) defining the topology of a particular local area. This command can also be used to assign testing network ID to specific set-top boxes.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC6
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
    network_id                  16  uimsbf    Network ID
    original_network_id         16  uimsbf    Original network ID
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

network_id	Unique identifier indicating the network ID.
original_network_id	Unique identifier indicating the original network ID.

4.7 Master/Slave

Refer to document [2] for a Master/Slave feature solution overview and document [3] for implementation guidelines.

4.7.1 Continuous Mode Initialization

Description

This command is used to set the parameters in order to initialise the Master/Slave continuous mode.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC7
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
    masterSmartcard             32  uimsbf
    validationPeriod             8  uimsbf    in days
     randomPeriod                8  uimsbf    in days
    timeout                      8  uimsbf    in hours
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`masterSmartcard`	this is the Smart card ID of the master Smart card without checksum.
`validationPeriod`	this value define the average time, expressed in days, between two validation procedures.
`randomPeriod`	the next validation procedure will occur in `validationPeriod` days +/- `randomPeriod` days. The targeted day will be randomly chosen in this bracket of time.
`timeout`	the timeout is the period of time during which the customer has to succeed with the validation procedure (insert the master Smart card in the slave STB). At the end of the `timeout` period, the STB will stop playing video and/or audio signal.

4.7.2 Cancellation

Description

This command id is used to disable the IRD Master/Slave mode continuous and single shot mode.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC7
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h02
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

4.7.3 Single Shot

Description

This command is used to set the parameters in order to initialise the single shot Master/Slave command. It is not possible to disable this command only; all Master/Slave modes must be disabled in order to cancel it. In other words, it's not possible to cancel a single shot command without cancelling the continuous mode.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC7
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h03
  }
  addressing_info()           0..n  uimsbf
  payload{
    masterSmartcard             32  uimsbf
    timeout                      8  uimsbf    in hours
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`masterSmartcard` — this is the Smart card ID of the master Smart card without checksum.

`timeout` — the timeout is the period of time within which the customer has to succeed with the validation procedure (insert the master Smart card in the slave STB). At the end of the `timeout` period, the STB will stop playing video and/or audio signal.

4.7.4 Automatic Master/Slave

Description

This command is used to set the parameters in order to initialise the automatic Master/Slave feature.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC7
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h04
  }
  addressing_info()          0..n  uimsbf
  payload{
    stbMode                      2  bslbf     0 master
                                               1 slave
                                               2 stand-alone
                                               3 reserved
    reserved                     6  bslbf     all bits set to 1
    masterSmartcard             32  uimsbf
    timeout                     16  uimsbf    in seconds
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

stbMode	This is the mode in which the STB is running
masterSmartcard	This is the Smart card ID of the master Smart card without checksum.
timeout	This is the period of time during which a slave STB can run without getting any data from a master STB.

4.8 Set PIN Code

Description

This command allows the head-end to change the set-top box PIN code. The operation field identifies the PIN code that has to be modified in case the set-top box manages several PIN codes.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC8
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01..h04
  }
  addressing_info()           0..n  uimsbf
  payload{
    pin_length                   8  uimsbf    PIN length
    for(i=0; i<N; i++){
      character                  8  uimsbf    PIN character
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

pin_length	Number of bytes the PIN code is composed of.
character	ASCII code of each character composing the PIN code.

Example

The following command will change the PIN code number 1 to "1234".

```
IRD_command = `640C00000007C801043132333469
```

In this example the 4-byte sequence number is equal to `00000007.

4.9 Force Stand-by

4.9.1 Standard

Description

This command allows the head-end to force a set-top box to enter in the stand-by mode. It could be an indirect way to force a set-top box to get a software download. Indeed, in most set-top boxes the download process is triggered by entering the standby mode.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC9
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()           0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

Example 1

Standard force stand-by:

```
IRD_command = `64 07 00001630 C9 01 36
```

Example 2

Force stand-by of set-top boxes belonging to the zone 123:

```
IRD_command = `64 0D 00001630 C9 81 00000000007B 3B
```

4.9.2 By Zone ID (deprecated)

Description

This command is deprecated. It is replaced by the standard “Force Stand-by” command with the zone addressing mode enabled (`addressing_mode.by_zone=1`).

From a legacy applications standpoint, this change consists in replacing the operation value from h02 to h81.

Format

```
IRD_command(){
  EMM_command                 8     uimsbf    0x64
  length                      8     uimsbf    0x0D
  command_body(){
    sequence_number           32    uimsbf
    command_id                8     uimsbf    0xC9
    operation                 8     uimsbf    0x02
    zone_id                   48    uimsbf    6-byte zone identifier
    checksum                  8     bslbf
  }
}
```

4.10 Configure Camlock

Description

This command allows the head-end to enable or disable the camlock feature. Setting the operation field to 0 disables the camlock feature, while setting that field to 1 enables this feature.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCA
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00..h01
  }
  addressing_info()           0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

operation	0x00 disable camlock
	0x01 enable camlock

4.11 Copy Protection

For further enlightenment on the subject of Copy Protection, please refer to document [4].

4.11.1 Validate POD_ID/Host_ID

Description

This command allows the head-end to validate a POD_ID and Host_ID couple, according to their absence from head-end managed CRLs. The operation field identifies that command as a validation command.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCB
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload{
    POD_ID                      64  uimsbf    Validated POD ID
    Host_ID                     40  uimsbf    Validated Host_ID
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

POD_ID	8 bytes value characterizing a valid POD_ID.
Host_ID	5 bytes value characterizing a valid Host_ID.

Example

The following command will validate a POD_ID/Host_ID couple of 0x0102030405060708/0x0102030405.

```
IRD_command = `641400000011CB0001020304050607080102030405 02
```

In this example the 4-byte sequence number is equal to `00000011.

4.11.2 Revoke POD_ID/Host_ID

Description

This command allows the head-end to revoke a POD_ID and Host_ID couple, according to their presence into head-end managed CRLs. The operation field identifies that command as a revocation command.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCB
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()           0..n  uimsbf
  payload{
    POD_ID                      64  uimsbf    Validated POD ID
    Host_ID                     40  uimsbf    Validated Host_ID
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

POD_ID	64 bits value characterizing a valid POD_ID.
Host_ID	40 bits value characterizing a valid Host_ID.

Example

The following command will revoke a POD_ID/Host_ID couple of 0x0102030405060708/0x0102030405.

```
IRD_command = `641400000013CB0101020304050607080102030405 01
```

In this example the 4-byte sequence number is equal to `00000013.

4.11.3 Force Authentication

Description

This command allows the head-end to force a POD_ID and Host_ID couple to restart the copy protection authentication process from beginning, as if inserted for the first time.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCB
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h02
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None.

4.11.4 Set Key Session Period

Description

This command allows the head-end to set a key session period for a given POD/SC.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCB
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h03
  }
  addressing_info()          0..n  uimsbf
  payload{
    key_session_period          16  uimsbf
  }
  checksum                      8  uimsbf
}
```

Payload Semantic

key_session_period	16 bits value giving the session key refresh time with a resolution of 10 second. Null means unlimited.

Example

The following command will set a key session period of 120 seconds.

```
IRD_command = `640900000017CB03000C26
```

In this example the 4-byte sequence number is equal to `00000017.

4.12 Restore Factory Settings

Description

This command allows the head-end to restore the set-top box's factory settings. Settings affected by this command are set-top box dependent. For instance, the favourite channel list and password may be cleared, and tuner settings reset to default values.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCC
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

4.13 Force Tuning

4.13.1 Force Tune

Description

This command forces the STB to tune to a service defined by the network_id/transport_id/service_id. If the STB is able to query the access rights needed for the service, then the tuning should occur only if the subscriber has access to the service.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hC1
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
    network_id                  16  uimsbf
    transport_id                16  uimsbf
    service_id                  16  uimsbf
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

network_id	corresponds to the network_id as described in the DVB Network Information Table (NIT).
transport_id	corresponds to the network_id as described in the DVB Network Information Table (NIT).
service_id	corresponds to the service_id as described in the DVB Service Description Table (SDT). It may also correspond to the program number found in the MPEG Program Map Table (PMT).

4.13.2 Force Tune with Timeout

Description

This command forces the STB to tune to a service defined by the network_id / transport_id / service_id for a defined duration (in seconds). If the STB is able to query the access rights needed for the service, then the tuning should occur only if the subscriber has access to the service. After the defined duration the STB shall tune back to the last previously watched service.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCD
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
    network_id                  16  uimsbf
    transport_id                16  uimsbf
    service_id                  16  uimsbf
    timeout                     16  uimsbf
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

network_id	corresponds to the network_id as described in the DVB Network Information Table (NIT).
transport_id	corresponds to the transport_id as described in the DVB Network Information Table (NIT).
service_id	corresponds to the service_id as described in the DVB Service Description Table (SDT). It may also correspond to the program number found in the MPEG Program Map Table (PMT).
timeout	Duration in seconds the STB has to remain tuned to the specified service. If the timeout is set to 0, the STB has to remain tuned forever.

Example 1

Forces the set-top box to tune to the service identified by the triplet (100, 2, 33), with a timeout of 2 minutes:

```
IRD_command = `64 0E 00001630 CD 01 0064 0002 0021 78 36
```

Example 2

Same examples for set-top boxes belonging to the zone 123:

```
IRD_command = `64 14 00001630 CD 81 00000000007B 0064 0002 0021 78 3B
```

4.13.3 Force Tune by Zone ID (deprecated)

Description

This command is deprecated. It is replaced by the standard "Force Tune" command (`command_id=hCD`, `operation=h01`) with the zone addressing mode enabled (`addressing_mode.by_zone=1`).

From a legacy applications standpoint, this change consists in replacing the operation value from h02 to h81 and moving the 6-byte zone ID at the beginning of the data field.

Format

```
IRD_command(){
  EMM_command                 8     uimsbf    0x64
  length                      8     uimsbf    0x15
  command_body(){
    sequence_number           32    uimsbf
    command_id                8     uimsbf    0xCD
    operation                 8     uimsbf    0x02
    data{
      network_id              16    uimsbf
      transport_id            16    uimsbf
      service_id              16    uimsbf
      timeout                 16    uimsbf
      zone_id                 48    uimsbf    6-byte zone identifier
    }
    checksum                  8     bslbf
  }
}
```

4.13.4 IPTV – Force Tune with Timeout

Description

This command forces an IPTV STB to tune to a service identified by an IGMP multicast address for a defined duration (in seconds). If the STB is able to query the access rights needed for the service, then the tuning should occur only if the subscriber has access to the service. After the defined duration the STB shall tune back to the last previously watched service.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCD
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h03
  }
  addressing_info()          0..n  uimsbf
  payload{
    ip_address                  32  uimsbf
    port_number                 16  uimsbf
    timeout                     16  uimsbf
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

ip_address	IGMP multicast IP address the STB has to get connected to.
port_number	IGMP port number
timeout	Duration in seconds the STB has to remain tuned to the specified service. If the timeout is set to 0, the STB has to remain tuned forever.

4.13.5 IPTV – Force Tune by Zone ID (deprecated)

Description

This command is deprecated. It is replaced by the "IPTV – Force Tune with Timeout" command (`command_id=hCD`, `operation=h03`) with the zone addressing mode enabled (`addressing_mode.by_zone=1`).

From a legacy applications standpoint, this change consists in replacing the operation value from h04 to h83 and moving the 6-byte zone ID at the beginning of the data field.

Format

```
IRD_command(){
  EMM_command                 8     uimsbf    0x64
  length                      8     uimsbf    0x15
  command_body(){
    sequence_number           32    uimsbf
    command_id                8     uimsbf    0xCD
    operation                 8     uimsbf    0x04
    data{
      ip_address              32    uimsbf
      port_number             16    uimsbf
      timeout                 16    uimsbf
      zone_id                 48    uimsbf    6-byte zone identifier
    }
    checksum                  8     bslbf
  }
}
```

4.14 Pop-up

4.14.1 Display Pop-Up

Description

This command allows the set-top box to display pop-up messages.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCF
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()          0..n  uimsbf
  payload{
    popup_id                    10  bslbf     Pop-up identifier
    total_segment                6  bslbf     Total number of segments
    persistence                  2  bslbf     0 normal
                                               1 timeout
                                               2 user acknowledged
                                               3 reserved
    segment_number               6  bslbf
    for (i=0; i<N; i++){
      message                    8  bslbf     Pop-up message body
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`popup_id` — Unique pop-up identifier

`total_segment` — Total number of segments required to carry the whole message. It's a 6-bit variable covering the range [1..63]. Each segment may carry up to 45 bytes.

`persistence` — Gives some information about the pop-up behavior:

- 0 Pop-up remains displayed until it is replaced by another one or removed by the "Remove Pop-Up" command defined in 4.14.2.
- 1 Pop-up automatically disappears after a while. The duration of the timeout is free, but should not be shorter than 10s.
- 2 Pop-up remains displayed until the user's acknowledgement (by pressing any key)

`segment_number` — Identifies the current segment. The first segment is equal to 0 and the last segment is equal to total_segment-1.

Notes

1. If the total length of a pop-up message is larger than 45 bytes, then it is split in several segments, each having the same pop-up identifier and consecutive segment numbers. As there are at the most 63 segments of 45 bytes per message, the maximum length of a message is equal to 2835 bytes.

2. The channel change is allowed during the display of a normal or a timeout pop-up, provided it remains displayed over the video stream. On the contrary, a user acknowledged pop-up may be removed by a channel change.
3. If a new pop-up (new popup_id) is received during the display of another pop-up, this latter one shall be replaced at once with the new one, including the persistence parameter. This means for instance that a normal pop-up may be replaced by a timeout pop-up if desired.

Example

1 segment pop-up message with the following parameters:

popup_id : 4
total_segment : 1
persistence: 0 (normal)
message : "Pay your bill!"

command : `641800000017CF00010100506179207 96F75722062696C6C2132

4.14.2 Remove Pop-Up

Description

This command removes any kind of pop-up displayed through the "Display Pop-Up" command. It is useful in case the head-end decides to remove a persistent pop-up.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hCF
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

4.15 MovieKey

Description

This command permits to send the MovieKey needed that allows decoding an asset (i.e. watch movie), identified by its `asset_id`. **This command is intended to the CDE only.** The MovieKey will be exported by the CDE to any decoder application that has previously registered to `caCdeRegisterIrdMovieKeyExportation()`, but the IRD-command itself is NOT exported to the decoder application

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD0
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload{
    moviekey_id                 32  uimsbf
    end_of_validity             32  uimsbf    Validity date of the MovieKey
    total_segments               8  bslbf
    segment_index                8  bslbf
    for(i=0; i<N; i++) {                      (Nmax=54)
      asset_MKey                 8  bslbf
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`moviekey_id`	Unique identification of the MovieKey.
`end_of_validity`	Date indicating the end of validity of the MovieKey, in UTC. The date is coded in unix date (number of seconds since 1st of January 1970, at 00:00:00). Maximum value is 7th of February 2106, 06:28:15
`total_segments`	Total amount of segments composing the MovieKey
`segment_index`	Identifies the current segment. The first segment is equal to 0 and the last segment is equal to `total_segments-1`.
`asset_MKey`	Bytes composing the MovieKey.

Example

The following commands will send a MovieKey related to an asset. The parameters are:

- `asset_id = '12345678`
- `end_of_validity` = 11th of June 2004, 14:20:00 = 1'086'963'600 = `'40C9BF90`
- MovieKey = `'04650101820098F5A0AB56D70242F8BB694B3B8724DE65D745F5AD7A13A405F37473CFE915A4DC6B3237D45F738001DA4403AF9918E8C6000D87DCF9122EE1FC03F90C02F0AC206DC986A66801DAE10542D6491FB75E081D5BA35D98C55347A8BBA8BE08EA5858`
- Total size of the MovieKey = 103 bytes ➔ will be split into two (`'02`) IRD commands (63+40).

- The first segment_index = '00
- The second segment_index = '01
- First 4-byte sequence number = `00000017
- Second 4-byte sequence number = `00000018

1. IRD_command =
   `644700000017D00012345678 40C9BF9002 00 04650101820098F5A0AB56D70242F8BB694B3B8724DE65D745F5AD7A13A405F37473CFE915A4DC6B3237D45F738001DA4403AF9918E8C6000D8761
2. IRD_command =
   '643A00000018D00012345678 40C9BF9002 01 CDF9122EE1FC03F90C02F0AC206DC986A66801DAE10542D6491FB75E081D5BA35D98C55347A8BBA8BE08EA585860

4.16 Push-VOD

4.16.1 Content Download and Playback Configuration

Description

This command allows the Head-End to enable or disable Push-VOD content download and/or playback features in a Set-Top Box. Each feature is associated to a single bit set to 1 when enabled and to 0 when disabled.

All features are enabled by default. The features configuration has to be stored in NVM so that no information is lost after power-cycling the set-top box.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD1
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload{
    content_download             1  bslbf     1 enable content download
                                               0 disable content download
    content playback             1  bslbf     1 enable content playback
                                               0 disable content playback
    reserved                     6  bslbf     Always set to '11 1111'
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`content_download` — This bit enables (1) or disables (0) the Push-VOD content download. If set to 1, the decoder application may open DIL download sessions (refer to [5]). If set to 0, the decoder application shall immediately close all DIL download sessions. Moreover, the decoder application shall not open any new download session before the Head-End enables it through a new command.

`content_playback` — This bit enables (1) or disables (0) the Push-VOD content playback. If set to 1, the decoder application is allowed to playback any Push-VOD content already on the HDD. If set to 0, the decoder application shall immediately stop any Push-VOD content playback and shall not start any new playback before the Head-End enables it through a new command.

Example

The following command disables Push-VOD content download, but enables playback of already downloaded content:

IRD-Command: '640800000004D1007FB0

with: 7F = '0111 1111'

- 0 disables new content download
- 1 enables playback of already downloaded content

4.16.2 Push-VOD Partition Formatting

Description

This command forces the STB to format the HDD partition(s) containing the Push-VOD content (assets and metadata files).

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD1
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

4.16.3 Erase Asset

Description

This command forces the STB to erase a Push-VOD asset identified by its unique `asset_id`, if it has already been downloaded by the DIL.

In addition, the decoder application shall set the lowest download priority for that asset through the DIL. It shall therefore call:

```
dilSetAssetPriority(asset_id, DIL_PRIORITY_IGNORE_ASSET);
```

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD1
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h02
  }
  addressing_info()          0..n  uimsbf
  payload{
    asset_id                   32  uimsbf    Unique asset identifier
  }
  checksum                     8  uimsbf
}
```

Payload Semantic

`asset_id`	This is the asset identifier, which is unique over the complete CAS.

4.16.4 Erase Metadata File

Description

This command forces the STB to erase a particular file on the HDD, if it has already been downloaded by the DIL.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD1
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h03
  }
  addressing_info()          0..n  uimsbf
  payload{
    for(i=0; i<N; i++){
      filename_char              8  uimsbf    Characters composing the filename
                                               to erase
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`filename_char`	Characters composing the filename of the file to erase on the STB HDD, relative to the `pxMetadataPath` the decoder application gave as initialisation parameter to the DIL.

Example

The following command requests the decoder application to erase a file named "A/0000022F/0000022F.pmt-01".

Ird-Command =
'641E00000005D103412f3030303030303232462f3030303030303232462e706d742d303134C

4.16.5 Set Downloads Wake-Up

Description

This command defines a certain number of time slots (maximum 17) during which the decoder shall be awake (and downloading).

The decoder shall store the time slots settings in NVM so that no information is lost after power-cycling the set-top box.

Each command resets the settings of the previous command. If the operator wants to clear all time slots (the STB shall never awake itself), it can send a command without time slot.

The default setting is no time slot.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD1
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h04
  }
  addressing_info()          0..n  uimsbf
  payload{
    for(i=0; i<N; i++){                        (N_max = 17)
      reserved                   2             Always 0x00
      start_day_of_week          3  uimsbf    Weekdays
      start_minutes             11  uimsbf    Max value = 0x05A0 (1440 min/day)
      reserved                   2             Always 0x00
      stop_day_of_week           3  uimsbf    Weekdays
      stop_minutes              11  uimsbf    Max value = 0x05A0 (1440 min/day)
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`start_day_of_week`	This value defines the day of the week the Nth time slot begins. If its value is `0x00`, the time slot is valid for all weekdays. Else, it corresponds to the weekday number (`0x01` being Monday and `0x07` being Sunday).
`start_minutes`	Beginning of the Nth time slot, defined as the number of minutes since midnight (00:00).
`stop_day_of_week`	This value defines the day of the week the Nth time slot ends. If its value is `0x00`, the time slot is valid for all weekdays. Else, it corresponds to the weekday number (`0x01` being Monday and `0x07` being Sunday).
`stop_minutes`	End of the Nth time slot, defined as the number of minutes since midnight (00:00).

Notes

1. If the operator wants to reset all time-slots, it can send a command with N=0.
2. The default settings is no time slot (by default, the decoder application never awakes itself)

Example

Let's consider following time slots:

Mon 02:00am – Mon 05:00am
Wed 01:00am – Wed 05:00am
Thu 10:30pm – Fri 03:00am
Everyday from 06:00am to 08:00am

These time-slots are to be translated like this:

Slot		Weekday		Time				Value
• Slot 1	start	Mon	001	02:00am	120min	000 0111 1000	→	0x0878
	stop	Mon	001	05:00am	300min	001 0010 1100	→	0x092C
• Slot 2	start	Wed	011	01:00am	60min	000 0011 1100	→	0x183C
	stop	Wed	011	05:00am	300min	001 0010 1100	→	0x192C
• Slot 3	start	Thu	100	10:30pm	1'350min	101 0100 0110	→	0x1D46
	stop	Fri	101	03:00am	180min	000 1011 0100	→	0x28B4
• Slot 4	start	All	000	06:00am	360min	001 0110 1000	→	0x0168
	stop	All	000	08:00am	480min	001 1110 0000	→	0x01E0

So the resulting IRD-Command would be:

```
'641B00000007D1040878092C183C192C1D4628B4016801E054
```

4.17 Force Software Download

4.17.1 Standard

Description

This command allows the head-end to ask the set-top box to check whether a download stream is available and performs the software update if necessary.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD2
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()          0..n  uimsbf
  payload{
    for(i=0; i<N; i++){
      version_number             8  uimsbf    Version
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`version_number`	String containing a version number. This version could be used by the application to know if a software update is necessary. This string is optional and its format is manufacturer dependent.

Example

The following command forces a software download without specifying any version number:

```
IRD_command = `640700000018D2002E
```

In this example the 4-byte sequence number is equal to `` `00000018 ``.

4.17.2 By Download ID

Description

This command allows the head-end to ask the set-top box to check whether a download stream is available and performs the software update if necessary.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD2
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
    unique_download_id          16  uimsbf
  }
  checksum                      8  uimsbf
}
```

Payload Semantic

unique_download_id	This 16-bit field uniquely identifies a software upgrade across the whole system. This ID can be used in order to retrieve the download stream signalling information that matches the set-top box platform.

4.18 Change Usage ID

4.18.1 Resident software

Description

This command allows the head-end to change the set-top box usage ID. All set-top boxes programmed with the same usage ID are related to the same download stream. For instance, field test set-top boxes may be assigned a usage ID that differs from production set-top boxes in order to be upgraded independently. This command concerns resident software embedded in the box. Refer to the next command for downloadable application.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD3
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()          0..n  uimsbf
  payload{
    usage_id                     8  uimsbf    Usage ID
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

usage_id	8-bit identifier used to create groups of set-top boxes that are assigned to the same download stream.

4.18.2 Downloadable applications

Description

This command is similar to the previous one. The only difference is that it targets downloadable applications instead of the resident software.

This command can be used to define which boxes participate to a field trial of a new downloadable application. If applications are signalled by means of linkage descriptors, the value of the usage ID can be used to select the right linkage descriptor.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD3
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  payload{
    usage_id                     8  uimsbf    Usage ID
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

usage_id	8-bit identifier used to create groups of set-top boxes that are related to the same downloadable application.

4.19 Set Community Type

Description

This command allows the head-end to change the set-top box community type that is used to customize the behavior of the set-top box application.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD4
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload{
    community_type               8  uimsbf    Community type
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`community_type`

8-bit identifier corresponding to the set-top box community type. A community type of 0 means that the set-top box is not community specific and has the standard behavior. Set-top boxes belonging to a community are associated a non-null community type.

4.20 Format Logical Disk

Description

This command allows the head-end to format one or several logical disks of a set-top box.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD5
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()          0..n  uimsbf
  payload{
    for(i=0; i<N; i++){
      logical_disk_id            8  uimsbf    Identifier of a logical disk
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`logical_disk_id`

8-bit identifier corresponding to the identifier of a logical disk. If no logical disk ID is specified in the command, it means that all disks have to be formatted.

Disk ID values are operator dependent but have to be common to all set-top boxes deployed on the same network.

Examples

The first command formats the disks identified by the ID 1 and 3. The second one formats all the disks.

```
IRD_command = `64 09 00005824 D5 00 0103 27
IRD_command = `64 07 00005825 D5 00 2B
```

4.21 Usage Monitoring

Description

This command allows the head-end to configure and activate the usage monitoring feature.

If an IRD command contains only a subset of the possible parameters, the ones that are not included will remain unchanged.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD6
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload{
    for(i=0; i<n; i++){
      tag                        8  uimsbf    Tag of the parameter
      length                     8  uimsbf    Length of the data
      value                      N  uimsbf    Value to set
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`tag` — 8-bit identifier corresponding to the parameter to modify.

`length` — 8-bit identifier corresponding to the length of the parameter to modify.

`value` — length-bytes corresponding to the value to set for the parameter.

Tag definition

Parameter	Tag	Length	Value
Activation flag	0x01	1	Boolean indicating whether the Usage Monitoring must be activated or not.
IP Server Address + Port	0x02	6 (4 + 2)	The server to connect to in order to report usage. (IP Address on 4 bytes + port on 2 bytes MSBF)
Trigger size	0x03	4	Integer indicating the minimum amount of data (in bytes, MSBF) that must be collected before triggering a report. A value of 0 disables the automatic report.
Regular Report Frequency	0x04	1	Integer indicating the frequency in hours of the reporting. A value of 0 disables the automatic report.
Regular Report Time Range	0x05	2	Integer pair defining the start and end time (in hours over 24 hours) between which the reporting can take place.
Data Encryption Key	0x06	2 + N	Key to use to encrypt the report (Key Id

			(MSBF) + Key).
Immediate Report	0x07	0	Trigger an immediate reporting.
Minimal Report Duration	0x08	1	Minimal duration in minutes that the user must watch a service to be logged in the report.

Examples

Command to activate to usage monitoring.

```
IRD_command = `64 0A 00000511 D6 00 010101 27
```

Command to set the reporting between 01:00 and 05:00.

```
IRD_command = `64 0B 00000512 D6 00 05020105 1D
```

Command to activate the monitoring, set the reporting frequency to 8 hours and set the address of a server (193.169.0.12 port 0080).

```
IRD_command = `64 15 00000513 D6 00 010101 040108 0206C1A9000C0050 4C
```

4.22 Broadcast Network Operator Lock

Description

This command allows the head-end to configure the list of provider Id allowed.

If an IRD command contains only a subset of the possible parameters, the ones that are not included will remain unchanged.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD7
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload{
    for(i=0; i<n; i++){
      tag                        8  uimsbf    Tag of the parameter
      length                     8  uimsbf    Length of the data
      value                      N  uimsbf    Value to set
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

`tag`	8-bit identifier corresponding to the parameter to modify.
`length`	8-bit identifier corresponding to the length of the parameter to modify.
`value`	length-bytes corresponding to the value to set for the parameter.

Tag definition

Parameter	Tag	Length	Value
Activation flag	0x01	1	Boolean indicating whether the Broacast Network Operator Lock must be activated or not.
Provider Id List	0x02	2*nbr of providerId	The list of provider Id allowed (2 bytes provider Id, MSBF).

Examples

Command to activate the Broadcast Network Operator Lock and to allow the following provider Id's 0x4302, 0x4304 and 0x4503.

```
IRD_command = `64 12 00000321 D7 00 010101 0206430243044503 4A
```

4.23 Zone ID

4.23.1 Set Zone ID

Description

This command allows the head-end to assign a zone to a set-top box. The zone identifier is stored in the set-top box. Its format is customer dependent. This zone has nothing to do with the smart card's zip code. The set-top box application can then rely on this zone in order to run specific operations.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD8
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload{
    zone_id                     48  uimsbf    6-byte zone identifier
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

zone_id	6-byte zone identifier to be assigned to the set-top box. Its format is customer specific.

Example

This command sets the zone ID 3:

```
IRD_command = `64 0D 00000124 D8 00 000000000003 25
```

4.23.2 Cancel Zone ID

Description

This command allows the head-end to cancel the zone ID of a set-top box. As a result, this set-top box will no longer process commands associated to a zone ID (e.g. Force tune by zone ID).

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD8
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

Example

This command cancels the zone ID:

```
IRD_command = `64 0D 00000125 D8 01 27
```

4.24 EMM Wake-Up Management

4.24.1 Introduction

The following sections define the IRD commands related to the EMM wake-up management. Refer to [6] for a detailed specification of the set-top box behavior.

4.24.2 Disabling the feature

Description

This command disables the EMM wake-up management feature. Upon such a command, the set-top box shall remove all parameters (default ones included) related to this feature and no longer wakes up for EMM filtering. The feature can be enabled again by setting new parameters.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hD9
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()          0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

Example

```
IRD_command = `64 07 00000125 D9 00 27
```

4.24.3 Setting new parameters

Description

This command is used to provide the set-top box with the parameters required by the EMM wake-up management. These parameters consist in a set of (DVB triplet, waking duration) couples indicating the services the set-top box has to tune to and how long in order to acquire EMMs, as well as a sleeping duration used to compute the next wake-up time. The command may also define an absolute wake-up time.

The set-top box shall store these parameters in persistent memory, including the sequence number, so that no information is lost when the box is powered off.

The sequence number changes every time new parameters are set in the IRD command. The set-top box shall then process the new command and store the new parameters. Each new command overwrites the complete set of parameters previously stored, including default parameters.

The set-top box shall be programmed with a default set of parameters applying until the first command is received.

Refer to [6] for further information about the behaviour of the set-top box.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf   h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf   hD9
    addressing_mode()            2  bslbf
    operation                    6  uimsbf   h01
  }
  addressing_info()            0..n  uimsbf
  payload{
    waking_loop_count            8  uimsbf   Number of entries in the loop
    for(i=0 ;i<N ;i++){
      reserved                   5  bslbf    b00000
      network_flag               1  bslbf    1 if network info is present
      transport_flag             1  bslbf    1 if transport info is present
      service_flag               1  bslbf    1 if service info is present
      if(network_flag){
        network_id              16  uimsbf   Network ID
      }
      if(transport_flag){
        transport_stream_id     16  uimsbf   Transport stream ID
      }
      if(service_flag){
        service_id              16  uimsbf   Service ID
      }
      waking_duration           16  uimsbf   Duration in minutes
    }
    reserved                     6  bslbf    b000000
    wake_up_time_flag            1  bslbf    1 if wake-up time is present
    sleeping_duration_flag       1  bslbf    1 if sleeping duration is
                                             present
    if(wake_up_time_flag){
      wake_up_year               8  uimsbf   Years since 1900
      wake_up_month              8  uimsbf   Month of the year (1..12)
      wake_up_day                8  uimsbf   Day of the month (1..31)
      wake_up_hour               8  uimsbf   Hours since midnight (0..23)
      wake_up_minute             8  uimsbf   Minutes after the hour (0..59)
    }
    if(sleeping_duration_flag){
      sleeping_duration         16  uimsbf   Duration in minutes
    }
  }
```

```
    checksum                    8  uimsbf
}
```

Payload Semantic

waking_loop_count	Number of entries in the loop.
network_flag	When set, this 1-bit field indicates the presence of the network information.
transport_flag	When set, this 1-bit field indicates the presence of the transport information.
service_flag	When set, this 1-bit field indicates the presence of the service information.
network_id	This 16-bit field carries the DVB network ID the STB shall tune to. It is optional. If not present, the TS ID and service ID below are not present either and the STB shall tune to the last service tuned before stand-by, or the default service.
transport_stream_id	This 16-bit field indicates the DVB transport stream the STB shall tune to. It is optional. If not present, the service ID below is not present either and the STB shall tune to any service of any transport stream of the network signaled by the network ID above.
service_id	This 16-bit field indicates the DVB service ID the STB shall tune to. It is optional. If not present, the STB shall tune to any service of the TS signaled by the transport stream ID above.
waking_duration	This 16-bit field indicates the number of minutes the STB shall keep tuned to a given service.
wake_up_time_flag	When set, this 1-bit field indicates the presence of an absolute wake-up time.
sleeping_duration_flag	When set, this 1-bit field indicates the presence of a sleeping duration.
wake_up_year	This 8-bit field is part of the absolute wake-up time and carries a number of years since 1900.
wake_up_month	This 8-bit field is part of the absolute wake-up time and indicates the month of the year (1..12).
wake_up_day	This 8-bit field is part of the absolute wake-up time and indicates the day of the month (1..31).
wake_up_hour	This 8-bit field is part of the absolute wake-up time and indicates the number of hours since midnight (0..23).
wake_up_minute	This 8-bit field is part of the absolute wake-up time and indicates the number of minutes after the hour (0..59).
sleeping_duration	This 16-bit field is used to compute the next wake-up time. It indicates the number of minutes after which the STB shall wake-up.

Example 1

The set-top box wakes up each hour and tunes to the last service for 5 minutes followed by the service identified by the DVB triplet (nid=200, tsid=2, sid=6) for 10 minutes.

- (wd=5 min)
- (nid=200, tsid=2, sid=6, waking_duration=10 min)
- sleeping_duration=45 min

```
IRD_command = `64 14 00000126 D9 01 02 00 0005 07 C8 02 06 000A 01 002D 10
```

Example 2

The set-top box wakes up on 16-Feb-2006 at 04:30 and tunes for one hour to the any service of any transport stream of the network ID 33.

- (nid=33 waking_duration=60 min)
- wake_up_time= 16-Feb-2006 04:30

```
IRD_command = `64 12 00000127 D9 01 01 04 21 003C 02 6A0210041E 24
```

4.25 Update Cohabitation Tables

Description

This command allows the head-end to update the ECM, EMM and IEMM cohabitation tables of the set-top box. Each table can contain several CAS IDs. Order of CAS IDs within a table shall be chosen so that an entry has precedence over the following one (descending priority).

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hDA
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  payload(){
    reserved                     5  bslbf     b00000
    ecm_table_flag               1  bslbf
    emm_table_flag               1  bslbf
    iemm_table_flag              1  bslbf
    if(ecm_table_flag){
      ecm_cas_id_num             8  uimsbf
      for(i=0; i< ecm_cas_id_num; i++){
        reserved                 7  bslbf     b0000000
        cardless_flag            1  bslbf     b0: card-based, b1: card-less
        ecm_cas_id              16  uimsbf
      }
    }
    if(emm_table_flag){
      emm_cas_id_num             8  uimsbf
      for(i=0; i< emm_cas_id_num; i++){
        reserved                 7  bslbf     b0000000
        cardless_flag            1  bslbf     b0: card-based, b1: card-less
        emm_cas_id              16  uimsbf
      }
    }
    if(iemm_table_flag){
      iemm_cas_id_num            8  uimsbf
      for(i=0; i< iemm_cas_id_num; i++){
        reserved                 7  bslbf     b0000000
        cardless_flag            1  bslbf     b0: card-based, b1: card-less
        iemm_cas_id             16  uimsbf
      }
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

xxx_table_flag	When set to 1, this 1-bit field indicates the presence of the ECM, EMM or IEMM cohabitation table.
xxx_cas_id_num	Number of ECM, EMM or IEMM CAS IDs.
cardless_flag	When set to 1, this 1-bit field signaled a card-less CAS ID.
xxx_cas_id	ECM, EMM or IEMM CAS ID.

Notes

1. If transmitted, a table shall never be empty

2. It is possible to transmit each table in separate commands

Example

This command carries the following cohabitation tables:

Flag	ECM CAS ID
1	0x1840
0	0x1800

Flag	EMM CAS ID
0	0x1800
1	0x1840
0	0x1801

Flag	IEMM CAS ID
1	0x1840
0	0x1800

```
IRD_command = `64 20 00000130 DA 00 07 02 011840 001800 03 001800 011840
              001801 02 011840 001800 AC
```

4.26 Reboot STB

Description

This command allows the head-end to force a set-top box reboot.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hDB
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

4.27 BGA – Embedded Smart cards

4.27.1 Disable BGA

Description

This command allows the head-end to disable the BGA of a STB. Once that IRD command has been received by the STB application, the STB shall no longer communicate with the Nagravision ICC soldered on the decoder's PCB.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hDC
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()           0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

4.27.2 Enable BGA

Description

This command allows the head-end to enable the BGA of a STB. Once that IRD command has been received by the STB application, the STB can communicate with the Nagravision ICC soldered on the decoder's PCB.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    Hdc
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h01
  }
  addressing_info()           0..n  uimsbf
  checksum                     8  uimsbf
}
```

Payload Semantic

None

4.28 CAK Commands

4.28.1 Set Pairing Configuration

Description

The CAK can be configured at compile time in order to require the strong pairing mode only or to allow the establishment of a secure channel. This configuration is the default CAK pairing mode.

The “Set Pairing Configuration” command allows the head-end to change the CAK pairing mode.

The CAK does not save the new pairing mode in persistent memory, which means that the CAK switches back to the default pairing mode each time the set-top box reboots. As a result, the head-end shall keep the command on air as long as necessary.

The head-end shall indicate the CA S/N of set-top boxes that are concerned by this command. If a set-top box receives a command that does not includes its CA S/N, the command is not executed and thus the current pairing mode remains unchanged.

Syntax

```
IRD_command(){
  header(){
    tag                              8  uimsbf    h64
    length                           8  uimsbf
    sequence_number                 32  uimsbf
    command_id                       8  uimsbf    hDD
    addressing_mode()                2  bslbf
    operation                        6  uimsbf    h00
  }
  addressing_info()              0..n  uimsbf
  payload(){
    reserved                         6  bslbf     b000000
    list_flag                        1  bslbf
    range_flag                       1  bslbf
    if(list_flag){
      list_num                       8  uimsbf
      for(j=0; j<list_num; j++){
        ca_sn                       32  uimsbf
      }
    }
    if(range_flag){
      range_num                      8  uimsbf
      for(i=0; i<range_num; i++){
        ca_sn_min                   32  uimsbf
        ca_sn_max                   32  uimsbf
      }
    }
    reserved                         7  bslbf     b0000000
    strong_pairing_required          1  bslbf
  }
  checksum                           8  uimsbf
}
```

Payload Semantic

`list_flag`	When set to 1, this flag indicates the presence of a list of CA S/N
`range_flag`	When set to 1, this flag indicates the presence of a range of CA S/N
`range_num`	The command can contain several ranges of CA S/N. This 1-byte field indicates the number of ranges signaled in the command

ca_sn	This 32-bit field corresponds a CA S/N of a list
ca_sn_min	This 32-bit field corresponds to the first CA S/N of a range
ca_sn_max	This 32-bit field corresponds to the last CA S/N of a range
strong_pairing_required	When set to 1, this flag indicates that the CAK shall establish a strong pairing session in order to process ECM. When set to 0, this flag indicates that a secure channel is sufficient. It does not mean that the strong pairing is forbidden.

Example 1

The following command requests the STB identified by the CA S/N h00010000 to work in strong pairing mode:

```
IRD_command = `64 0E 00000132 DD 00 02 01 00010000 01 1E
```

Example 2

The following command requests the 256 STBs being in the CA S/N range [h00010000 – h000100FF] to work in strong pairing mode:

```
IRD_command = `64 12 00000132 DD 00 01 01 00010000 000100FF 01 1F
```

4.29 Set Callback Parameters

Description

This command defines the IP, PPP and/or phone callback parameters for a STB. Upon reception of this command, the STB shall store all parameters included in the command in persistent memory.

Although the syntax of the command defines optional parameters, a given command always contains all parameters required. Therefore, parameters that are no longer transmitted in a command shall be erased from persistent memory. In the same way, sending a command without any callback parameters is a way to erase all these parameters from persistent memory.

If the command carries connection parameters for several types of callback, it is up to the STB software to determine which parameters are required for the callback and use them appropriately.

Syntax

```
IRD_command(){
  header(){
    tag                          8  uimsbf    h64
    length                       8  uimsbf
    sequence_number             32  uimsbf
    command_id                   8  uimsbf    hDE
    addressing_mode()            2  bslbf
    operation                    6  uimsbf    h00
  }
  addressing_info()          0..n  uimsbf
  payload{
    reserved                     5  bslbf     b000000
    ip_info_flag                 1  bslbf     1 if IP callback info is present
    ppp_info_flag                1  Bclbf     1 if PPP callback info is present
    phone_info_flag              1  bslbf     1 if phone callback info is present
    if(ip_info_flag){
      ip_address                32  uimsbf
      port_number               16  uimsbf
    }
    if(ppp_info_flag){
      ppp_phone_num_length       8  uimsbf    Length of the ppp phone no. (1-16)
      ppp_phone_number           N  uimsbf    Phone number for PPP callback
      ppp_ip_address            32  uimsbf    IP address fir PPP callback
      ppp_port_number           16  uimsbf    Port for PPP callback
    }
    if(phone_info_flag){
      phone_num_length           8  uimsbf    Length of the phone no. (1-16)
      phone_number               M  uimsbf    Phone number for RAW callback
      ppp_username_length        8  uimsbf    Length of the username for login
      ppp_username               N  uimsbf    Username for PPP login
      ppp_password_length        8  uimsbf    Length of the password for login
      ppp_password               N  uimsbf    Password for PPP login
    }
  }
  checksum                       8  uimsbf
}
```

Payload Semantic

ip_info_flag	If set to 1, indicates that IP callback information is present. The IP callback information is the IP address and port.
ppp_info_flag	If set to 1, indicates that PPP callback information is present. The PPP callback information is the PPP phone number, PPP IP address and PPP port.
phone_info_flag	If set to 1, indicates that phone callback information is

	present. The phone callback information is just the phone number.
ip_address	IP address the STB has to connect to for callback.
port_number	Port number the STB uses for callback.
ppp_phone_num_length	The length of the phone number used for PPP callback. The valid values are from 1 to 16, since the maximum length of a phone number is 16 bytes.
ppp_phone_number	The phone number the STB uses for PPP callback. This phone number is specified in ASCII format.
ppp_ip_address	IP address the STB has to connect to for PPP callback.
ppp_port_number	Port number the STB uses for PPP callback.
ppp_username_length	Length of the username used to login for PPP callback. If the length is set to zero, then the ppp_username field is ignored.
ppp_username	The username used for PPP callback. The username does not have to be ASCII characters. It can be set to anything.
ppp_password_length	The length of the password used for PPP callback. If the length is set to zero, then the ppp_password field is ignored.
ppp_password	The password used for PPP callback. The username does not have to be ASCII characters. It can be set to anything.
phone_num_length	The length of the phone number used for RAW callback. The valid values are from 1 to 16, since the maximum length of a phone number is 16 bytes.
phone_number	The phone number the STB uses for RAW callback. This phone number is specified in ASCII format.

Example 1

This command sets only the phone number of the STB to be 13105551212. It is used for RAW phone callback.

```
IRD_command = `64 14 00000133 DE 00 01 0B 3133313035353531323132 EC
```

Example 2

This command sets only the IP address and port of the STB to be 123.156.123.156 and port 2110. It is used for IP callback.

```
IRD_command = `64 0E 00000134 DE 00 04 7B9C7B9C083E AA
```

Example 3

This command sets the phone number of the STB to be 13105554444 and IP address and port to be 123.156.189.123 and port 6789. It is used for PPP callback.

```
IRD_command = `64 1C 00000135 DE 00 02 0B 3133313035353534343434 7B9CBD7B1A85
00 00 F3
```

Example 4

This command sets the phone number of the STB to be 12134441212, the IP address to be 111.222.123.255, the port to be 1234, the PPP username to be Nagra1234 and the password to be NET9999, It is used for PPP callback.

```
IRD command = `64 2C 00000145 DE 00 02 0B 3132313334343431323132 6FDE7BFF04D2
09 4E6167726131323334 07 4E455439393939 C1
```

Example 5

This command just sets the username and password for the PPP callback. The username is set to FFAABBCCDD (hex) and the password is set to ABCDEF112233 (hex). Notice that the IP address and port still have to be sent since there is no way to tell if the IP address and port have been changed or not. The IP address is set to 111.222.255.255 and the port is set to 1020.

```
IRD_command = `64 1C 00000141 DE 00 02 00 6FDEFFFF03FC 05 FFAABBCCDD 06
ABCDEF112233 F1
```

Example 6

This command sets the phone number of the STB to be 13105551212, the PPP phone number to be 13105554444, the PPP IP address to be 123.156.189.123, the PPP port to be 6789, the IP address to be 123.156.123.156 and port to be 2110. It is used to support all the different STBs.

```
IRD_command = `64 2E 00000136 DE 00 07 7B9C7B9C083E 0B 3133313035353534343434
7B9CBD7B1A85 00 00 0B 31333130353535313231323132 45
```

5 Specific IRD Commands

For any specific command required by a manufacturer that doesn't belongs to the set of generic commands defined in chapter 4, the procedure here after has to be followed:

- The manufacturer has to issue a formal document specifying the format and the behavior of the specific command. The command must comply with the general format defined in §2, but is restricted to the definition of the *operation* and *data* fields:

```
IRD_command(){
  EMM_command              8
  length                   8     uimsbf
  command_body{
    sequence_number        32    uimsbf
    command_id             8     uimsbf
    operation              8     uimsbf
    for(i=0; i < N; i++){
      data                 8     bslbf
    }
    checksum               8     bslbf
  }
}
```

- The specification is provided to NagraVision for approval by sending an email to the following address:

  cak@nagra.com

- NagraVision evaluates the specification to see if it is acceptable and assigns a value to the *command_id* field. This guarantees a global consistency all over the networks and will avoid conflicts between different commands.
  NagraVision reserves the right to modify the command and move it in the set of generic commands if its usage suits a wider scope.
- In case the command remains a specific command, the manufacturer updates the specification with the *command_id* assigned by NagraVision and publishes a new version of the document.
- In case the command becomes a generic command, NagraVision updates the present document with the new command and publishes a new version.

If the request for a specific command comes from an operator instead of a manufacturer, the procedure here above remains the same, except that the specification is written by the operator. It is then provided to manufacturers providing set-top boxes over the operator network for implementation.

—— END OF DOCUMENT ——