iPod Accessory Protocol Interface Specification

Release R30

2007-10-02

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Figures and Tables

Introduction to iPod Accessory Protocol Interface Specification

This document specifies the electrical and software interfaces to both the 30-pin iPod dock connector and 9-pin Audio/Remote connectors to the Apple iPod. The iPod models covered by this specification are shown in Figure I-1 (page 17).

Figure I-1 Apple iPod models covered by this specification

The features described in this specification are supported in various versions of the third-generation iPod (the first iPod with the 30-pin connector), the iPod mini, the fourth-generation iPod (the iPod with the gray click wheel), the iPod photo, the iPod nano, the fifth-generation iPod, the second-generation iPod nano, iPod classic, iPod 3G nano, and iPod touch. To determine if a command or feature is supported in a particular iPod model or firmware release, see Table 3-2 (page 45).

The minimum iPod System Software versions supported by this specification are:

- Version 2.0 of the third-generation (3G) iPod.
- Version 3.0 of the fourth-generation (4G) iPod.
- Version 1.0 of the iPod mini, iPod photo, iPod nano, fifth-generation iPod (5G), second-generation iPod nano, iPod classic, and iPod 3G nano.
- Version 1.1 of the iPod touch.

Note: This document does not apply to the first- and second-generation iPods, nor to the iPod shuffle.

Organization of This Document

The information in this document is arranged in two parts and several appendixes:

Part I, "Hardware Description" (page 21) provides information about the physical and functional characteristics of the 30-pin connector and the 9-pin Audio/Remote connector. It includes the following chapters:

- "Connector Pin Designations" (page 23) lists the connector signals and their pin assignments for both the 30-pin connector and the 9-pin Audio/Remote connector.
- "Functional Description" (page 27) describes the functional characteristics of the 30-pin connector and the 9-pin Audio/Remote connector.

Part II, "iPod Accessory Protocol" (page 41), describes the software interface for communicating with the iPod. It includes the following chapters:

- "Protocol Features and Availability" (page 43) gives an overview of the General and accessory lingoes and their availability.
- "Protocol Transport Links" (page 47) describes the serial UART port link and the USB port link over which accessories can communicate using the iPod Accessory Protocol (iAP).
- "The Protocol Core and the General Lingo" (page 55) gives an overview of the iAP and describes the General Lingo, which all accessories must support.
- "Accessory Lingoes" (page 107) describes the various device-specific accessory lingoes that are part of the iAP and their commands.

Several appendixes provide additional information for both hardware and software designers:

- "iPod Power States and Accessory Power" (page 235), describes the requirements for using iPod accessory power.
- "iTunes Tagging" (page 239) describes the iTunes tagging feature for HD radio accessories.

- "Interfacing With the 3G iPod" (page 255) summarizes some of the model-specific design requirements for 3G iPod support.
- "Sample Accessory Circuits" (page 257) presents sample schematics for handling audio and video in iPod accessories.
- "AC Adapter Guidelines" (page 267) contains design guidelines for third-party developers of AC adapter accessories for the iPod touch.
- "Physical Dimensions and Connector Specifications" (page 269) provides a diagram of the 9-pin Audio/Remote connector.

At the end of this document are a glossary of terms and a document revision history.

See Also

For further information, refer to the latest revisions of these additional documents:

- *iPod Extended Interface Specification*, which describes the remote command and response protocol that lets the user interface of the iPod be translated into other environments.
- *IEEE 1394a Specification*
- *USB 2.0 High Speed Specification*
- *USB Device Class Definition for Audio Devices*
- *USB Device Class Definition for Audio Data Formats*
- *USB Device Class Definition for Human Interface Devices (HID)*

Hardware Description

The following chapters describe the physical characteristics of both the iPod 30-pin connector and the iPod 9-pin Audio/Remote connector. "Connector Pin Designations" (page 23) describes the connector signals and pin assignments for both connectors. "Functional Description" (page 27) describes the features and functionality provided by each connector.

Note: Only the 3G iPod, iPod mini, 4G iPod, and iPod photo have a 9-pin Audio/Remote connector.

Connector Pin Designations

This chapter describes the connector signals and pin assignments for the available iPod connectors.

The 30-Pin Connector

This section lists the connector signals and pin assignments for the 30-pin connector.

Table 1-1 30-pin connector pin assignments

Pin	Signal name	I/O	Function
1	DGND	GND	Digital ground in iPod
2	DGND	GND	Digital ground in iPod
3	TPA+	I/O	FireWire signal (**Deprecated**)
4	USB D+	I/O	USB signal
5	TPA-	I/O	FireWire signal (**Deprecated**)
6	USB D-	I/O	USB signal
7	TPB+	I/O	FireWire signal (**Deprecated**)
8	USB Vbus	I	USB power in; used to detect a USB host.
9	TPB-	I/O	FireWire signal (**Deprecated**)
10	Accessory Identify	I	See "Accessory Detect and Identify" (page 32) for details.
11	F/W PWR+	I	FireWire and charger input power (8 V to 15 V DC) (**Deprecated**)
12	F/W PWR+	I	FireWire and charger input power (8 V to 15 V DC) (**Deprecated**)

Pin	Signal name	I/O	Function
13	Accessory Power	O	3.3 V is the nominal output from the iPod. Nominal current in low power mode is 5 mA, with current limited to 100 mA in high power mode.
14	Reserved		
15	DGND	GND	Digital ground in iPod
16	DGND	GND	Digital ground in iPod
17	Reserved		
18	RX	I	iPod accessory protocol (Receive data to iPod from device)
19	TX	O	iPod accessory protocol (Transmit data from iPod to device)
20	Accessory Detect	I	See "Accessory Detect and Identify" (page 32) for details.
21	S Video Y / Component Video Pr	O	Either the luminance signal of S Video or the (Pr) signal of component video. Only on iPod models that support the appropriate video signal.
22	S Video C / Component Video Y	O	Either the chrominance signal of S Video or the luma signal of component video. Only on iPod models that support the appropriate video signal.
23	Composite Video / Component Video Pb	O	Either the composite video signal or the (Pb) signal of component video. Only on iPod models that support the appropriate video signal.
24	Remote Sense	I	See "Minimizing Crosstalk and Noise" (page 38). Remote Sense is supported only on iPods that support video output; however, it is okay to connect this pin on other iPod models.
25	LINE-IN L	I	Line level input to the iPod for the left channel.
26	LINE-IN R	I	Line level input to the iPod for the right channel.
27	LINE-OUT L	O	Line level output to the iPod for the left channel.
28	LINE-OUT R	O	Line level output to the iPod for the right channel.
29	Audio Return	—	Audio return. This is a signal and should never be grounded inside the accessory.
30	DGND	GND	Digital ground in iPod.
31	Chassis		Chassis ground for connector shell.
32	Chassis		Chassis ground for connector shell.

The 9-Pin Audio/Remote Connector

This section lists the connector signals and pin assignments for the 9-pin Audio/Remote connector. For physical connector layout and dimensions, refer to "Physical Dimensions and Connector Specifications" (page 247).

Note: Pin 2 (Headphone Detect) is not accessible on the external connector contacts. It is available only internally to the iPod.

Table 1-2 9-pin Audio/Remote connector pin assignments

Pin	Signal name	I/O	Function
1	Audio Out Left / Mono Mic In	I/O	25 mW audio out left channel. Also doubles as mono mic in (not supported on the iPod mini).
2	Headphone Detect	I	Internal switch to detect headphone plug insertion.
3	Audio Return	—	Audio return for left and right audio.
4	Audio Out Right	O	25 mW audio out right channel.
5	Composite Video	O	iPod photo only.
6	Accessory Power	O	3.3 V accessory power. Accessory power is connected in parallel to the 30-pin connector and 9-pin Audio/Remote connector. 100 mA is the maximum current that can be supplied from the iPod to the combined loads on the two ports.
7	TX	O	iPod accessory protocol (Transmit data from iPod to device)
8	RX	I	iPod accessory protocol (Receive data to iPod from device)
9	DGND	GND	Digital ground for accessory

Note: Not all iPods have a 9-pin Audio/Remote connector. See Table 2-1 (page 27) for details.

Functional Description

This chapter describes the functional characteristics of both the 30-pin connector and the 9-pin Audio/Remote connector. The following tables provide a description of the capabilities of each of the iPod models that support iAP.

Table 2-1 High-level operating features of each iPod model

Features	iPod Models									
	3G iPod	**iPod mini**	**4G iPod**	**iPod photo**	**iPod nano**	**5G iPod**	**2G nano**	**iPod classic**	**3G nano**	**iPod touch**
9-pin Audio/Remote connector	Yes	Yes	Yes	Yes	No	No	No	No	No	No
30-pin connector	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Number of serial ports	1	2	2	2	1	1	1	1	1	1
FW data	Yes	Yes	Yes	Yes	No	No	No	No	No	No
FW charging	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
USB data	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
USB charging	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S-Video / Composite video	No	No	No	Yes	No	Yes	No	Yes	Yes	Yes
Component video	No	No	No	No	No	No	No	Yes	Yes	Yes
Authentication	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
iAP over serial	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Features	iPod Models									
	3G iPod	**iPod mini**	**4G iPod**	**iPod photo**	**iPod nano**	**5G iPod**	**2G nano**	**iPod classic**	**3G nano**	**iPod touch**
iAP over USB	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Video browsing	No	No	No	No	No	Yes	No	Yes	Yes	No
Digital audio	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes

Table 2-2 Sleep features of each iPod model

Features	iPod Models									
	3G iPod	**iPod mini**	**4G iPod**	**iPod photo**	**iPod nano**	**5G iPod**	**2G nano**	**iPod classic**	**3G nano**	**iPod touch**
Deep sleep	Yes	Yes	Yes	Yes	No	No	No	No	No	No
Hibernation	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Light sleep time before deep sleep or hibernate	36 hours	36 hours	36 hours	36 hours	14 hours	14 hours	30 mins	30 mins	30 mins	N/A

Note: The 2G iPod nano, iPod classic, and 3G iPod nano enter hibernate mode after 30 minutes of inactivity; the iPod touch does not have a light sleep stage before hibernating. In hibernate mode, the iPod turns off accessory power and does not respond to commands over serial transport or USB. Self-powered accessories can wake a hibernating iPod by providing USB or FireWire power. The only other way to wake a hibernating iPod remotely is to create a transition on the ACC_DET pin from floating to ground. The ACC_DET pin must remain floating for at least 200 ms before transitioning to ground. This should be done only when transitioning from hibernate mode to active mode; toggling the ACC_DET line during normal, powered operation is not permitted.

For accessories that identify themselves as Simple Remote or Display Remote devices, the time until hibernate mode is extended from 30 minutes to 4 hours.

The iPod design is trending toward putting the iPod into hibernate mode as often as possible. Accessory developers should consider this when designing future devices.

The 30-Pin Connector

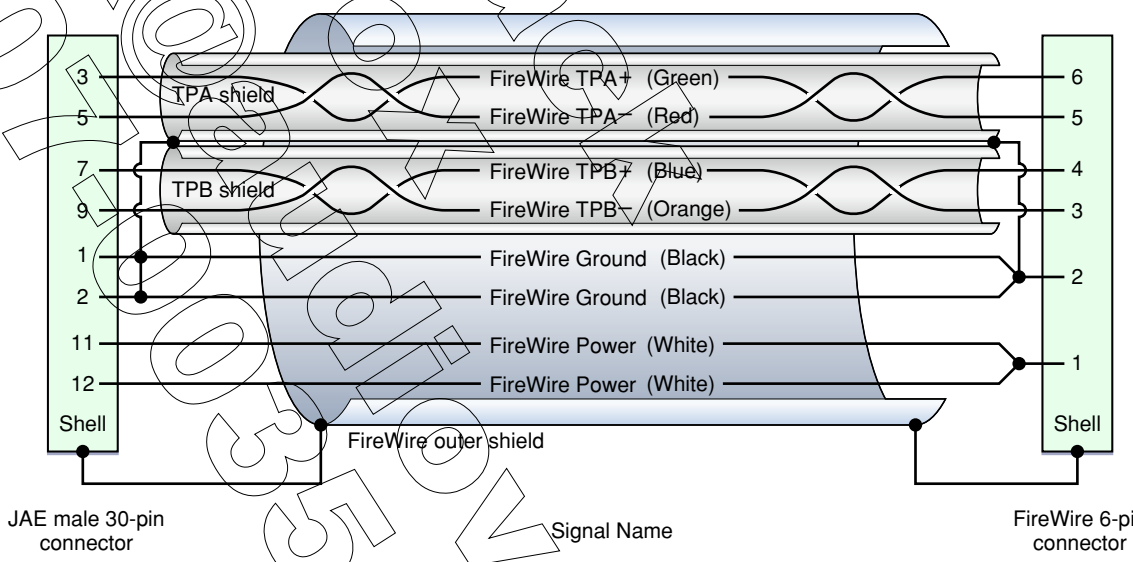
This section describes the features and physical characteristics of the 30-pin connector.

FireWire

Note: The 30-pin connector FireWire interface is deprecated. Only the 3G iPod, iPod mini, 4G iPod and iPod photo support FireWire data connections. All other iPod models are limited to FireWire charging only.

For legacy purposes, the 30-pin connector FireWire interface is designed to the IEEE standard 1394a, supporting transfer rates up to 400 Mbps. Per the IEEE 1394a specification, digital twisted pairs of wires need to be reversed as shown in Figure 2-1 (page 29).

Figure 2-1 30-pin to FireWire cable



The FireWire power pins on the 30-pin connector support 8-volt to 15-volt DC power input (8-volt to 30-volt power is allowed for legacy accessories). FireWire pins require an 8 watt power supply. **Developers of new iPod accessories that charge the iPod are required to design accessories that use the USB power pins for charging instead of the FireWire pins.**

USB 2.0

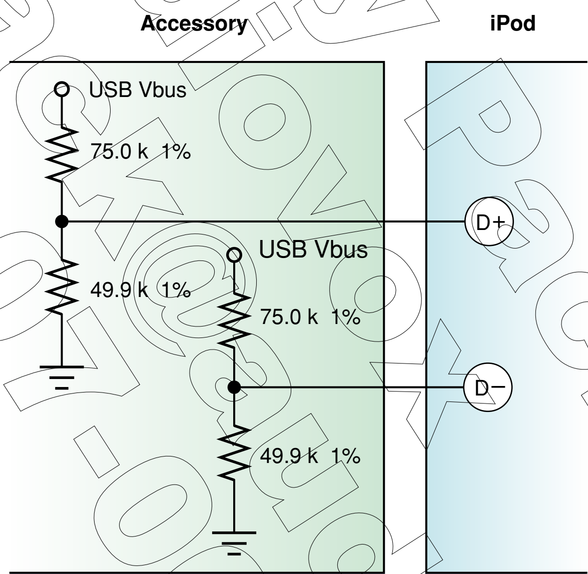
The 30-pin connector includes a USB interface designed to the USB 2.0 High Speed Specification. For full specifications of the Universal Serial Bus, you should refer to http://www.usb.org/developers/docs.

The iPod provides two configurations, or modes, of USB device operation: mass storage and iPod USB Interface (iUI). The iUI allows the iPod to be controlled using iAP, using a USB Human Interface Device (HID) interface as a transport mechanism. See "USB Port Link" (page 48) for more information.

If a device is to power the iPod using the USB power pins and it does not communicate with the iPod using the USB data pins (or pass through those pins, as in a dock), D+ and D- should be connected in the device as shown in Figure 2-2 (page 30) and Figure 2-3 (page 30). If the device can source 500

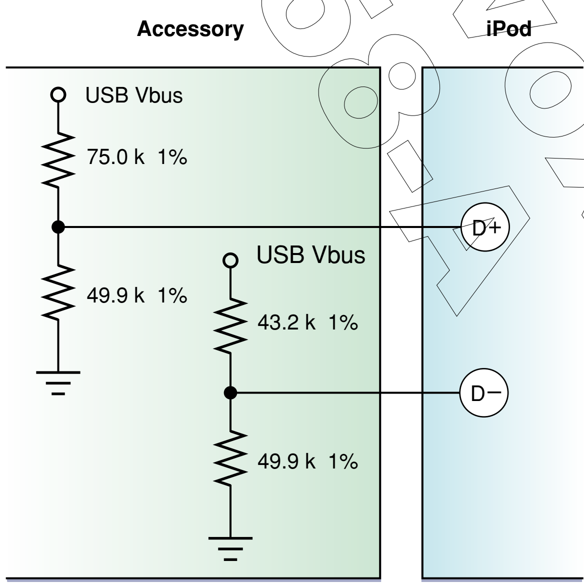
mA (maximum), connect the pins as shown in Figure 2-2 (page 30). The accessory device nominally +5 volt source should be USB Vbus power. For information about USB power, see http://www.usb.org/.

Figure 2-2 D+ and D- connections for a 500 mA USB power brick



If the device can source 1 A (maximum), connect the D+ and D- pins as shown in Figure 2-3 (page 30).

Figure 2-3 D+ and D- connections for a 1 A USB power brick



Note that devices which use the iPod in mass storage mode or in iUI mode should not make these connections.

USB power bricks are required to use appropriate resistors as shown above; the iPod needs them to determine how much current will be supplied. To prevent electrical problems, some models of iPods will not charge if an attached power brick lacks the necessary resistors.

AC adapters for the iPod touch must be designed to minimize electrical interference with the touch screen; see "AC Adapter Guidelines" (page 267) for more information.

Accessory 3.3 V Power

The iPod Accessory Power pin supplies 3.0 V to 3.3 V ±5% (2.85 V to 3.465 V) over the 30-pin connector. A maximum current of 100 mA is shared between the 30-pin connector and the 9-pin Audio/Remote connector.

By default, the iPod supplies a 5 mA current. Proper software accessory detect is required to turn on high power (up to 100 mA) during active device usage. When devices are inactive, they must consume less than 5 mA current.

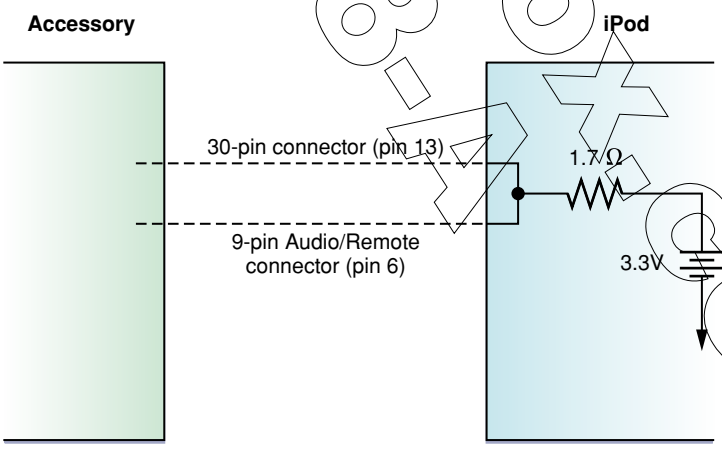
Accessory power is switched off for at least 2 seconds during the iPod bootstrap process. This is done to ensure that accessories are in a known state and can be properly detected. All accessories are responsible for re-identifying themselves 80 ms after the iPod completes the bootstrap process and transitions accessory power from the off to the on state.

Note: Self-powered accessories, such as automotive head units or docks, must wait until 80 ms after the attached iPod has turned on accessory power before trying to communicate with it.

Accessory power is grounded through the F/W GND pins.

See the schematic diagram in Figure 2-4 (page 31).

Figure 2-4 Accessory power



When Accessory Power is off, the Serial Receive block in the iPod's UART may also be off. To avoid incompatibility, the accessory should turn off the serial marking state it sends to the RX line (pin 18) of the 30-pin connector when Accessory Power goes low. Any attempt by the accessory to drive the UART serial line high when the iPod's Serial Receive block is off turns on protection diodes in the iPod. This condition wastes power in the accessory and can cause adverse behavior in the iPod.

Accessory Detect and Identify

Note: Some of the material in this section does not apply to accessories that need to support the 3G iPod. See "Interfacing With the 3G iPod" (page 255).

There are two types of iPod accessories; *serial accessories* that communicate with the iPod using iAP and *resistor-based accessories* that need access to specific iPod behaviors.

Resistor-based accessories use an Accessory Identify resistor (R_{ID}) to get access to a specific iPod behavior. These devices tend to be simple accessories, such as battery packs and car chargers, and have one specific purpose. When attached, these accessories unlock the iPod features described in Table 2-3 (page 33), based on the R_{ID} used.

Accessories that communicate with the iPod using iAP are serial accessories. Serial accessories that communicate via UART must use an R_{ID} with one of the values shown in Table 2-3 (page 33); if the accessory does not need to unlock any specific resistor-based behavior, the value should be 549 kΩ. Serial accessories that communicate via USB do not require an R_{ID}. However, it is useful to include an R_{ID} with a value of 191 kΩ anyway, to force the iPod USB Interface (iUI) to be the iPod's default USB Configuration. See "Choosing an iPod USB Configuration" (page 48) for more information about the iUI.

Accessory developers should choose the 549 kΩ or 191 kΩ resistor value based on their product's needs and use iAP to register the device's supported features.

A simple resistor to ground allows the iPod to determine what has been plugged into the 30-pin connector. There is an internal pullup on the Accessory Identify pin. Two pins, Accessory Identify and Accessory Detect, are required for accessory identification and detection. See the schematic diagram in Figure 2-5 (page 32)

Figure 2-5 Accessory identify and detect

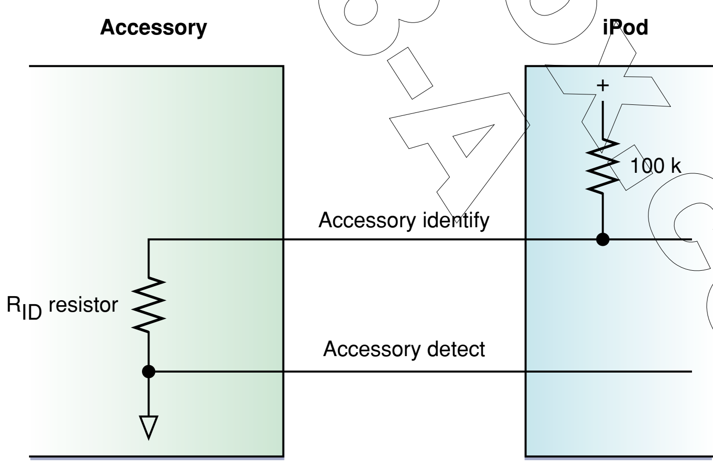


Table 2-3 (page 33) shows the specified resistor values for known accessories.

Note: Resistor tolerance for R_{ID} must be 1% or less.

Table 2-3 R_{ID} values, corresponding accessory functions, and iPod behavior

R_{ID}	Accessory Function	Exceptional iPod behavior
3.01 kΩ	3G and 4G simple docks	The 3G, 4G, and mini iPods will beep when connected. **This function is deprecated.**
191 kΩ	iAP over USB	The iPod will make the iPod USB interface (iUI) its default, and the mass storage configuration will become configuration 2.
255 kΩ	Battery pack (FireWire only)	The iPod will not charge its own battery from the accessory power supply. The iPod user interface will display the battery level as full, rather than charging.
549 kΩ	iAP over UART	No exceptional behavior.
1 MΩ	Car charger	The iPod will pause its playback when FireWire or USB power is removed.

Accessories that connect to the dock connector with a cable should choose a cable length that ensures that the voltage between the Accessory Identify pin (pin 10) and ground is in the desired voltage range. Careful attention should be paid to the ground conductors. Cable lengths will depend on cable resistance and wire gauge.

Note: Devices that identify themselves as 3G and 4G simple docks, battery packs, or car chargers are also able to communicate with iPods via iAP.

iPod Detect for Self-Powered Accessories

If a powered accessory needs to detect that an iPod has been plugged in, even when the iPod has no battery power, is hibernating, or is deep sleeping, it must tie its pin 30 to its supply voltage through a resistor and examine the state of the signal on the line. The signal state will be high when no iPod is attached The signal state will be pulled low when an iPod is attached because pin 30 on the iPod is tied to ground.

Note: Use pins 15 and 16 for digital ground.

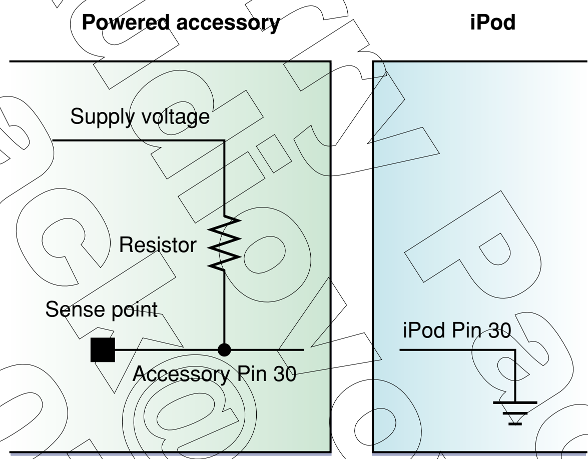


Figure 2-6 iPod detect

Note: The elapsed time it takes an iPod to bootstrap and be fully operational varies by model type, media types present, storage capacity, and the starting boot state (e.g., deep sleep, hibernate, or reset). An iPod will not receive or respond to any iAP commands while accessory power is switched off. To use the iAP, accessory devices must wait until 80 ms after the iPod switches on steady accessory power before sending any commands.

UART iPod Accessory Protocol Communication

Accessories using the iPod Accessory Protocol (iAP) over the UART serial port link use two pins, RX and TX, to communicate to and from the iPod. The signaling levels for these pins are shown in Table 2-4 (page 34).

Table 2-4 iPod RX and TX signaling levels

		MIN	MAX	Units
Input Voltage High	V_{IH}	2.00	3.3	V
Input Voltage Low	V_{IL}	0.0	0.5	V
Output Voltage High	V_{OH}	2.5	3.47	V
Output Voltage Low	V_{OL}	0.0	0.5	V

A device with a 549 kΩ R_{ID} is a serial accessory; these devices use the iPod Accessory Protocol. Attaching a serial accessory to the 30-pin connector of a 3G iPod makes any accessories attached to the 9-pin Audio/Remote connector inactive, because the 3G iPod shares the serial port between these two connectors. The iPod mini, 4G iPod, and iPod photo models have two serial ports, so plugging in a 30-pin connector port serial accessory does not deactivate the 9-pin Audio/Remote connector.

USB iPod Accessory Protocol Communication

The iPod has two mutually exclusive USB Configurations that allow it to function as a mass storage device or as a device capable of performing iAP over USB. When R_{ID} is not 191 kΩ, USB Configuration 1 is the mass storage device while USB Configuration 2 is the iUI device. When R_{ID} is 191 kΩ, the iPod swaps the configuration numbers; therefore USB hosts that always select configuration 1 can enable the iAP over USB on the iPod. See "USB Port Link" (page 48) for more information on the iPod USB Interface (iUI) and iAP over USB.

iPod Video Signal Levels

Video signal levels from the iPod 30-pin connector are shown in Table 2-5 (page 35). These levels assume that the video outputs are properly load-terminated into 75 Ω ±1%.

By default, the iPod sends only a composite video signal. Some iPod models also send S-Video signals by default. Use the General lingo command `SetiPodPreferences` to select the appropriate signal for your accessory.

To receive video signals from most iPods, an accessory must be authenticated. The iPod photo and 5G iPod are exempt from this requirement.

Table 2-5 Video signal levels in volts peak-to-peak

Description	Minimum	Typical	Maximum
Composite Video Amplitude	0.30		1.0
Composite Video Burst Amplitude (NTSC)	0.21	0.28	0.33
Composite Video Sync Amplitude (NTSC)	0.28	0.28	0.33
S-Video Luminance Amplitude	0.30		1.0
S-Video Chrominance Amplitude	0.0		0.70
S-Video Chrominance Burst Amplitude (NTSC)	0.21	0.28	0.33
Composite Video Burst Amplitude (PAL)	0.27	0.30	0.33
Composite Video Sync Amplitude (PAL)	0.27	0.30	0.33
S-video Burst Amplitude (PAL)	0.27	0.30	0.33
Component Video Luminance ("Y")	0.3		1.0
Component Video Chrominance ("Pr")	0.0		0.7
Component Video Chrominance ("Pb")	0.0		0.7

Line Level Input (Left & Right)

The 30-pin connector supports both stereo and mono audio input. Mono audio input is through the left channel. The input level is 0.600 Vrms maximum (supported on the 2G nano, 5G iPod, iPod classic, and 3G nano). Figure 2-7 (page 36) shows a reference schematic for line-in impedance on all models except the iPod classic. The corresponding reference schematic for the iPod classic is shown in Figure 2-8 (page 36) .

Figure 2-7 iPod equivalent input circuits (except the iPod classic)

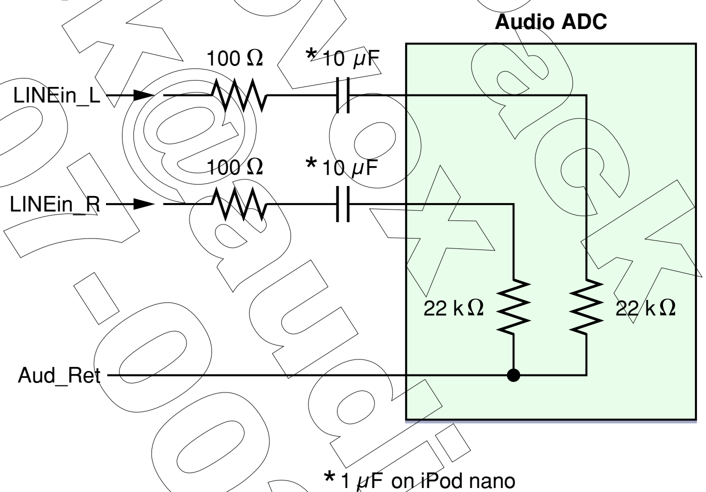
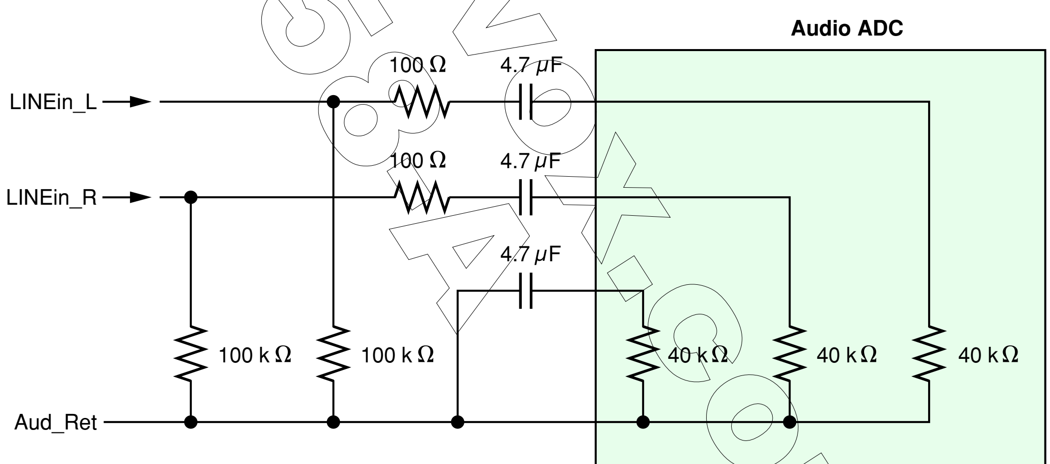


Figure 2-8 iPod equivalent input circuits for the iPod classic



Note: The Audio Return pin is NOT a ground and should not be tied directly to the digital ground (DGND); treat it as a signal. Left and right audio are referenced to Audio Return.

Line Level Output (Left & Right)

Stereo audio output from the iPod is at a fixed level; it is not adjustable. The output level is 0.900 Vrms maximum for the 3G iPod, 4G iPod, iPod mini, iPod photo, 5G iPod, iPod classic, and iPod touch. The output level is 0.700 Vrms for the iPod nano, 2G nano, and 3G nano.

Figure 2-9 (page 37) shows a reference schematic for line-out and headphone out impedance on all models except the iPod classic. The corresponding reference schematic for the iPod classic is shown in Figure 2-10 (page 38) . Note that values are approximate, and they may be changed in the future without notice.

Figure 2-9 iPod equivalent output circuits (except the iPod classic)

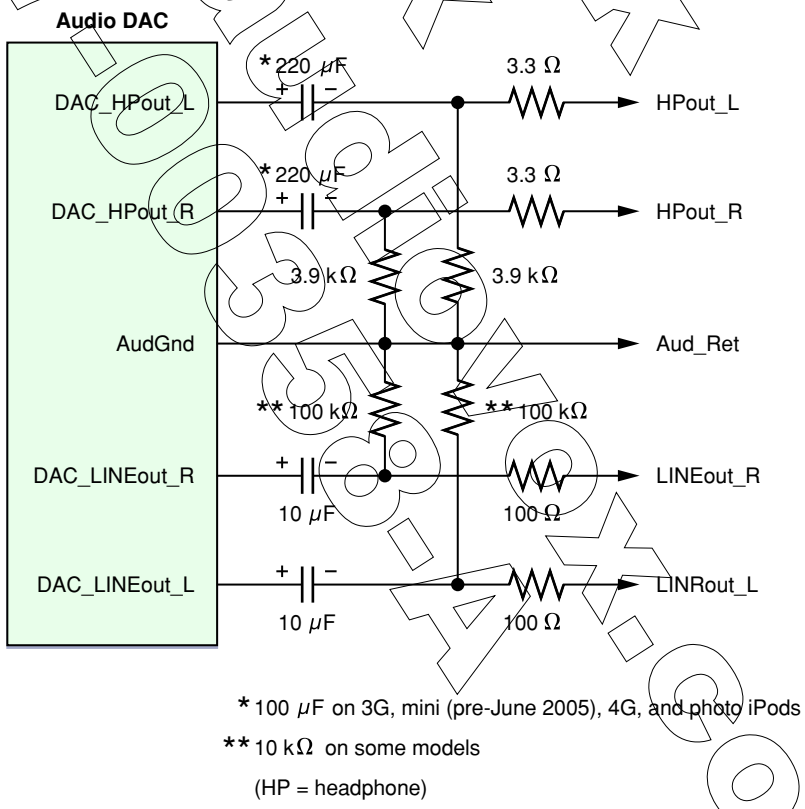
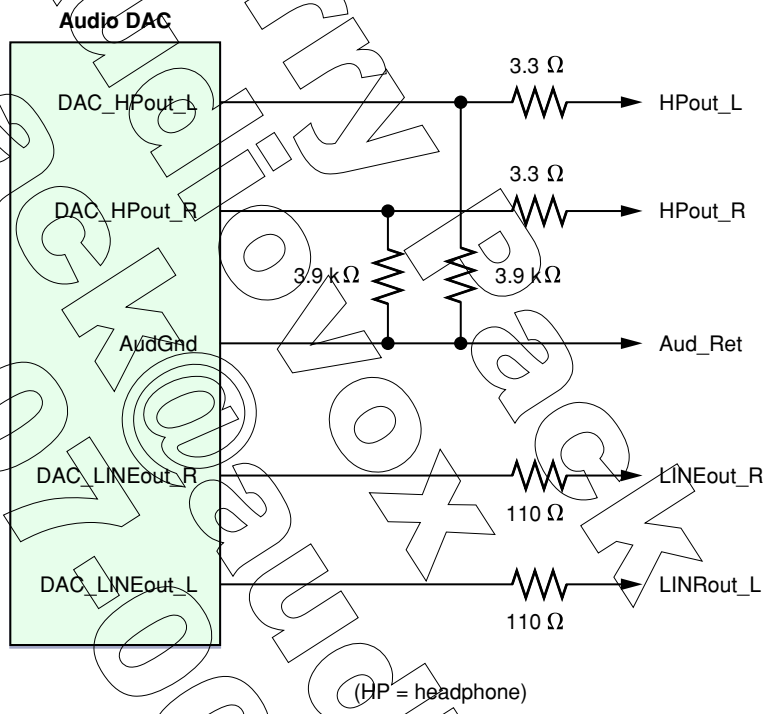


Figure 2-10 iPod equivalent output circuits for the iPod classic



Note: Under some circumstances the 2G iPod nano, the iPod classic, and 3G nano disable their line-out circuits to save power. Line-out is disabled only if a headphone is detected, no accessory resistor is detected, **and** power is not attached to the iPod.

Overall Grounding Requirements

Chassis ground is tied to the specified pins; see "Connector Pin Designations" (page 23).

IMPORTANT: Digital ground should not be tied to Audio Return.

Minimizing Crosstalk and Noise

Accessories that receive audio or video signals from an iPod must be designed to minimize crosstalk and extraneous noise. Both the accessory's circuitry and physical trace layout must take these issues into account.

iPods with video output capability use single-ended analog audio and video signals. To conserve pins on connectors and conductors in cables, the audio and video signals share a common return path. To mitigate video-to-audio crosstalk, active differential amplifiers are used in the audio path. The differential amplifier stage is configured to sense a low side return voltage representing the video-to-audio crosstalk. This signal is then applied to the load in such a way that minimum noise current flows through the load. For sample schematics and advice on circuit layout, see "Sample Accessory Circuits" (page 257).

In iPods without video capability, pins 21, 22, and 23 are not connected internally. See "30-pin Connector" (page 23) for descriptions of these pins.

9-pin Audio/Remote Connector

This section describes the features and physical characteristics of the 9-pin Audio/Remote connector.

Audio Out

The 9-pin Audio/Remote connector provides stereo output of approximately 25 mW per channel. Output volume is controlled by the iPod.

Mono Mic In on the 9-Pin Audio/Remote Connector

Note: Depending on the mic element, a preamp may be required.

Mono microphone input is through the left channel. On the 3G iPod, filtered electric power is supplied by an internal device.

Accessory 3.3 V Power

The iPod supplies 3.0-volt through 3.3-volt accessory power ±5% (2.85 V through 3.465 V) over the 9-pin Audio/Remote connector; this is the same as the 30-pin connector. A maximum current of 100 mA is shared between the 30-pin connector and the 9-pin Audio/Remote connector. For more information, see "Accessory 3.3 V Power" (page 31).

UART iPod Accessory Protocol Communication

Accessories using the iPod Accessory Protocol (iAP) over the UART serial port link use two pins, RX and TX, to communicate to and from the iPod. The signaling levels for the RX and TX pins are shown in Table 2-4 (page 34).

Note: The 9-pin Audio/Remote connector does not support the R_{ID} feature.

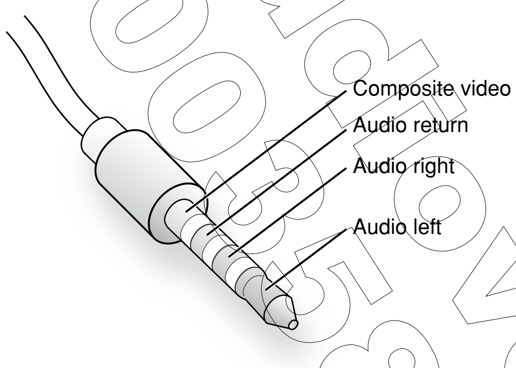
iPod touch Carrying Case Design

The iPod touch user interface senses the presence of one or more fingers on its upper surface. Any material between the surface and a user's hand, even a very thin sheet of plastic, can affect the performance of the touch interface and must be tested for compatibility. Touch screen covers must be thinner than 0.3 mm and should be designed so that there are no air gaps between the cover and the screen surface. Covers must not be electrically conductive.

Headphone Jack on the iPod photo and 5G iPod

For legacy purposes only, the iPod photo and 5G iPod can provide composite video signals through their headphone jack, as shown in Figure 2-11 (page 40). These signals have the same electrical characteristics as the composite video signals available from the 30-pin connector; see Table 2-5 (page 35). **This use of the composite video signal on the headphone jack is deprecated.**

Figure 2-11 Pinout for the headphone jack on the iPod photo and 5G iPod



iPod Accessory Protocol

iPod Accessory Protocol (iAP) allows the iPod to communicate with a functional range of external devices. This specification breaks the protocol into three logical components: the protocol transport link, the core iAP functionality, and the individual accessory lingoes. Transport linking is described in "Protocol Transport Links" (page 47). Devices can use USB and UART serial as channels to transfer iAP packets. "The Protocol Core and the General Lingo" (page 55) describes the commands needed for iPods to communicate with all external devices. It encompasses the basic packet definition and the General Lingo, which allows for accessory identification, authentication, and retrieval of iPod information. The accessory lingoes, described in "Accessory Lingoes" (page 107), consist of the individual dialect commands. Each accessory lingo corresponds to a functional class of external devices.

Every external device must support a protocol transport link, the protocol core, and one or more lingoes, as required for its function. For example, the iPod standard in-line remote control is a UART serial device that uses the General lingo and the Simple Remote lingo.

Protocol Features and Availability

Table 3-1 (page 43) lists the protocol versions for each lingo and the firmware releases in which they were available. Table 3-2 (page 45) lists hardware and software features outside these protocols.

In Table 3-1 (page 43), "NL" indicates that the given lingo was not implemented in the specified model loaded with the specified firmware. "NV" indicates that although the lingo was implemented, its protocol version could not be read from the iPod. Footnotes are listed below the table.

Table 3-1 iPod models, firmware, and lingo versions

iPod model	Firmware		Lingoes										
	Version	Date	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x0A	0x0C
3G [1]	2.0.0	04/2003	NV	NL	NV	NL	NL	NL	NL	NL	NL	NL	NL
	2.1.0	10/2003	NV	NV	NV	NL	1.00	NL	NL	NL	NL	NL	NL
	2.2.0	02/2004	NV	NV	NV	NL	1.02	NL	NL	NL	NL	NL	NL
	2.3.0	02/2005	NV	NV	NV	NL	1.02	NL	NL	NL	NL	NL	NL
mini	1.0.0	02/2004	NV	NL	NV	NL	1.01	NL	NL	NL	NL	NL	NL
	1.1.0	03/2004	NV	NL	NV	NL	1.03	NL	NL	NL	NL	NL	NL
	1.2.0	11/2004	1.00	NL	1.00	NL	1.05	1.00	NL	NL	NL	NL	NL
	1.3.0	02/2005	1.00	NL	1.00	NL	1.05	1.00	NL	NL	NL	NL	NL
	1.4.0	06/2005	1.02	NL	1.00	1.01	1.05	1.00	NL	NL	NL	NL	NL
	1.4.1	01/2006	1.02	NL	1.00	1.01	1.05	1.00	NL	NL	NL	NL	NL
4G	3.0.0	07/2004	NV	NV	NV	NL	1.04	NV	NL	NL	NL	NL	NL
	3.0.1	08/2004	NV	NV	NV	NL	1.04	NV	NL	NL	NL	NL	NL
	3.0.2	11/2004	1.00	1.00	1.00	NL	1.05	1.00	NL	NL	NL	NL	NL
	3.1.0	06/2005	1.02	1.00	1.00	1.01	1.05	1.00	NL	NL	NL	NL	NL
	3.1.1	01/2006	1.02	1.00	1.00	1.01	1.05	1.00	NL	NL	NL	NL	NL

iPod model	Firmware		Lingoes										
	Version	Date	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x0A	0x0C
Photo	1.0.0	10/2004	1.00	1.00	1.00	NL	1.05	1.00	NL	NL	NL	NL	NL
	1.1.0	03/2005	1.01	1.00	1.00	1.00	1.06	1.00	NL	NL	NL	NL	NL
	1.2.0	06/2005	1.02	1.00	1.00	1.01	1.06	1.00	NL	NL	NL	NL	NL
	1.2.1	01/2006	1.02	1.00	1.00	1.01	1.06	1.00	NL	NL	NL	NL	NL
nano	1.0.0	09/2005	1.02	NL	1.00	1.02	1.07	1.00	NL	1.00	1.00	NL	NL
	1.1.0	01/2006	1.03	NL	1.00	1.03	1.09	1.00	NL	1.00	1.00	NL	NL
	1.1.1	03/2006	1.03	NL	1.00	1.03	1.09	1.00	NL	1.00	1.00	NL	NL
	1.2.0	06/2006	1.04	NL	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.00 [2]	NL
	1.3	10/2006	1.05	NL	1.02	1.05	1.11	1.01	NL	1.00	1.00	1.01	NL
	1.3.1	03/2007	1.05	NL	1.02	1.05	1.11	1.01	NL	1.00	1.00	1.01	NL
5G	1.0.0	10/2005	1.03	1.01	1.01	1.03	1.08	1.00	1.00	1.00	1.00	NL	NL
	1.1.0	01/2006	1.03	1.01	1.01	1.03	1.09	1.00	1.00	1.00	1.00	NL	NL
	1.1.1	03/2006	1.03	1.01	1.01	1.03	1.09	1.00	1.00	1.00	1.00	NL	NL
	1.1.2	06/2006	1.03	1.01	1.01	1.03	1.09	1.00	1.00	1.00	1.00	NL	NL
	1.2.0	09/2006	1.05	1.01	1.02	1.05	1.11	1.01	1.00	1.00	1.00	1.01	NL
	1.2.1	11/2006	1.05	1.01	1.02	1.05	1.11	1.01	1.00	1.00	1.00	1.01	NL
2G nano	1.0.0	09/2006	1.04	1.01	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.00 [2]	NL
	1.0.1	09/2006	1.04	1.01	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.00 [2]	NL
	1.0.2	09/2006	1.04	1.01	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.00 [2]	NL
	1.1	10/2006	1.04	1.01	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.00 [2]	NL
	1.1.1	10/2006	1.04	1.01	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.00 [2]	NL
	1.1.2	02/2007	1.04	1.01	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.02	NL
	1.1.3	05/2007	1.06	1.01	1.02	1.04	1.10	1.01	NL	1.00	1.00	1.02	NL
classic	1.0	09/2007	1.07	1.01	1.02	1.05	1.12	1.01	NL	1.00	1.00	1.03	1.01
	1.0.1	09/2007	1.07	1.01	1.02	1.05	1.12	1.01	NL	1.00	1.00	1.03	1.01

iPod model	Firmware		Lingoes										
	Version	Date	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x0A	0x0C
3G nano	1.0	09/2007	1.07	1.01	1.02	1.05	1.12	1.01	NL	1.00	1.00	1.03	1.01
	1.0.1	09/2007	1.07	1.01	1.02	1.05	1.12	1.01	NL	1.00	1.00	1.03	1.01
touch	1.1	09/2007	1.07	NL	1.02	1.05	1.12	1.01	NL	NL	1.00	1.02	NL
	1.1.1	09/2007	1.07	NL	1.02	1.05	1.12	1.01	NL	NL	1.00	1.02	NL

[1] Supporting the 3G iPod requires special design considerations. See Appendix C, "Interfacing With the 3G iPod" (page 255).

[2] Because version 1.00 of Lingo 0x0A contained a bug that was corrected in version 1.01, accessories should attempt Digital Audio only with iPods whose Lingo 0x0A version is higher than or equal to 1.01. See Table 6-164 (page 219).

The General lingo (Lingo 0x00) allows accessory devices to extract the lingo version number from the iPod. This protocol version information can be used to determine which features the connected iPod supports. Please refer to sections "Lingo 0x00: General Lingo" (page 64) and "Command 0x0F: RequestLingoProtocolVersion" (page 83) for more information on the General lingo and the `RequestLingoProtocolVersion` command.

Table 3-2 (page 45) lists iPod hardware and software features that are not part of specific lingoes. The numbers in the table show the firmware versions in which each of these features was introduced.

Table 3-2 Features supported by specific iPod firmware versions

Features	Software versions									
	3G	mini	4G	photo	nano	5G	2G nano	classic	3G nano	touch
Serial baud rates: 38400/57600	—	1.2.0	3.0.2	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.1
Serial autobaud on framing errors	—	1.4.0	3.1.0	1.1.0	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.1
Display notification on unsupported iAP device attach	—	—	—	—	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.1
iPod detects detach for all valid R_{ID} values	—	1.4.0	3.1.0	1.2.0	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.1
Authentication Version 1.00	—	—	—	—	1.0.0	1.0.0	1.0.0	1.0.0	1.0.0	1.1
Authentication Version 2.00	—	—	—	—	1.2.0	1.2.0	1.0.0	1.0.0	1.0.0	1.1

Features	Software versions									
	3G	mini	4G	photo	nano	5G	2G nano	classic	3G nano	touch
USB audio	—	—	—	—	1.2.0 [1]	1.2.0	1.1.2	1.0.0	1.0.0	1.1
Video browsing	—	—	—	—	—	1.2.0	—	1.0.0	1.0.0	—

[1] Because version 1.00 of Lingo 0x0A contained a bug that was corrected in version 1.01, accessories should attempt Digital Audio only with iPods whose Lingo 0x0A version is higher than or equal to 1.01. See Table 6-164 (page 219).

iPod Games

At this time, iPod games cannot be controlled by accessories. Only Simple Remote commands can affect iPod games. The iPod may ignore other commands while a game is being played.

Protocol Transport Links

Accessories may communicate with the iPod using iAP over the serial port link or the USB port link. This chapter describes those transport links.

Note: The 3G iPod, iPod mini, 4G iPod, and iPod photo do not support iAP over USB.

Developers should note that both links cannot be used simultaneously by a single device; only one transport can be used at a time. If an accessory first registers for accessory lingoes using the serial port, it must re-identify on that port as a General lingo device before identifying on the USB port. This action clears the supported lingoes on the serial port before the accessory registers for the same lingoes using the USB port. With the exception of the General lingo, lingoes can be active on only one port at a time. Identifying for accessory lingoes on two ports at one time causes the second `Identify` command to return an error ACK.

UART Serial Port Link

The iPod Accessory Protocol builds upon the RS-232 serial specification. However, the signaling levels are non-standard. The RS-232 specification states that a mark is –7 V and a space is +7 V. In this protocol, a mark is 2.85 V through 3.465 V and a space is 0 V through 0.9 V.

Accessories must communicate with all iPod models at either 19200 bps or 57600 bps and maintain their baud rates within ±2% of the chosen rate over the entire temperature range of the accessory. The temperature range of the accessory must be greater than or equal to the temperature range of the iPod (0–35° C). Once an accessory has started communicating with an iPod, it cannot change its baud rate.

Note: Current iPod models are capable of automatic baud rate detection (autobauding) between 9600 bps and 24000 bps, as shown in Table 3-2 (page 45). However, this feature may not be available in future iPods; newly designed accessories should communicate only at either 19200 bps or 57600 bps.

As is detailed later in this specification, a sync byte (`0xFF`) must precede every packet when using the UART serial port link, to ensure that the automatic baud rate detection is accurate. All serial communications use 8 data bits, no parity bits, and one stop bit (8-N-1). Serial hardware flow control (RTS/CTS and DTR/DSR) and XON/XOFF protocols are not used and will be ignored.

USB Port Link

The iPod is a USB 2.0–compliant device that supports two mutually exclusive modes of operation:

- Mass storage device. This is the default configuration when attached to a typical USB host such as a PC or Macintosh. This mode is used for syncing music and content, transferring files, and so forth.
- iAP-enabled device. This is the configuration needed to support iAP using the iPod USB Interface (iUI). This mode must be selected by the USB host before it can be used.

Note: Apple recommends against designing accessories that use an iPod as a mass storage device. Accessories should always use the iUI to interact with an iPod.

These two mutually exclusive modes of operation are each represented by a USB Configuration. When an iPod is attached to USB, the USB host (the accessory) must select one of the configurations and set it as the active configuration during the bus initialization. Alternatively, an accessory can use a resistor to make the iUI the default configuration, as shown in Table 2-3 (page 33).

The iPod supports Full-Speed as well as High-Speed bus operation. However, High-Speed hosts are strongly encouraged for better data throughput.

Choosing an iPod USB Configuration

The initialization and configuration of an attached USB device is documented in the USB 2.0 specification. This document does not cover this topic in detail, but instead provides information specific to the iPod. To distinguish an iPod, a USB host can check the device descriptor of attached USB devices for the following fields:

- Vendor ID: 0x05AC
- Product ID: 0x12*nn*

Note that product IDs vary, depending on the type of iPod. There will be an expanding list of iPod models that support iUI. Although an iPod may make string descriptors available that identify its manufacturer, product name, serial number, configuration, and so on, the accessory should not use these strings to determine whether the connected USB device is an iPod. The strings may change in future iPods.

IMPORTANT: An iPod accessory should use the Vendor ID and the most significant byte of the Product ID to recognize an iPod on the USB. It can complete the identification by checking for the presence of an iUI configuration on the detected iPod.

Figure 4-1 (page 49) shows the USB configuration and interface descriptors in iPods that do not support USB audio. Figure 4-2 (page 49) shows the USB Configuration and interface descriptors in iPods that do support USB audio.

Figure 4-1 Configuration and interface descriptors for iPods without USB audio

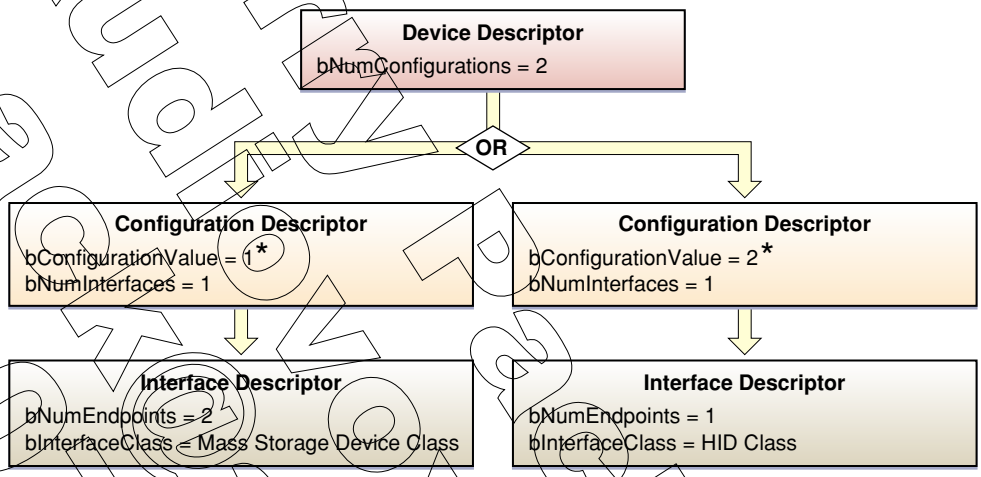
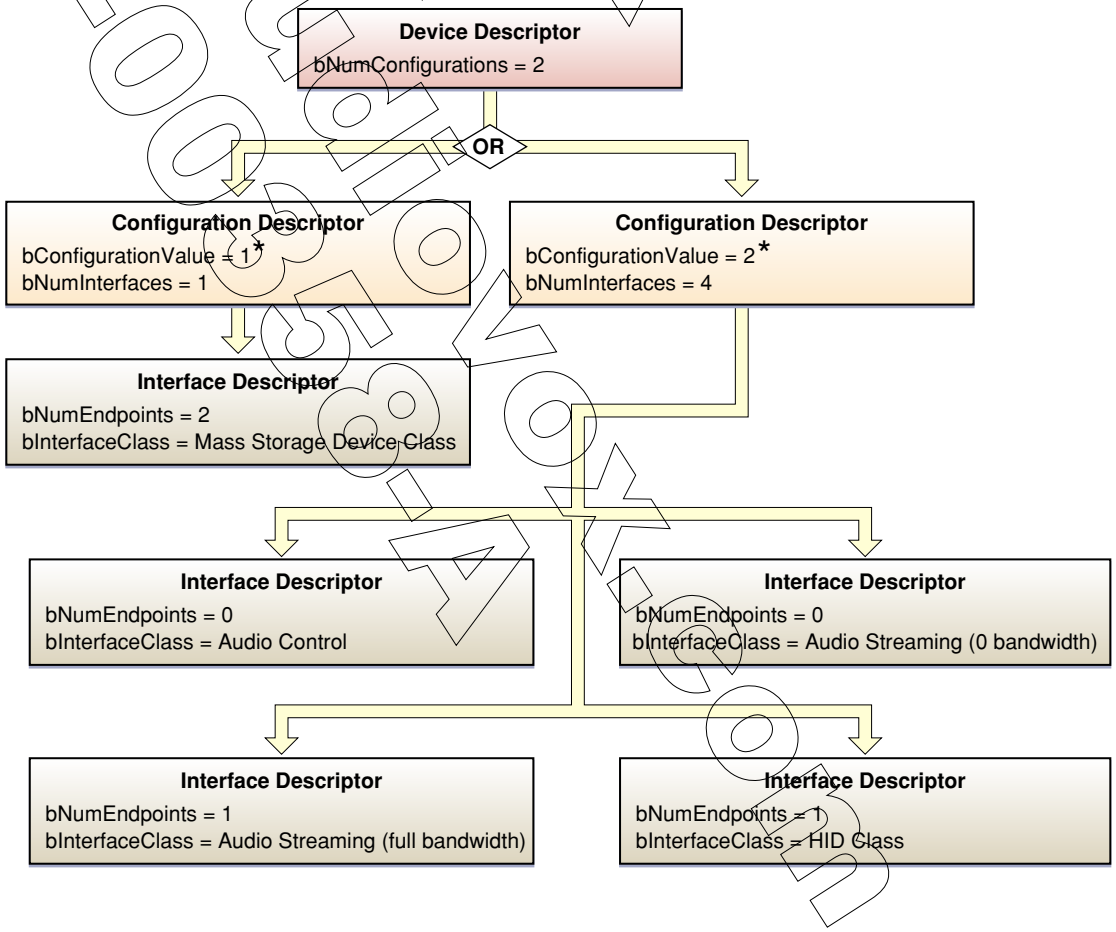


Figure 4-2 Configuration and interface descriptors for iPods with USB audio



* R_{ID} = 191 k Ω can be used to swap configuration values

Some operating systems have USB driver subsystems that already provide support for disk devices and HID devices. These systems automatically recognize and configure an iPod with its default configuration. Unfortunately, some of them are designed to work only with the default configuration and prevent any custom application from changing their behavior to use the second configuration, as is needed to utilize the iUI.

In order to accommodate such hosts, the iPod provides a mechanism to make the iUI configuration the default, using an R_{ID}.

Accessory Identify Resistor and iUI

The presence of a 191 kΩ R_{ID} (see Figure 2-11 (page 40)) on USB attachment triggers the iPod to present the iUI configuration as the first, or default, configuration. The second configuration becomes iPod as a Mass Storage class disk device.

Note that iPod accessory developers that wish to support USB-capable iPods as well as older UART serial-only iPods with the same connector must decide whether the use of a 191 kΩ resistor is beneficial to them. Older UART serial-only iPods require a 549 kΩ R_{ID} to function properly and preclude the use of a 191 kΩ resistor to trigger USB functionality on newer iPods.

Older iPods detect that a serial accessory has been detached only when its associated 549 kΩ R_{ID} has been disconnected from the iPod. These older iPods check only whether the 549 kΩ resistor state has changed. If an accessory uses the 191 kΩ resistor on these older iPods, the iPods cannot tell when the accessory has been detached because the accessory does not use the 549 kΩ resistor. The only recovery for this situation is to reset the iPod or for the user to attach and detach an accessory that does have the 549 kΩ resistor. See Table 3-2 (page 45) for a list of the iPods that support detecting detach for all R_{ID} values.

It is possible to create an accessory that supports both iAP over USB and the older UART serial-only iPods using the same connector. Accessory developers have two options:

1. Create an accessory that dynamically switches between the 191 kΩ and 549 kΩ R_{ID} values, based on the type of iPod that is connected.

2. Use a 549 kΩ R_{ID} and have the USB host manually select the iUI USB Configuration.

iPod USB Interface (iUI) Configuration

The iUI configuration allows the iPod to communicate via iAP over USB and enables USB digital audio on supported models. The USB Human Interface Device (HID) interface is the transport link and uses two endpoints for communication: the control endpoint (endpoint number 0) is used for OUT data, while the HID interrupt endpoint is used for IN data.

The iPod HID interface utilizes several vendor-specific HID reports, some of which are used to transport data from the host (output reports) and some of which are used to transport data to the host (input reports). In order to send data to the iPod, a host chooses one or more appropriately sized HID reports in which to embed the iAP packet and sends this to the iPod HID interface with USB

Set_Report commands. The iPod reassembles the iAP packet and processes it. The process is repeated in reverse when the iPod sends responses or iAP packets to the host. In this case, the data is sent on an interrupt pipe associated with the HID interface.

The different HID report sizes, endpoint requirements, and particulars are all described in the USB descriptors that accompany the interface.

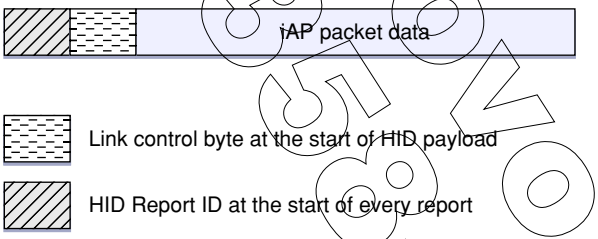
IMPORTANT: Accessories must always parse the HID descriptor each time an iPod is connected, because the HID Report ID and size descriptions may change.

HID as a Transport

As mentioned earlier, the HID interface breaks iAP packets up into a stream of vendor-specific HID reports and transports them across USB in either direction. To help manage this, it breaks this stream up into logical sets of reports, where a set of reports encompasses one or more complete iAP packets. For instance, a set could be a single HID report containing one iAP packet or a set of seven HID reports containing a total of three iAP packets.

A vendor-specific HID report, as defined by the USB specification, consists of a Report ID followed by a payload of data that is specific to the vendor and its usage. The payload of an iPod vendor-specific HID report is a link control byte (LCB), followed by iAP packet data. An example is shown in "iPod Accessory Protocol" (page 41).

Figure 4-3 iPod vendor-specific HID report



The HID Report ID indicates the type of report and implies the size of the report. Every report of a given type is the same size. The iPod specifies several different report types. The USB host should analyze the HID report descriptor of the iPod at runtime to determine which Report ID corresponds to the most appropriate report type for each transfer. Note that the HID report descriptor may change in future iPods.

Usage of a HID Report ID is defined by the USB specification and is not specific to the iPod, in contrast to the LCB and the rest of the payload.

The link control byte provides a mechanism for grouping sets of reports and is used by the HID interface to manage the data flow, as described in Table 4-1 (page 52).

Table 4-1 Link control byte usage

Bit	Name	Usage
Bit 0	Continuation	0 indicates that this HID report is the first in a set of one or more reports. This also implies that any previous sets are completed. Any incomplete iAP packets received prior to the arrival of this report are flushed and lost. 1 indicates that this report is not the start of a set, but is a continuing part of a set.
Bit 1	More to Follow	0 indicates that this report is the last in a set. Any following reports must be part of another set. 1 indicates that the current report set is not yet complete and there is at least one more report expected.
Bits 2–7	Reserved	Set to 0.

In general, iAP packets can be packed into HID reports in any manner, given the following limitations:

- All unused space within any HID report must be set to 0x00.
- If there is more than one iAP packet in the same HID report, there must be no unused space between them.
- If an iAP packet is split across multiple HID reports, all component reports must be in the same logical set of reports.

Figure 4-4 (page 53) shows the different report packing scenarios that are possible, including one packet per report, multiple packets per report, and multiple reports per packet.

Figure 4-4 Possible report packing scenarios

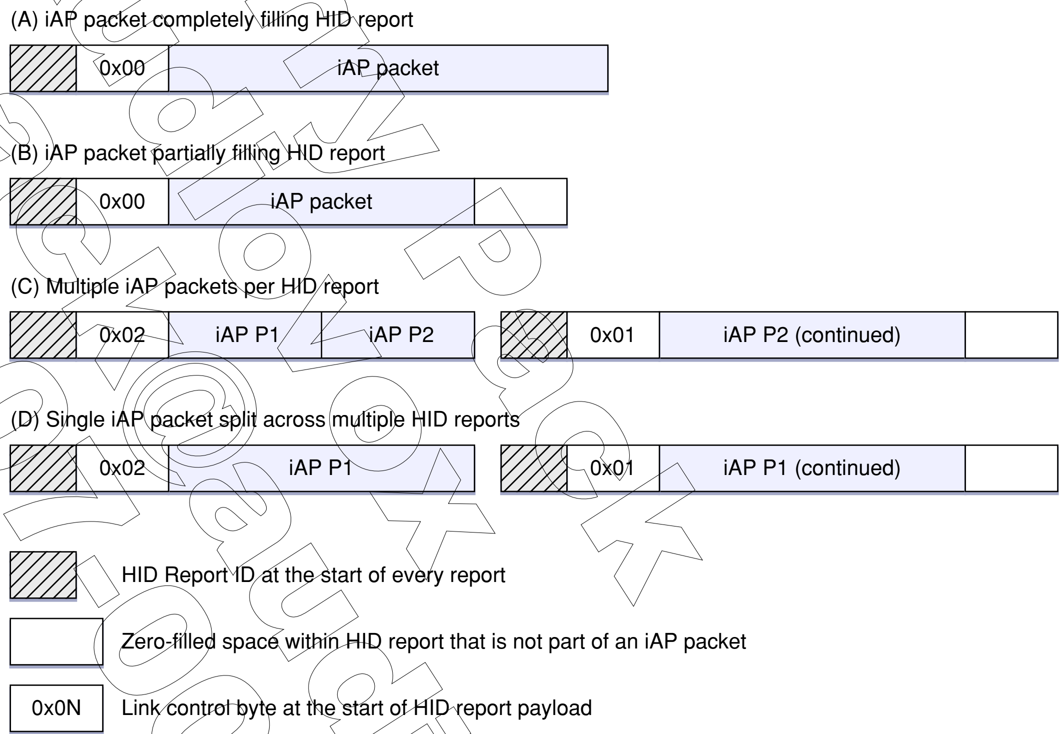
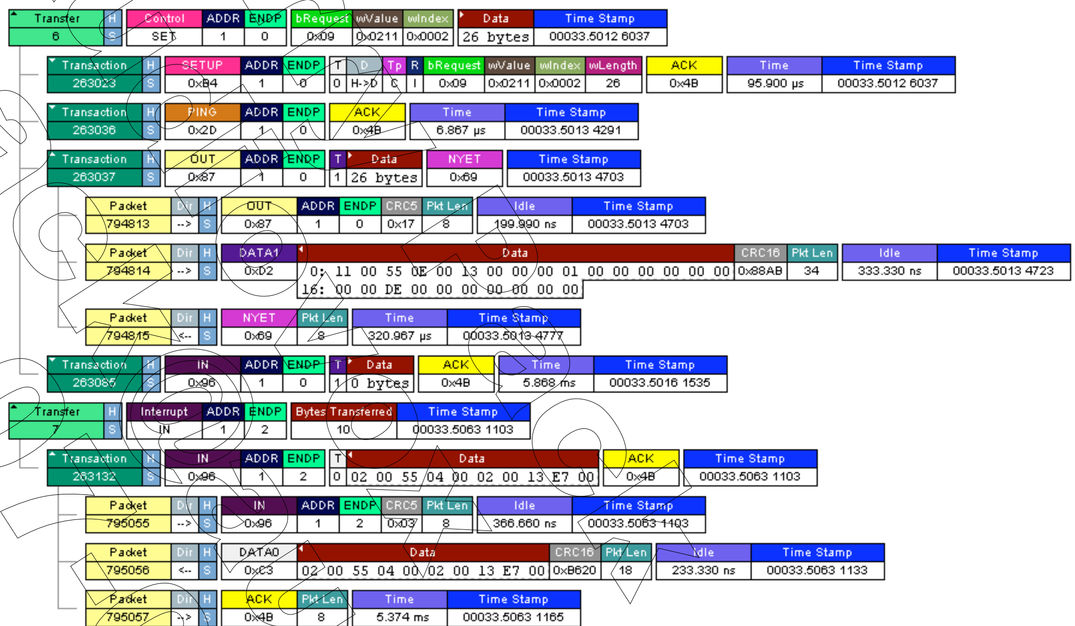


Figure 4-5 (page 54) illustrates a sample USB transport interchange that uses iAP to send an `IdentifyDeviceLingoes` command and receive back an `ACK` command. `IdentifyDeviceLingoes` is described in "Command 0x13: IdentifyDeviceLingoes" (page 84); the corresponding `ACK` command is described in "Command 0x2: ACK" (page 71). In this example, the commands were sent between a high-speed USB 2.0 host and a 1G iPod nano, using a cable with an embedded 191 kΩ resistor as R_{ID} (see "Accessory Detect and Identify" (page 32)). For clarity, the diagram omits SOF and NAK messages.

Figure 4-5 Transferring `IdentifyDeviceLingoes` and `ACK` commands over USB



iPod Sleep Behavior When Attached to USB Host

An iPod will not transition to Light Sleep while it remains attached to an active USB host. To ensure that a physically attached iPod goes to sleep, a host system should pause music playback before switching off its host controller.

Attaching USB power to a sleeping iPod wakes it up.

The Protocol Core and the General Lingo

This chapter covers the basics of the iPod Accessory Protocol (iAP), including accessory identification and authentication. It also describes the packet format used for iAP command and gives detailed descriptions of the commands included in the General lingo.

iAP is used in both directions of a link. Every device is encouraged to implement both sending and receiving capabilities. It should be possible to determine the direction (device to iPod or iPod to device) of a packet only from its contents. This means that no packet is valid for sending from both the iPod and the device.

All devices must be able to handle variable-length packets. For example, even though most identify packets currently have no defined data, a device must be able to understand an identify packet with extra data and should respond to the best of its ability. At a minimum, the device must not lose sync to the packet signaling.

Device Signaling and Initialization

When attached, the accessory should detect the iPod and initiate the identification process. This section describes the initialization process for both the UART serial port link and the USB port link.

Packet Signaling and Initialization Using the UART Serial Port Link

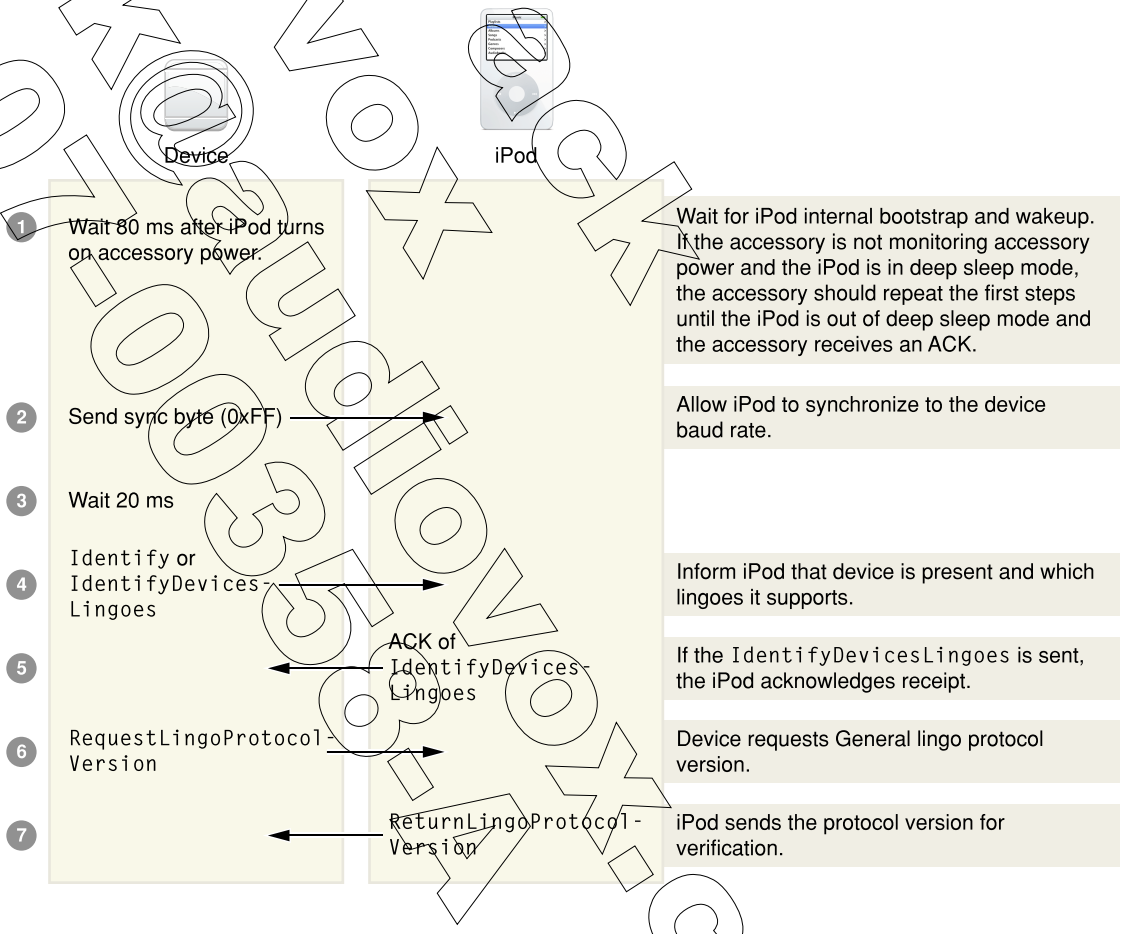
When using the UART serial port link, the identification process starts with the device waiting 80 ms and then transmitting a sync byte. After sending the sync byte, the accessory should wait another 20 ms before sending either "Command 0x01: Identify" (page 69) or "Command 0x13: IdentifyDeviceLingoes" (page 84) to notify the iPod of its supported lingoes. The iPod sends an ACK in response to the `IdentifyDeviceLingoes` command, but not in response to the `Identify` command.

The accessory should next check the version of the General lingo protocol supported by the iPod, by sending "Command 0x0F: RequestLingoProtocolVersion" (page 83) using the ID of the General lingo. The iPod responds with a "Command 0x10: ReturnLingoProtocolVersion" (page 84) packet containing the protocol version of the General lingo.

Note: If the iPod does not respond, the accessory should retry the `IdentifyDeviceLingoes` and `RequestLingoProtocolVersion` commands. If the retry also fails, the accessory should assume that these commands are not supported. If an iPod fails to ACK the `IdentifyDeviceLingoes` command, the accessory should reidentify using the `Identify` command.

Figure 5-1 (page 56) shows the command traffic for device identification when using the UART serial port link.

Figure 5-1 Command traffic for device identification



A device must reidentify itself if it receives a "Command 0x00: RequestIdentify" (page 69) command from the iPod.

Once accessory packet transmission has begun, the maximum time between transmitted data bytes is 25 milliseconds. If the inter-character delay exceeds 25 ms, the iPod discards any packet characters already received, plus any remaining characters received received before the start of the next valid packet. The iPod may exceed the 25 ms inter-character timing requirement in its outgoing packets when under heavy system load situations.

One known limitation exists when waking an iPod from sleep: the iPod UART is not available for the first few milliseconds after waking from sleep. If an external device sends a packet to the iPod while it is asleep, the first packet will be lost. Accessories should follow the steps below to wake up an iPod and ensure that the first packet is not lost:

1. Send a sync byte; this should wake up the iPod.
2. Wait for 20 ms.
3. Send the command packet with sync byte.

Note: Supporting the 3G iPod requires special design considerations. See "Interfacing With the 3G iPod" (page 255) .

iAP Signaling and Initialization Using the USB Port Link

When using iAP over USB, initialization involves the USB host detecting the attached iPod and setting its iUI configuration. Upon completion of this process, the host may send General lingo iAP commands to the iPod using the HID interface. USB host access to the iAP accessory lingoes is only enabled after the host has identified itself and been authenticated by the iPod. Prior to authentication, only the General lingo commands are available to unauthenticated hosts.

To begin the authentication process, the host must send "Command 0x13: IdentifyDeviceLingoes" (page 84), identifying the lingoes it supports. The host may optionally request authentication using the authentication Options byte. If immediate authentication is requested, the device should be ready to respond to authentication requests from the iPod after the iPod acknowledges this command. Refer to "Authentication" (page 57) for more information. After USB hosts have initialized the iPod and set its iUI configuration, they should follow steps shown in Figure 5-4 (page 62) to start the authentication process.

In addition to identifying itself at plug-in, a host may need to reidentify during an iAP session if the iPod so requests. If the host receives "Command 0x00: RequestIdentify" (page 69), it must send an `IdentifyDeviceLingoes` command and repeat the authentication process.

Note: USB hosts do not have to reconfigure iUI when responding to the `RequestIdentify` command from the iPod.

Authentication

Authentication is a mechanism used by the iPod to verify whether an attached device is an authorized accessory and by an accessory to authenticate the iPod, if desired. Certain functionality on the iPod is accessible only after a device has been authenticated as an authorized accessory. This functionality is summarized in Table 5-1 (page 58) and includes the use of any accessory lingo command over the USB transport. General lingo commands over USB are free. After the device identification process, authentication may be independently initiated from either the iPod or the device. In the identify and authentication sequences, the accessory should check the ACK status for all commands. If a failure status is returned, the process has failed and should be retried and/or aborted.

Because device authentication can take a significant amount of time (up to 75 seconds for Level V2 authentication), it is performed as a background process. A device connected to an iPod is granted a minimum amount of non-critical functionality immediately, and both a timer and a retry counter are started to guarantee that the authentication process will conclude.

Note: To participate in the authentication process, an accessory device must contain an authentication coprocessor provided by Apple.

Table 5-1 Lingo commands requiring authentication

Lingoes	UART port link	USB port link
Lingo 0x00: General	No (0x00-0x19, 0x20-0x2B) Yes (0x1A-0x1F)	No (0x00-0x19, 0x20-0x2B) Yes (0x1A-0x1F)
Lingo 0x01: Microphone	No (0x00–0x03) Yes (0x04–0x0B)	Yes
Lingo 0x02: Simple Remote	No (0x00) Yes (0x01–0x04)	Yes
Lingo 0x03: Display Remote	No (0x00–0x07; 0x1A–0x1E) Yes (0x08–0x19; 0x1F–0x20)	Yes
Lingo 0x04: Extended Interface	No (0x0000–0x003A) Yes (0x003B)	Yes
Lingo 0x05: RF Transmitter	No	Yes
Lingo 0x06: USB Host Control	Yes	N/A
Lingo 0x07: RF Tuner	Yes	Yes
Lingo 0x08: Accessory Equalizer	Yes	Yes
Lingo 0x09: Reserved	N/A	N/A
Lingo 0x0A: Digital Audio	Yes	Yes
Lingo 0x0B: Reserved	N/A	N/A
Lingo 0x0C: Storage	Yes	Yes
Lingoes 0x0D–0x1F: Reserved	N/A	N/A

Levels of Device Authentication

The iAP supports three levels of security in the process by which an iPod may authenticate an accessory device connected to it:

- **None.** Older iPod models through the iPod mini do not support authentication. They operate freely with simple accessories that do not require authentication, but they cannot work with newer or more sophisticated accessories whose range of functionality requires authentication. For details, see Table 2-1 (page 27) .

- **Level V1.** Version 1.00 authentication is based on public and private keys, where each accessory has a private key and every iPod has the associated public key. The accessory authentication process is a part of the iAP General lingo (0x00) command protocol and is controlled by the iPod internal software. Devices identify themselves to the iPod as speaking specific lingoes and supporting authentication. The iAP queries the device's authentication information, sends a challenge to the device, and verifies that the device responds to the challenge correctly.
- **Level V2.** Version 2.00 authentication is based on standard X.509 Version 3 certificates. Information about this standard can be found at the IETF website: http://tools.ietf.org/html/rfc3280. Each certificate is generated and signed by a recognized certificate authority (CA) and has a unique serial number. Level V2 authentication uses certificate classes to identify the type of accessory being authenticated, as shown in Table 5-2 (page 59) .

IMPORTANT: Current iPod models support Level V1 authentication only as a legacy technology, to make them compatible with existing accessory devices. All new accessory designs must use Level V2 authentication.

Table 5-2 iPod X.509 authentication certificate classes

Class ID	Description
1	Automotive accessories that support the iAP Digital Audio lingo (Lingo 0x0A)
2	Nonautomotive accessories that do not support the Digital Audio lingo
3	Nonautomotive accessories that support the Digital Audio lingo

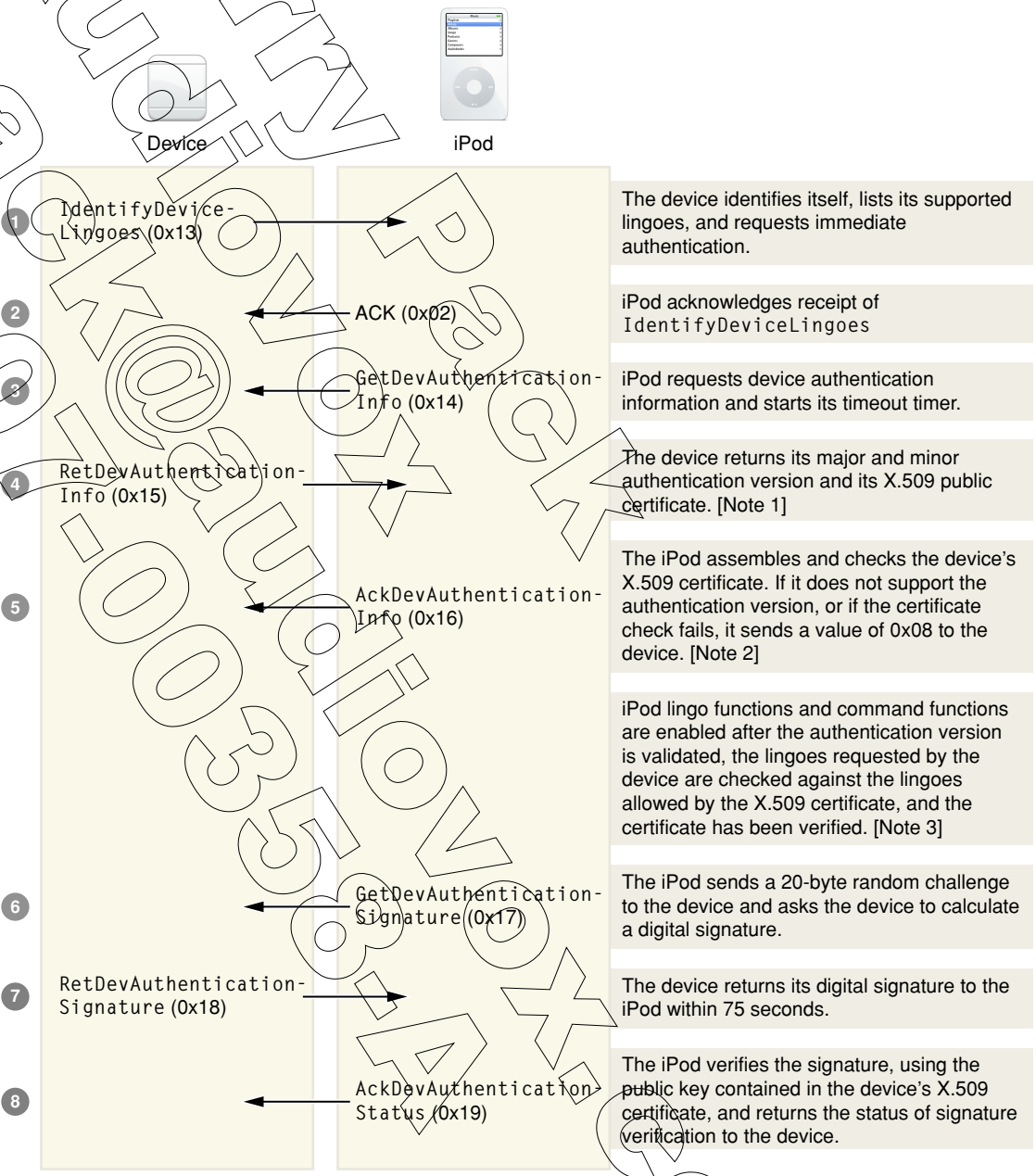
iPod Authentication of Device

The process of authenticating the device is initiated by the iPod, based on the settings used in the `IdentifyDeviceLingoes` command (see "Command 0x13: IdentifyDeviceLingoes" (page 84)). The authentication options also allow a device to request authentication of an iPod, as described in "Device Authentication of iPod" (page 62) .

Figure 5-2 (page 60) summarizes the authentication process, using Level V2 authentication. This is the authentication process that all new devices should use. Command details can be found in "General Lingo Command Details" (page 69) .

For legacy purposes, Figure 5-3 (page 61) shows the process implemented by some existing devices, using Level V1 authentication. This process is not supported for new device designs.

Figure 5-2 iPod authentication of device



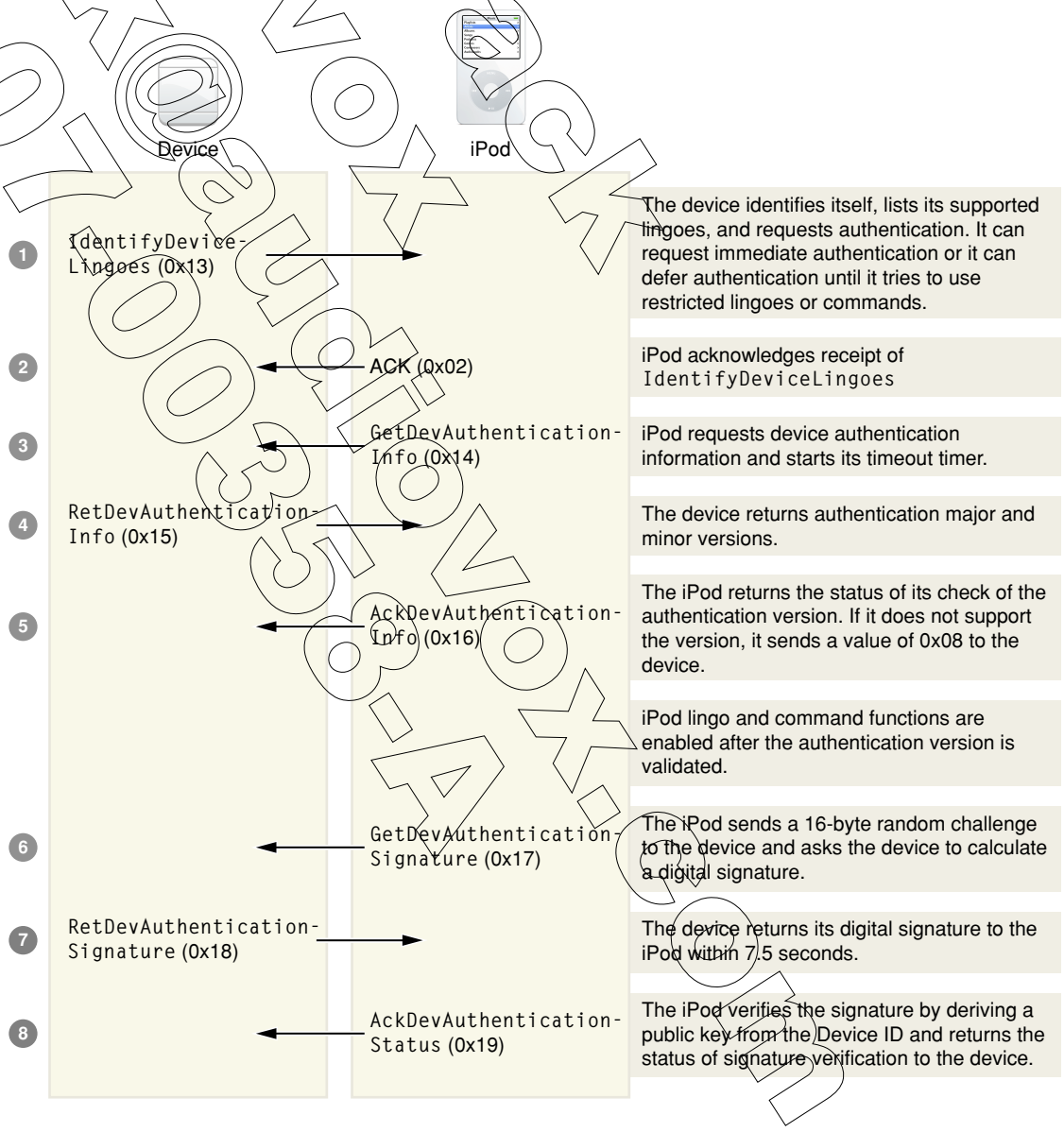
If authentication fails at any point in the level V2 process and the retry count is exhausted, the iPod may display a "Device Not Supported" message to the user.

Notes:

1. Not more than 500 certificate bytes may be sent at once. If the X.509 certificate is larger than 500 bytes, the device must divide it into sections, each smaller than 500 bytes, for reassembly by the iPod. The iPod will send a general lingo `ACK` command for each certificate section up to, but not including, the last one.

2. The iPod parses the X.509 certificate into an allowed lingoes mask. It uses the mask to confirm that the device's identified lingoes are allowed by the certificate. If the device requests lingoes not allowed by the certificate, authentication fails. The iPod also verifies that the certificate is valid.
3. The iPod and the device can perform noncritical operations while background authentication is in progress. The command timeout counter and retry counter are not reset until authentication is complete or has failed.

Figure 5-3 Legacy device authentication



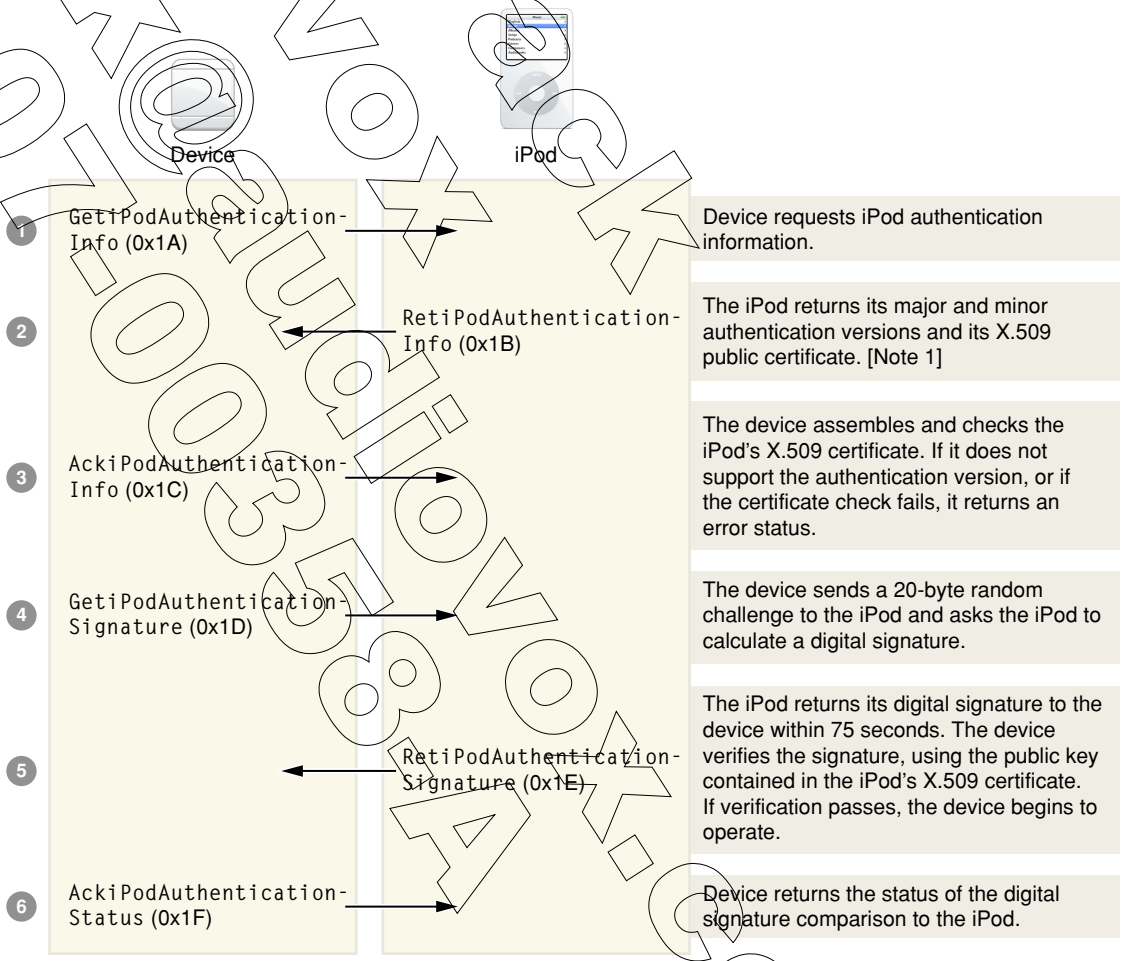
If authentication fails at any point in the level V1 process and the retry count is exhausted, the iPod may display a "Device Not Supported" message to the user.

Device Authentication of iPod

The accessory can initiate a process to authenticate the iPod, any time after it has been authenticated by the iPod. This process is summarized in Figure 5-4 (page 62) . See "General Lingo Command Summary" (page 65) for information on the individual commands.

Note: Device authentication of the iPod is currently supported only for authentication Level V1.

Figure 5-4 Device authentication of the iPod



Note:

1. Not more than 500 certificate bytes may be sent at once. If the X.509 certificate is larger than 500 bytes, the iPod divides it into sections, each smaller than 500 bytes, for reassembly by the device. The device does not acknowledge any certificate sections sent by the iPod until the complete certificate has been received and verified.

Command Packet Formats

This section describes the general format for iAP packets. Use the small packet format for payloads up to 255 bytes. Use the large packet format for payloads greater than 255 bytes.

Small Packet Format

For command packets whose payloads are 255 bytes or less, use the small packet format. The small packet format is shown in Table 5-3 (page 63) .

Table 5-3 Small packet format

Byte number	Value	Meaning
0x00	0xFF	Sync byte
0x01	0x55	Packet start byte
0x02	0x*NN*	Packet payload length
0x03	0x*NN*	Lingo ID
0x04	0x*NN*	Command ID
0x05…0x*NN*	0x*NN*	Command data
(last byte)	0x*NN*	Packet payload checksum

Note that the command ID and command data format for packets with currently unspecified lingoes may not follow the format indicated here (1 byte command ID, 0x*N* bytes command data). Also note that a packet payload length of 0x00 is not valid for the small packet format; it is reserved as a marker for the large packet format.

Large Packet Format

For command packets whose payloads are between 256 bytes and 65535 bytes in length, use the large packet format. The large packet format is shown in Table 5-4 (page 63) .

Table 5-4 Large packet format

Byte number	Value	Meaning
0x00	0xFF	Sync byte
0x01	0x55	Packet start byte
0x02	0x00	Packet payload length marker
0x03	0x*NN*	Packet payload length (bits 15:8)

Byte number	Value	Meaning
0x04	0x*NN*	Packet payload length (bits 7:0)
0x05	0x*NN*	Lingo ID
0x06	0x*NN*	Command ID
0x07…0x*NN*	0x*NN*	Command data
(last byte)	0x*NN*	Packet payload checksum

Packet Details

The sync byte (0xFF) is not considered part of the packet. It is sent merely to facilitate automatic baud rate detection and correction when using a UART serial port link and, in some cases, to wake up the iPod. It is not necessary to send the sync byte when using USB as a link.

The packet payload length is the number of bytes in the packet, not including the sync byte, packet start byte, packet payload length byte, or packet payload checksum byte. That is, it is the length of the command ID, lingo, and command data. Thus, the packet payload data length for a `RequestIdentify` command would be 0x02. The Lingo ID specifies the broad category that the communication falls under. The Command ID is a more specific indication of the significance of the packet and is interpreted differently depending on the Lingo ID.

Note: Unless otherwise specified, all data units larger than bytes must be transferred in a big-endian order; that is, 32 bits should be sent as bits 31:24, followed by bits 23:16, and so forth. Similarly, 16 bits should be sent as bits 15:8, followed by bits 7:0. This includes the 16-bit large packet format payload length: the high byte of the length is sent first, followed by the low byte of the length.

The sum of all the bytes from the packet payload length (or marker, if applicable) through the packet payload checksum is 0x00. The checksum should be calculated appropriately, by adding the bytes together as signed 8-bit values, discarding any signed 8-bit overflow, and then negating the sum to create the signed 8-bit checksum byte. All packets received with a nonzero checksum are presumed to be corrupted and will be discarded.

Lingo 0x00: General Lingo

The General lingo is intended for housekeeping commands and must be supported by all devices. In addition to the General lingo, external devices implement function-specific lingoes. For example, a microphone device attached to the mono microphone input of the 9-pin Audio/Remote connector on the iPod uses the Microphone lingo (0x01). The Simple Remote lingo (0x02) is used by Apple's simple in-line remote control. An external RF transmitter device uses the RF Transmitter lingo (0x05).

General Lingo Command Summary

Table 5-5 (page 65) gives a summary of all commands in the General lingo, including the command ID, the length of the associated data, the first version of the General lingo protocol in which the command is supported, and what device authentication (if any) is required to use the command.

Table 5-5 General lingo commands

Command	ID	Data length	Protocol Version	Authentication Required
RequestIdentify	0x00	0x00	All	None
Identify	0x01	0x01	All	None
ACK	0x02	0x02 or 0x06	1.00	None
RequestRemoteUIMode	0x03	0x00	1.00	None
ReturnRemoteUIMode	0x04	0x01	1.00	None
EnterRemoteUIMode	0x05	0x00	1.00	None
ExitRemoteUIMode	0x06	0x00	1.00	None
RequestiPodName	0x07	0x00	1.00	None
ReturniPodName	0x08	0x*NN*	1.00	None
RequestiPodSoftwareVersion	0x09	0x00	1.00	None
ReturniPodSoftwareVersion	0x0A	0x03	1.00	None
RequestiPodSerialNum	0x0B	0x00	1.00	None
ReturniPodSerialNum	0x0C	0x*NN*	1.00	None
RequestiPodModelNum	0x0D	0x00	1.00	None
ReturniPodModelNum	0x0E	0x*NN*	1.00	None
RequestLingoProtocolVersion	0x0F	0x01	1.00	None
ReturnLingoProtocolVersion	0x10	0x03	1.00	None
Reserved	0x11–0x12	N/A	N/A	N/A
IdentifyDeviceLingoes	0x13	0x0C	1.01	None
GetDevAuthenticationInfo	0x14	0x00	1.01	None
RetDevAuthenticationInfo	0x15	0x*NN*	1.01	None
AckDevAuthenticationInfo	0x16	0x01	1.01	None

Command	ID	Data length	Protocol Version	Authentication Required
GetDevAuthenticationSignature	0x17	0x11 or 0x15	1.01	None
RetDevAuthenticationSignature	0x18	0x*NN*	1.01	None
AckDevAuthenticationStatus	0x19	0x01	1.01	None
GetiPodAuthenticationInfo	0x1A	0x00	1.01	Level V2
RetiPodAuthenticationInfo	0x1B	0x*NN*	1.01	Level V2
AckiPodAuthenticationInfo	0x1C	0x01	1.01	Level V2
GetiPodAuthentication-Signature	0x1D	0x*NN*	1.01	Level V2
RetiPodAuthentication-Signature	0x1E	0x*NN*	1.01	Level V2
AckiPodAuthenticationStatus	0x1F	0x01	1.01	Level V2
Reserved	0x20–0x22	N/A	N/A	N/A
NotifyiPodStateChange	0x23	0x01	1.02	None
GetiPodOptions	0x24	0x00	1.05	None
RetiPodOptions	0x25	0x08	1.05	None
Reserved	0x26	N/A	N/A	N/A
GetAccessoryInfo	0x27	0x*NN*	1.04	None
RetAccessoryInfo	0x28	0x*NN*	1.04	None
GetiPodPreferences	0x29	0x01	1.05	None
RetiPodPreferences	0x2A	0x02	1.05	None
SetiPodPreferences	0x2B	0x03	1.05	None
Reserved	0x2C–0xFF	N/A	N/A	N/A

To identify itself, the accessory should send an `IdentifyDeviceLingoes` command (0x13). The only exception is accessories that support the 3G iPod, which must use the deprecated "Command 0x01: Identify" (page 69) .

The iPod may send a `RequestIdentify` command to the device to ask it to reidentify itself. There is currently no data defined for this command.

The Identify or IdentifyDeviceLingoes command returned in response to a RequestIdentify packet does not need to have the extra sync bytes and delays used during the device startup process (described in "Device Signaling and Initialization" (page 55)). Refer to "Command 0x01: Identify" (page 69) and "Command 0x13: IdentifyDeviceLingoes" (page 84) for detailed information on these commands.

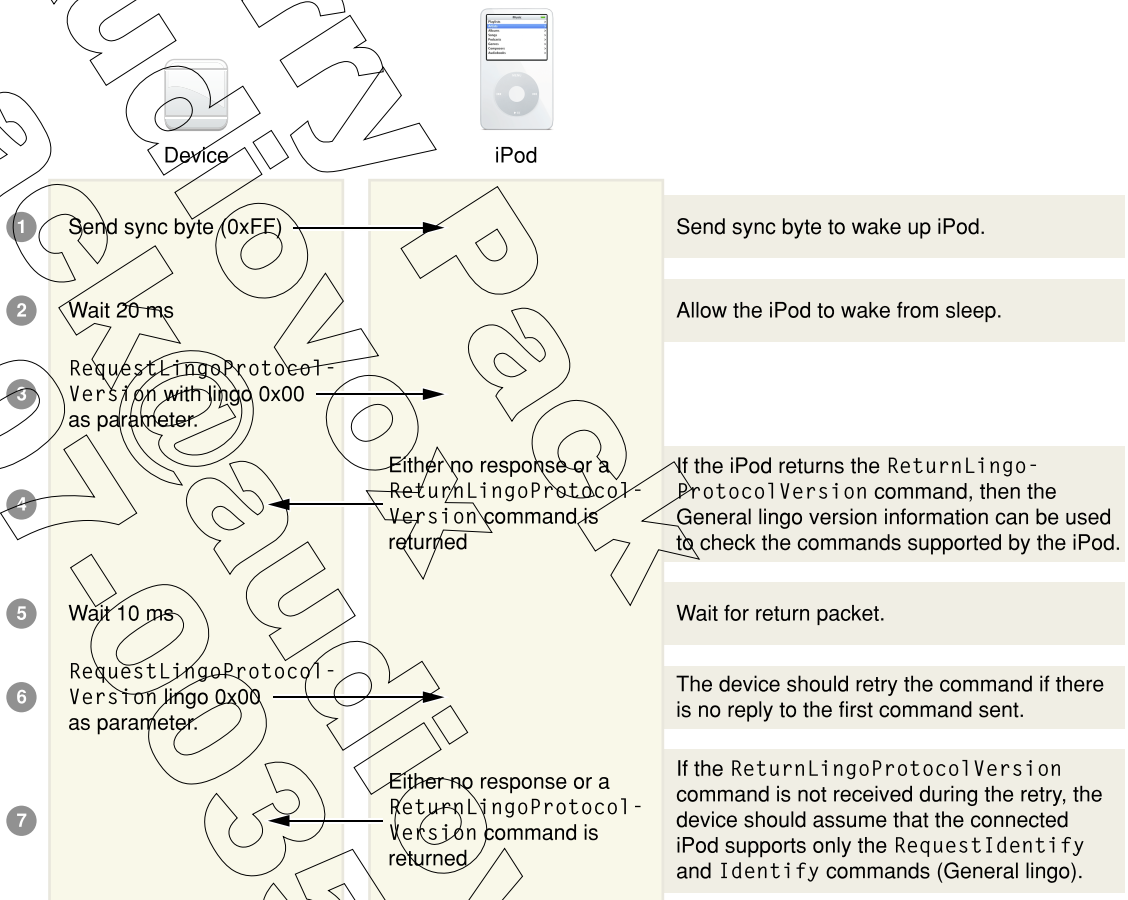
The remaining General lingo commands can be used to obtain general information from the iPod. These commands allow the device to request the name, serial number, model number, and software version number of the iPod. The RequestLingoProtocolVersion command allows a device to query the iPod for the lingo protocol versions of all supported lingoes on the iPod. The ACK command is used by the iPod to report command error conditions and has an ACK pending feature to notify the requesting device how long to wait for responses to certain commands.

An accessory device should send a RequestLingoProtocolVersion command, passing lingo 0x00 as the parameter, to determine which features the connected iPod supports. Refer to "Protocol Features and Availability" (page 43) for features supported in each version of the General lingo. If the iPod does not respond to the RequestLingoProtocolVersion command, the device should retry the command once. If no response is received the second time, the device should assume that the connected iPod supports the only RequestIdentify and Identify General lingo commands.

Note: Supporting the 3G iPod requires special design considerations. See "Interfacing With the 3G iPod" (page 255).

Figure 5-5 (page 68) shows the sequence of the events that should be used to test for the full set of General lingo commands. This assumes the accessory device is connected and has already been through the identification process.

Figure 5-5 Testing for the General lingo over the UART serial port link



History of the General lingo protocol

Table 5-6 (page 68) lists changes introduced with each version of the General lingo protocol.

Table 5-6 General lingo revision history

Lingo version	Command changes	Features
No version	Add: 0x00, 0x01	Request identification, identify as single lingo device
1.00	Add: 0x02, 0x07–0x10	ACK response, request iPod name, software version, serial number, and model number; support for 38400 and 57600 baud rates
1.01	Add: 0x13–0x1F	Identify as multilingo device, authentication V1.00 process
1.02	Add: 0x23	Notify device of iPod state changes (Sleep/Wake/Hibernate)
1.03	None	(Internal code restructuring)

Lingo version	Command changes	Features
1.04	Add: 0x27, 0x28	Get/return accessory device information, authentication V2.00 process
1.05	Add: 0x24–0x25, 0x29–0x2B	Video browsing preferences
1.06	None	Line-out preferences, two-way Level V2 authentication
1.07	None	Additional video preferences

General Lingo Command Details

This section describes the General lingo commands and their packet formats.

Command 0x00: RequestIdentify

Direction: iPod → Device

The iPod sends this command to prompt accessories to reidentify themselves. If an accessory receives this command, it should respond with "Command 0x13: IdentifyDeviceLingoes" (page 84) . The only exception is accessories that support the 3G iPod, which must respond with the deprecated "Command 0x01: Identify" (page 69) .

Table 5-7 `RequestIdentify` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x00	Lingo ID: General lingo. All devices must support this lingo.
4	0x00	Command: `RequestIdentify`
5	0xFE	Checksum

Command 0x01: Identify

Direction: Device → iPod

This command is deprecated. It should be used only by accessories that need to support the 3G iPod. Supporting the 3G iPod requires other special design considerations; see "Interfacing With the 3G iPod" (page 255) .

Accessories that use this command send it to notify the iPod that an accessory has been attached and to register the lingo it supports. Accessories should identify at boot time and any time they receive "Command 0x00: RequestIdentify" (page 69) from the iPod. The iPod does not send an `ACK` command in response.

Note: The `Identify` command should be used only over the UART serial port link.

Accessories should follow the identification guidelines in "Packet Signaling and Initialization Using the UART Serial Port Link" (page 55) and "iAP Signaling and Initialization Using the USB Port Link" (page 57) to guarantee they have established communication with the iPod when using this command. Accessory devices that support more than one lingo (not including the General lingo) or that plan to use the USB transport should use "Command 0x13: IdentifyDeviceLingoes" (page 84) .

Note: The `Identify` command disables all but free lingoes on the current port. For serial ports, this means lingoes 0x00–0x03 and 0x05 may be used, not including unauthenticated commands; the USB port will be able to use only the general lingo, 0x00. Lingo conflict checking is added to ensure that single-instance lingoes, such as display remote, are usable on only one port at a time. An identification failure results in the device speaking only the General lingo.

Table 5-8 `Identify` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x01	Command: `Identify`
5	0x0N	Supported lingo. For example, if the device supports the Simple Remote lingo, this byte should be 0x02.
6	0x*NN*	Checksum

The `Identify` command has facilities for RF Transmitter devices to draw more than 5 mA power from the iPod. The `Identify` command sent by an RF transmitter device contains the supported lingo and the optional power bitfields described in Table 5-10 (page 71) . These bits define the power requirements of the RF Transmitter device. Table 5-9 (page 70) shows the format of an `Identify` command for an RF transmitter device.

Table 5-9 `Identify` packet for RF transmitter devices

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x06	Packet payload length
3	0x00	Lingo ID: General lingo
4	0x01	Command: `Identify`
5	0x05	Supported lingo: RF Transmitter lingo
6	0x00	Reserved: set to 0x00
7	0x02	Number of valid bits in the RF transmitter power option flag
8	0x0*N*	Option flag bits. See Table 5-10 (page 71) .
9	0x*NN*	Checksum

The option flag byte consists of bitfields. At this time, the most significant 6 bits of the option flags are reserved and should be zero. Bit 1 is defined for Apple use only and must be zero. Bit 0 should be 1 if the FM transmitter requires more than 5 mA during active transmission. This will be the case if the transmitter is powered by the iPod.

Table 5-10 Power option bits for RF transmitter devices

Bit	Description
0	RF transmitter iPod power consumption requirements. Possible values are: 0 = device requires 5 mA or less power from iPod at all times 1 = device requires more than 5 mA power from iPod (up to 100mA max) during playback operation
1	Reserved. This bit must be set to 0 by external devices.
2–7	Reserved. Set to 0.

Command 0x02: ACK

Direction: iPod → Device

The iPod sends the `ACK` command to notify the device of command completion status and errors. The `ACK` command may come in one of two forms, depending on the status being returned. For any status other than command pending, the normal ACK packet structure shown in Table 5-11 (page 71) is used.

Table 5-11 `ACK` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x04	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x02	Command: `ACK`
5	0x*NN*	Command result status. Possible values are: 0x00 = Success (OK) 0x01 = ERROR: Unknown database category 0x02 = ERROR: Command failed 0x03 = ERROR: Out of resources 0x04 = ERROR: Bad parameter 0x05 = ERROR: Unknown ID 0x06 = Command Pending. See Table 5-12 (page 72) . 0x07 = ERROR: Not authenticated 0x08 = ERROR: Bad authentication version 0x09 Reserved 0x0A ERROR: Certificate invalid 0x0B ERROR: Certificate permissions invalid 0x0C – 0x10 Reserved 0x11 = ERROR: Invalid accessory resistor ID value 0x12 – 0xFF Reserved
6	0x*NN*	The ID of the command being acknowledged
7	0x*NN*	Checksum

If the status returned by the `ACK` command is Command Pending, an additional field is added to the ACK packet that represents the amount of time, in milliseconds, that a device should wait to receive the final packet indicating that the current command completed or returned an error status.

After receiving a Command Pending ACK, the device should wait for up to the specified number of milliseconds for a final ACK response. If no final ACK packet is received before the specified amount of time expires, the device should retry the command.

Table 5-12 `ACK` packet with Command Pending status

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x08	Length of packet payload
3	0x00	Lingo ID: General lingo

Byte number	Value	Comment
4	0x02	Command: `ACK`
5	0x06	Command result status: Command Pending
6	0x*NN*	The ID of the command being acknowledged.
7	0x*NN*	Maximum amount of time to wait for pending response, in milliseconds (bits 31:24).
8	0x*NN*	Maximum pending wait, in milliseconds (bits 23:16)
9	0x*NN*	Maximum pending wait, in milliseconds (bits 15:8)
10	0x*NN*	Maximum pending wait, in milliseconds (bits 7:0)
11	0x*NN*	Checksum

Command 0x03: RequestRemoteUIMode

Direction: Device →iPod

The device requests the Extended Interface mode from the iPod. The iPod responds with "Command 0x04: ReturnRemoteUIMode" (page 73) . This command may be used only if the accessory requests Lingo 0x04, using `Identify` or `IdentifyDeviceLingoes`.

Table 5-13 `RequestRemoteUIMode` packet

Byte number	Value	Meaning
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet
2	0x02	Packet payload length
3	0x00	Lingo ID: General lingo
4	0x03	Command ID: `RequestRemoteUIMode`
5	0xFB	Packet payload checksum byte

Command 0x04: ReturnRemoteUIMode

Direction: iPod → Device

The iPod returns the current operating mode of the iPod UI. This is either Standard UI mode or Extended Interface mode. If the returned mode byte is nonzero (true), the iPod is in Extended Interface mode. This command may be used only if the accessory requests Lingo 0x04, using `Identify` or `IdentifyDeviceLingoes`.

Table 5-14 `ReturnRemoteUIMode` packet

Byte number	Value	Meaning
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet
2	0x03	Packet payload length
3	0x00	Lingo ID: General lingo
4	0x04	Command ID: `ReturnRemoteUIMode`
5	0xNN	Mode byte. If nonzero (true), the iPod is in Extended Interface Mode. If zero (false), the iPod is in Standard UI mode.
6	0xNN	Packet payload checksum byte

Command 0x05: EnterRemoteUIMode

Direction: Device → iPod

The device sends this command to the iPod to force it to enter the Extended Interface mode. If the iPod is already in the Extended Interface mode, it immediately returns a General lingo `ACK` command packet, notifying the user that the command was successful. This command may be used only if the accessory requests Lingo 0x04, using `Identify` or `IdentifyDeviceLingoes`.

If the iPod needs to switch modes, it returns a General lingo ACK Pending command packet, informing the device how long it will take the iPod to switch modes. This is followed by an ACK packet notifying the device that the iPod successfully changed modes. Devices should honor the timeout returned by the ACK pending before assuming the original command has failed.

If audio is playing when the iPod enters Extended Interface mode, the iPod will pause playback. If video is playing, the iPod will stop playback.

Table 5-15 `EnterRemoteUIMode` packet

Byte number	Value	Meaning
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet
2	0x02	Packet payload length
3	0x00	Lingo ID: General lingo
4	0x05	Command ID: `EnterRemoteUIMode`
5	0xF9	Packet payload checksum byte

Command 0x06: ExitRemoteUIMode

Direction: Device → iPod

The device sends this command to the iPod to force it to exit the Extended Interface mode. If the iPod is already in the Standard UI mode, it immediately returns a General lingo `ACK` command packet, notifying the user that the command was successful. This command may be used only if the accessory requests Lingo 0x04, using `Identify` or `IdentifyDeviceLingoes`.

If the iPod needs to switch modes, it sends a General lingo ACK Pending command packet informing the device how long it will take the iPod to switch modes. This is followed by an ACK packet notifying the device that the iPod successfully changed modes. Devices should honor the timeout returned by the ACK pending before assuming the original command has failed.

Table 5-16 `ExitRemoteUIMode` packet

Byte number	Value	Meaning
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet
2	0x02	Packet payload length
3	0x00	Lingo ID; General lingo
4	0x06	Command ID: `ExitRemoteUIMode`
5	0xF8	Packet payload checksum byte

Command 0x07: RequestiPodName

Direction: Device → iPod

Retrieves the name of the iPod. The iPod responds with a "Command 0x08: ReturniPodName" (page 76) command containing the name of the iPod as a null-terminated UTF-8 character array.

Table 5-17 `RequestiPodName` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x07	Command: `RequestiPodName`
5	0xF7	Checksum

Command 0x08: ReturniPodName

Direction: iPod → Device

The iPod sends this command in response to the "Command 0x07: RequestiPodName" (page 75) message from the device. The iPod name is encoded as a null-terminated UTF-8 character array. If the iPod name has not been modified by the user, it is returned as "iPod".

Note: Starting with version 1.02 of the General lingo, the `ReturniPodName` command on Windows-formatted iPods returns the iTunes name of the iPod instead of the Windows volume name.

Table 5-18 `ReturniPodName` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x08	Command: `ReturniPodName`
5	0x*NN*...	The name of the iPod as a null-terminated UTF-8 character array.
(last byte)	0x*NN*	Checksum

Command 0x09: RequestiPodSoftwareVersion

Direction: Device → iPod

Retrieves the software version information for the iPod. The iPod responds with a "Command 0x0A: ReturniPodSoftwareVersion" (page 77) containing the major, minor, and revision version numbers.

Note: This command requests the iPod software version and not the version of a particular lingo protocol.

Table 5-19 `RequestiPodSoftwareVersion` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x09	Command: `RequestiPodSoftwareVersion`

Byte number	Value	Comment
5	0xF5	Checksum

Command 0x0A: ReturniPodSoftwareVersion

Direction: iPod → Device

The iPod sends this command in response to the "Command 0x09: RequestiPodSoftwareVersion" (page 76) message from the device. The iPod returns each version number as an individual byte, with the major version number sent first. For example, if the major, minor, and revision bytes are returned as 0x01, 0x02, and 0x03, the iPod software version number is 1.02.03.

Table 5-20 `ReturniPodSoftwareVersion` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x05	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x0A	Command: `ReturniPodSoftwareVersion`
5	0x*NN*	iPod major version number.
6	0x*NN*	iPod minor version number.
7	0x*NN*	iPod revision version number.
8	0x*NN*	Checksum

Command 0x0B: RequestiPodSerialNum

Direction: Device → iPod

Retrieves the serial number string of the iPod. The iPod responds with a "Command 0x0C: ReturniPodSerialNum" (page 78) containing the serial number as a null-terminated UTF-8 character array.

Table 5-21 `RequestiPodSerialNum` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload

Byte number	Value	Comment
3	0x00	Lingo ID: General lingo
4	0x0B	Command: `RequestiPodSerialNum`
5	0xF3	Checksum

Command 0x0C: ReturniPodSerialNum

Direction: iPod → Device

The iPod sends this command in response to the "Command 0x0B: RequestiPodSerialNum" (page 77) message from the device. The iPod serial number is encoded as a null-terminated UTF-8 character array.

Table 5-22 `ReturniPodSerialNum` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x0C	Command: `ReturniPodSerialNum`
5	0x*NN*...	The iPod serial number, as a null-terminated UTF-8 character array.
(last byte)	0x*NN*	Checksum

Command 0x0D: RequestiPodModelNum

Direction: Device → iPod

Retrieves model information for the iPod. The iPod responds with a "Command 0x0E: ReturniPodModelNum" (page 79) containing the model number of the iPod as a 32-bit integer (model ID) and as a null-terminated UTF-8 character array. If an internal memory error occurs while the iPod is processing this command, the iPod returns an `ACK` command with the Command Failed error status. The returned model number can be used to determine what iPod hardware has been connected. See "Command 0x0E: ReturniPodModelNum" (page 79) for details.

Table 5-23 `RequestiPodModelNum` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x02	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x0D	Command: `RequestiPodModelNum`
5	0xF1	Checksum

Command 0x0E: ReturniPodModelNum

Direction: iPod → Device

The iPod sends this command in response to the "Command 0x0D: RequestiPodModelNum" (page 78) message from the device. The iPod model number is encoded as 32-bit integer (model ID) and as a null-terminated UTF-8 character array. A device should use the model information to determine what type of iPod has been connected and, with the version returned from "Command 0x0F: RequestLingoProtocolVersion" (page 83), what features of iAP are available.

Table 5-24 `ReturniPodModelNum` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x0E	Command: `ReturniPodModelNum`
5	0x*NN*	iPod Model ID (bits 31:24). See Table 5-25 (page 80) .
6	0x*NN*	iPod Model ID (bits 23:16)
7	0x*NN*	iPod Model ID (bits 15:8)
8	0x*NN*	iPod Model ID (bits 7:0)
9 … *N*	0x*NN*…	The iPod model number as a null-terminated UTF-8 character array. See Table 5-26 (page 80) and Table 5-27 (page 81) .
(last byte)	0x*NN*	Checksum

Table 5-25 (page 80) shows the existing iPod model IDs. Table 5-26 (page 80) and Table 5-27 (page 81) list the model number strings returned for each iPod model.

Table 5-25 iPod model IDs

iPod model ID	iPod hardware
0x0003*NNNN*	3G iPod. This is the white iPod with 4 buttons above a white click wheel.
0x0004*NNNN*	iPod mini: original 4 GB model.
0x0005*NNNN*	4G iPod. This is the white iPod with a gray click wheel.
0x0006*NNNN*	iPod photo
0x0007*NNNN*	2nd generation iPod mini (models M9800LL – M9807LL, 4 GB and 6 GB)
0x000B*NNNN*	5G iPod
0x000C*NNNN*	iPod nano
0x0010*NNNN*	2G iPod nano
0x0011*NNNN*	iPhone
0x0013*NNNN*	iPod classic
0x0014*NNNN*	3G iPod nano
0x0015*NNNN*	iPod touch

Table 5-26 iPod model number strings M8948LL-M9974LL and P8948LL-P9830LL

iPod model ID strings	iPod hardware
M8948LL, P8948LL	3G iPod: 30 GB
M8976LL, P8976LL	3G iPod: 10 GB
M9160LL, P9160LL	iPod mini: 4 GB silver
M9244LL, P9244LL	3G iPod: 20 GB
M9245LL, P9245LL	3G iPod: 40 GB
M9268LL, P9268LL	4G iPod: 40 GB
M9282LL, P9282LL, P9659LL, P9660LL, P9661LL, P9662LL	4G iPod: 20 GB
M9434LL, P9434LL	iPod mini: 4 GB green
M9435LL, P9435LL	iPod mini: 4 GB pink
M9436LL, P9436LL	iPod mini: 4 GB blue
M9437LL, P9437LL	iPod mini: 4 GB gold
M9460LL, M8946LL, P8946LL, P9460LL	3G iPod: 15 GB

iPod model ID strings	iPod hardware
M9575LL	4G iPod: 20 GB (HP-branded)
M9576LL	4G iPod: 40 GB (HP-branded)
M9585LL, P9585LL	iPod photo: 40 GB
M9586LL, P9586LL	iPod photo: 60 GB
M9787LL	4G iPod: 20 GB black (U2 special edition)
M9800LL, P9800LL	2nd Generation (2G) iPod mini: 4 GB silver
M9801LL, P9801LL	2G iPod mini: 6 GB silver
M9802LL, P9802LL	2G iPod mini: 4 GB blue
M9803LL, P9803LL	2G iPod mini: 6 GB blue
M9804LL, P9804LL	2G iPod mini: 4 GB pink
M9805LL, P9805LL	2G iPod mini: 6 GB pink
M9806LL, P9806LL	2G iPod mini: 4 GB green
M9807LL, P9807LL	2G iPod mini: 6 GB green
M9829LL, P9829LL	2nd Generation (2G) iPod photo: 30 GB
M9830LL, P9830LL	2G iPod photo: 60 GB
M9872LL	2G iPod photo: 30 GB (HP-branded)
M9873LL	2G iPod photo: 60 GB (HP-branded)
M9973LL	2G iPod mini: 4 GB silver (HP-branded)
M9974LL	2G iPod mini: 6 GB silver (HP-branded)

Table 5-27 iPod model number strings MA002LL/PA002LL and higher

iPod model ID strings	iPod hardware
MA002LL, PA002LL, PA148LL, PA323LL, MA444LL, PA444LL	5G iPod: 30 GB white
MA003LL, PA003LL, PA150LL	5G iPod: 60 GB white
MA004LL, PA004LL, PA115LL	iPod nano: 2 GB white
MA005LL, PA005LL, PA116LL	iPod nano: 4 GB white
MA079LL, PA079LL	2G iPod photo: 20 GB

iPod model ID strings	iPod hardware
MA080LL	2G iPod photo: 20 GB (HP-branded)
MA099LL, PA099LL, PA100LL	iPod nano: 2 GB black
MA107LL, PA107LL, PA108LL	iPod nano: 4 GB black
MA127LL	2G iPod photo: 20 GB black (U2 special edition)
MA146LL, PA146LL, PA149LL, MA446LL, PA446LL	5G iPod: 30 GB black
MA147LL, PA147LL, PA151LL	5G iPod: 60 GB black
MA215LL	2G iPod photo: 20 GB (Harry Potter special edition)
MA253LL	5G iPod: 30 GB white (Harry Potter special edition)
MA305LL	5G iPod: 30 GB black (Harry Potter special edition)
MA350LL, PA350LL, PA351LL	iPod nano: 1 GB white
MA352LL, PA352LL, PA353LL	iPod nano: 1 GB black
MA426LL, PA426LL, PA427LL	2G iPod nano: 4 GB silver
MA428LL, PA428LL, PA429LL	2G iPod nano: 4 GB blue
MA448LL, PA448LL, PA449LL	5G iPod: 80 GB white
MA450LL, PA450LL, PA451LL	5G iPod: 80 GB black
MA452LL, MA664LL	5G iPod: 30 GB black (U2 special edition)
MA477LL, PA477LL, PA478LL	2G iPod nano: 2 GB silver
MA487LL, PA487LL, PA488LL	2G iPod nano: 4 GB green
MA489LL, PA489LL, PA490LL	2G iPod nano: 4 GB pink
MA497LL, PA497LL, PA498LL	2G iPod nano: 8 GB black
MA725LL, PA725LL	2G iPod nano: 4 GB red
MA899LL, PA899LL	2G iPod nano: 8 GB red
MA501LL	iPhone 4 GB
MA712LL	iPhone 8 GB
MB029LL, PB029LL, PB031LL	iPod classic: 80 GB silver

iPod model ID strings	iPod hardware
MB145LL, PB145LL, PB146LL	iPod classic: 160 GB silver
MB147LL, PB147LL, PB148LL	iPod classic: 80 GB black
MB150LL, PB150LL, PB151LL	iPod classic: 160 GB black
MA978LL, PA978LL, PA979LL, MB245LL	3G iPod nano: 4 GB silver
MA980LL, PA980LL, PA981LL, MB247LL	3G iPod nano: 8 GB silver
MB249LL, PB249LL, PB250LL, MB251LL	3G iPod nano: 8 GB blue
MB253LL, PB253LL, PB254LL, MB255LL	3G iPod nano: 8 GB green
MB257LL, PB257LL, PB258LL, MB259LL	3G iPod nano: 8 GB red
MB261LL, PB261LL, PB262LL, MB263LL	3G iPod nano: 8 GB black
MA623LL, PA623LL, PA624LL, PA839LL	iPod touch: 8 GB
MA627LL, PA627LL, PA628LL	iPod touch: 16 GB

Command 0x0F: RequestLingoProtocolVersion

Direction: Device → iPod

Retrieves version information for any of the lingoes supported by the iPod. The iPod responds with a "Command 0x10: ReturnLingoProtocolVersion" (page 84) containing the major and minor version information of the requested iPod lingo. This command has one parameter, the lingo whose version information is requested. The iPod returns an `ACK` command with a bad parameter status if an accessory calls this command with an invalid or unsupported lingo ID. The device should use the `RequestLingoProtocolVersion` command to determine what iAP features are available for each lingo used.

Table 5-28 `RequestLingoProtocolVersion` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x0F	Command: `RequestLingoProtocolVersion`
5	0x*NN*	The lingo for which to request version information.
6	0x*NN*	Checksum

Command 0x10: ReturnLingoProtocolVersion

Direction: iPod → Device

The iPod sends this command in response to the "Command 0x0F: RequestLingoProtocolVersion" (page 83) message from the device. The major and minor version information for the requested lingo are returned.

Table 5-29 `ReturnLingoProtocolVersion` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x05	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x10	Command: `ReturnLingoProtocolVersion`
5	0x*NN*	The lingo for which version information is being returned.
6	0x*NN*	The major protocol version for the given lingo.
7	0x*NN*	The minor protocol version for the given lingo.
8	0x*NN*	Checksum

Command 0x13: IdentifyDeviceLingoes

Direction: Device → iPod

The device sends this command to signal its presence and to identify its supported lingoes. In response, the iPod sends an `ACK` command. The `IdentifyDeviceLingoes` command is used by multi-lingo devices to report all supported lingoes and should be used in place of the `Identify` (0x01) command. The `IdentifyDeviceLingoes` command resets all device information set by a previous `Identify` command, including the authentication retry counter and any previously granted authentication access permissions. The payload of this command includes three fields: the Device Lingoes Spoken, Options, and Device ID fields.

> **Note:** The `IdentifyDeviceLingoes` command disables all but free lingoes on the current port unless authentication is requested. Immediate authentication is also required for level V2. For serial ports, this means that lingoes 0x00-0x03 and 0x05 may be used, not including unauthenticated commands; the USB port will be able to use only the General lingo, 0x00.

Table 5-30 `IdentifyDeviceLingoes` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)

Byte number	Value	Comment
1	0x55	Start of packet (SOP)
2	0x0E	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x13	Command: `IdentifyDeviceLingoes`
5-8	0x*NN*	Device Lingoes Spoken; see Table 5-31 (page 85) .
9-12	0x*NN*	Options; see Table 5-32 (page 86) .
13-16	0x*NN*	Device ID. Devices must send a unique identifier, supplied by the iPod Authentication Coprocessor, if they require authentication. If a device does not require authentication, it can send the Device ID to 0x00000000 and set the authentication option bits to 0x0.
17	0x*NN*	Checksum

Note: For information about the iPod Authentication Coprocessor, contact iPodDev@apple.com.

Use the Device Lingoes Spoken field as a bit field to set each bit corresponding to the lingoes supported by the accessory. For example, if an accessory device supports both the Microphone and Simple Remote lingoes, the bit field is 0x00000007 or the low byte in binary is 00000111.

Table 5-31 Device Lingoes Spoken bits

Bit	Supported lingo
0	General lingo (must be set by all devices)
1	Microphone lingo
2	Simple Remote lingo
3	Display Remote lingo
4	Extended Interface lingo
5	RF Transmitter lingo
6	USB Host Control lingo
7	RF Tuner lingo
8	Accessory Equalizer lingo
9	Reserved; set to 0
10	Digital Audio lingo
11	Reserved; set to 0

Bit	Supported lingo
12	Storage lingo
31:13	Reserved; set to 0

The bits of the Options field are defined as shown in Table 5-32 (page 86) .

Table 5-32 `IdentifyDeviceLingoes` Options bits

Bits	Meaning
1:0	Authentication control bits. These bits have the following meanings: 00 = no authentication is supported or required 01 = defer authentication until an authenticated command is used (authentication level V1 only); see Note below 10 = authenticate immediately after identification (required for authentication level V2). 11 = reserved
3:2	Power control bits. These bits have the following meanings: 00 = low power only 01 = intermittent high power 10 = reserved 11 = reserved
31:4	Reserved; set to 0

Note: Certain lingoes require immediate authentication. Requesting deferred authentication with the Microphone, USB Host Control, RF Tuner, Accessory Equalizer and Digital Audio lingoes results in a command failed `ACK` return from the iPod.

Devices identifying using the `IdentifyDeviceLingoes` command receive notifications when the iPod changes state. See "Command 0x23: NotifyiPodStateChange" (page 95) for more details.

Note: If a device uses an invalid device ID during an identification attempt, the iPod returns a bad parameter error (0x04) ACK.

Command 0x14: GetDevAuthenticationInfo

Direction: iPod → Device

The iPod sends this command to obtain authentication information from the device. The command is sent only if the device has indicated that it supports authentication in its `IdentifyDeviceLingoes` options bits and has passed a valid, nonzero device ID. In response, the device sends "Command 0x15: RetDevAuthenticationInfo" (page 87) .

Table 5-33 `GetDevAuthenticationInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x14	Command: `GetDevAuthenticationInfo`
5	0xEA	Checksum

Command 0x15: RetDevAuthenticationInfo

Direction: Device → iPod

The device indicates the iAP authentication version that it supports by returning this command in response to a "Command 0x14: GetDevAuthenticationInfo" (page 86) command from the iPod. The authentication version returned by this command must be consistent with the device ID sent in the `IdentifyDeviceLingoes` command.

The returned packet may have either of two formats, depending on whether the authentication process is level V1 or level V2. See Table 5-34 (page 87) and Table 5-35 (page 87) .

Table 5-34 `RetDevAuthenticationInfo` packet, authentication level V1

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x15	Command: `RetDevAuthenticationInfo`
5	0x*NN*	Authentication protocol major version number
6	0x*NN*	Authentication protocol minor version number
7	0x*NN*	Checksum

Table 5-35 `RetDevAuthenticationInfo` packet, authentication level V2

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)

Byte number	Value	Comment
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x15	Command: `RetDevAuthenticationInfo`
5	0x02	Authentication major version (0x02)
6	0x00	Authentication minor version (0x00)
7	0x*NN*	X.509 certificate current section index (0 = first section, 1 = second section, and so on)
8	0x*NN*	X.509 certificate maximum section index (0 = one section total, 1 = two sections, and so on)
9	0x*NN*...	X.509 certificate data. If the data length exceeds 500 bytes it must be broken into sections, each not more than 500 bytes.
(last byte)	0x*NN*	Checksum

Command 0x16: AckDevAuthenticationInfo

Direction: iPod → Device

The iPod sends this command in response to "Command 0x15: RetDevAuthenticationInfo" (page 87). It indicates the current state of the device authentication information. If the device receives a nonzero status, the iPod does not support the device's authentication version and the device will fail authentication.

Table 5-36 `AckDevAuthenticationInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x16	Command: `AckDevAuthenticationInfo`
5	0x*NN*	Status of authentication information. Possible values are: 0x00 = Authentication information supported 0x08 = Authentication information unsupported 0x0A = Certificate is invalid 0x0B = Certificate permissions are invalid

Byte number	Value	Comment
6	0x*NN*	Checksum

Command 0x17: GetDevAuthenticationSignature

Direction: iPod → Device

The iPod sends this command to authenticate a device that has identified itself as requiring authentication. Authentication occurs either immediately upon identification or when the device attempts to use a restricted lingo or command. The device calculates its digital signature based on the challenge offered by the iPod and sends the results back to the iPod using "Command 0x18: RetDevAuthenticationSignature" (page 90) .

The authentication retry counter is used to track the number of retries. If the returned signature cannot be verified, the iPod responds with a nonzero "Command 0x19: AckDevAuthenticationStatus" (page 90), followed immediately by another "Command 0x17: GetDevAuthenticationSignature" (page 89) .

The retry counter is set to 0x01 for the first authentication attempt and incremented each time the iPod retries the `GetDevAuthenticationSignature` command. Devices using level V1 authentication are allowed up to four retries and a maximum of 30 seconds (up to 7.5 seconds per retry) for the authentication process. Devices using level V2 authentication are allowed up to two retries and a maximum of 150 seconds (up to 75 seconds per retry) for the authentication process. If a device fails to respond to the `GetDevAuthenticationSignature` command, the authentication process will fail and the device will not be able to use any of the authenticated commands.

Table 5-37 `GetDevAuthenticationSignature` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x13 or 0x17	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x17	Command: `GetDevAuthenticationSignature`
5–20 or 5–24	0x*NN*…	An offered challenge sent for the device to sign and return (16 bytes for level V1 authentication, 20 bytes for level V2).
NN	0x*NN*	Authentication retry counter. The authentication process terminates after the maximum retry count or maximum timeout interval has been reached, whichever comes first. When the authentication process fails, only nonauthenticated lingoes and commands are usable.
(last byte)	0x*NN*	Checksum

Command 0x18: RetDevAuthenticationSignature

Direction: Device → iPod

The device sends this command to the iPod in response to "Command 0x17: GetDevAuthenticationSignature" (page 89) . The iPod verifies the digital signature, calculated by the device based on the offered challenge. If verification passes, the iPod authenticates the device and updates its lingo and command access permissions accordingly. The authentication status is sent to the device using "Command 0x19: AckDevAuthenticationStatus" (page 90) .

Table 5-38 `RetDevAuthenticationSignature` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x18	Command: `RetDevAuthenticationSignature`
5	0x*NN*…	The digital signature calculated by the device, based on the offered challenge (variable length).
(last byte)	0x*NN*	Checksum

Command 0x19: AckDevAuthenticationStatus

Direction: iPod → Device

The iPod sends this command to the device in response to the "Command 0x18: RetDevAuthenticationSignature" (page 90) command. It indicates the current device authentication state. If the device receives a nonzero status, the device has failed authentication and will only be able to use unauthenticated lingo commands.

If the device receives a zero status, the iPod has successfully authenticated the device. The device may then use the requested authenticated lingoes and commands. Optionally, the device may begin the process of authenticating the iPod, by sending "Command 0x1A: GetiPodAuthenticationInfo" (page 91) .

Table 5-39 `AckDevAuthenticationStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x00	Lingo ID: General lingo

Byte number	Value	Comment
4	0x19	Command: `AckDevAuthenticationStatus`
5	0x*NN*	Status of the authentication operation: 0x00 = Authentication operation passed !0x00 = Authentication operation failed
6	0x*NN*	Checksum

Command 0x1A: GetiPodAuthenticationInfo

Direction: Device → iPod

The device sends this command to obtain authentication information from the iPod. The device should send this command only if the device has indicated that it supports authentication in its "Command 0x13: IdentifyDeviceLingoes" (page 84) options bits and the iPod has successfully authenticated the device. (Device authentication is successful when the device receives the "Command 0x19: AckDevAuthenticationStatus" (page 90) command with a status of 0x00). In response to `GetiPodAuthenticationInfo` the iPod sends "Command 0x1B: RetiPodAuthenticationInfo" (page 91) .

Table 5-40 `GetiPodAuthenticationInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x1A	Command: `GetiPodAuthenticationInfo`
5	0xE4	Checksum

Command 0x1B: RetiPodAuthenticationInfo

Direction: iPod → Device

The iPod returns this command in response to "Command 0x1A: GetiPodAuthenticationInfo" (page 91) from the device.

Table 5-41 `RetiPodAuthenticationInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)

Byte number	Value	Comment
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x1B	Command: `RetiPodAuthenticationInfo`
5	0x02	Authentication major version (0x02)
6	0x00	Authentication minor version (0x00)
7	0x*NN*...	X.509 certificate current section index (0 = first section, 1 = second section, and so on)
8	0x*NN*	X.509 certificate maximum section index (0 = one section total, 1 = two sections, and so on)
9	0x*NN*	X.509 certificate data. If the data length exceeds 500 bytes it must be broken into sections, each not more than 500 bytes.
(last byte)	0x*NN*	Checksum

Note: Authentication version 2.00 (major version 0x02, minor version 0x00) is the only version supported by iPods that can be authenticated by devices.

Command 0x1C: AckiPodAuthenticationInfo

Direction: Device → iPod

The device sends this command to the iPod in response to "Command 0x1B: RetiPodAuthenticationInfo" (page 91) . It indicates the current state of the iPod authentication information version. If the device sends a nonzero status, it indicates that it will not be able to authenticate the iPod due to an invalid X.509 certificate.

Table 5-42 `AckiPodAuthenticationInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x1C	Command: `AckiPodAuthenticationInfo`

Byte number	Value	Comment
5	0x*NN*	Status of the authentication information: 0x00 = authentication information valid !0x00 = authentication information not valid
6	0x*NN*	Checksum

Command 0x1D: GetiPodAuthenticationSignature

Direction: Device → iPod

The device uses this command to send an offered challenge to the iPod for digital signature. In response, the iPod returns its signed challenge to the device using "Command 0x1E: RetiPodAuthenticationSignature" (page 93) . Accessories should implement the authentication retry feature described in "Command 0x17: GetDevAuthenticationSignature" (page 89), allowing the iPod two retries in 150 seconds. The retry counter should be set to 0x01 in the first `GetiPodAuthenticationSignature` command sent to the iPod and should be incremented for each subsequent attempt. Authentication should fail after either the retry count or maximum response interval have been exceeded since the first `GetiPodAuthenticationSignature` command was sent.

Table 5-43 `GetiPodAuthenticationSignature` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x1D	Command: `GetiPodAuthenticationSignature`
5–24	0x*NN*...	An offered challenge for the iPod to sign and return (20 bytes)
25	0x00	Authentication retry counter. A maximum of two retries or 150 seconds, whichever happens first, should occur before the authentication process is terminated.
26	0x*NN*	Checksum

Command 0x1E: RetiPodAuthenticationSignature

Direction: iPod → Device

The iPod sends this command to the device in response to "Command 0x1D: GetiPodAuthenticationSignature" (page 93) . The device verifies the digital signature, calculated by the iPod based on the offered challenge, and, if verification passes, authenticates the iPod. The device sends the authentication status to the iPod using "Command 0x1F: AckiPodAuthenticationStatus" (page 94) .

Table 5-44 `RetiPodAuthenticationSignature` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x1E	Command: `RetiPodAuthenticationSignature`
5 … *N*	0x*NN*...	The digital signature calculated by the iPod
(last byte)	0x*NN*	Checksum

Command 0x1F: AckiPodAuthenticationStatus

Direction: Device → iPod

The device sends this command to the iPod in response to "Command 0x1E: RetiPodAuthenticationSignature" (page 93) . It indicates the current iPod authentication state. The device should return a nonzero ACK for each failed authentication attempt.

Table 5-45 `AckiPodAuthenticationStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x1F	Command: `AckiPodAuthenticationStatus`
5	0x*NN*	Status of authentication information: 0x00 = authentication operation passed !0x00 = authentication operation failed
6	0x*NN*	Checksum

Command 0x23: NotifyiPodStateChange

Direction: iPod → Device

The iPod sends this notification command when the iPod state is about to change to devices that identify using "Command 0x13: IdentifyDeviceLingoes" (page 84) . If the device identifies using "Command 0x01: Identify" (page 69), this notification is not sent. The state change byte indicates the specific iPod state transition. If the iPod is switching from a Power On state to a Light Sleep state, devices must immediately reduce their power consumption below the 5 mA maximum current. When the iPod has transitioned to a Deep Sleep or Hibernate state, self-powered accessories are expected to automatically reidentify themselves 80 ms after accessory power is restored.

Note: Firmware version 1.0.0 of the iPod nano reports the iPod state change incorrectly. The `StateChg` enumeration is decremented by one, so that 0x00 represents a transition to Deep Sleep, 0x01 represents a transition to Hibernate, and so forth. All future versions of the nano firmware will conform to the specification below.

Table 5-46 `NotifyiPodStateChange` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x23	Command: `NotifyiPodStateChange`
5	0x*NN*	`StateChg` (1 byte). The iPod state change. Possible values are: 0x00 = no state change 0x01 = accessory power going to Deep Sleep state (no power) 0x02 = accessory power going to Hibernate state (no power) 0x03 = accessory power going to Light Sleep state (less than 5 mA current) 0x04 = accessory power going to the Power On state 0x05–0xFF = reserved
6	0x*NN*	Checksum

Command 0x24: GetiPodOptions

Direction: Device → iPod

The accessory device sends this command to ask the iPod to return a 64-bit field that defines the options that the iPod supports. In response, the iPod sends a `RetiPodOptions` command.

Table 5-47 `GetiPodOptions` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x00	Lingo ID: General lingo
4	0x24	Command ID: `GetiPodOptions`
5	0xDA	Checksum

Command 0x25: RetiPodOptions

Direction: iPod → Device

The iPod sends this command in response to a `GetiPodOptions` command from the accessory.

Table 5-48 `RetiPodOptions` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x0A	Length of packet
3	0x00	Lingo ID: General lingo
4	0x25	Command ID: `RetiPodOptions`
5	0x*NN*	Option bits (bits 63:56): Reserved
6	0x*NN*	Option bits (bits 55:48): Reserved
7	0x*NN*	Option bits (bits 47:40): Reserved
8	0x*NN*	Option bits (bits 39:32): Reserved
9	0x*NN*	Option bits (bits 31:24): Reserved
10	0x*NN*	Option bits (bits 23:16): Reserved
11	0x*NN*	Option bits (bits 15:8): Reserved
12	0x*NN*	Option bits (bits 7:0): Bit 1: iPod supports line-out usage Bit 0: iPod supports video output

Byte number	Value	Comment
13	0x*NN*	Checksum

Command 0x27: GetAccessoryInfo

Direction: iPod → Device

The iPod sends this command to devices that identify themselves using the `IdentifyDeviceLingoes` command to obtain certain information from the accessory. The accessory responds with the `RetAccessoryInfo` command. The iPod will use the information gathered to:

- Display accessory information in the Settings:About box on the iPod
- Display a message to the user if the iPod firmware needs to be updated to support the accessory
- Display a message to the user if the iPod firmware does not support the accessory

The `GetAccessoryInfo` command sends the Accessory Info Type and the Accessory Info Type parameters to the accessory. Table 5-50 (page 98) lists the number of parameter bytes for every corresponding Accessory Info Type. The iPod requests each of the Accessory Info Types in the order in which they appear in Table 5-50 (page 98) . Because the Accessory minimum supported iPod firmware version request (which sends the iPod model number as a parameter) is sent before any Accessory minimum supported lingo version requests, the accessory has the option of changing its responses to those appropriate for the iPod model.

When the `GetAccessoryInfo` command is sent with the accessory minimum supported iPod firmware version info type, the 4-byte iPod model number and the 3-byte iPod firmware version are sent as parameters. See "Command 0x0E: ReturniPodModelNum" (page 79) for the model number format and "Command 0x0A: ReturniPodSoftwareVersion" (page 77) for the firmware version format.

When the `GetAccessoryInfo` command is sent with the accessory minimum supported lingo version info type, the 1-byte lingo number for which the iPod is requesting the minimum supported version is sent as a parameter. The iPod will send the `GetAccessoryInfo` command with this Accessory Info Type for every lingo that the accessory indicates it supports.

The iPod begins sending `GetAccessoryInfo` commands as soon as an accessory identifies itself successfully via the `IdentifyDeviceLingoes` command. If the accessory does not respond, the iPod waits 5 seconds for a response before timing out and retrying. It retries a maximum of three times.

Table 5-49 `GetAccessoryInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x27	Command ID: `GetAccessoryInfo`

Byte number	Value	Comment
5	0x*NN*	Accessory Info Type (see Table 5-50 (page 98))
n...	0x*NN*...	Accessory Info Type parameters (see Table 5-50 (page 98) for length)
(last byte)	0x*NN*	Checksum

Table 5-50 Accessory Info Type values

Value	Meaning	Parameter bytes
0x00	Accessory info capabilities	0
0x01	Accessory name	0
0x02	Accessory minimum supported iPod firmware version	7
0x03	Accessory minimum supported lingo version	1
0x04	Accessory firmware version	0
0x05	Accessory hardware version	0
0x06	Accessory manufacturer	0
0x07	Accessory model number	0
0x08	Accessory serial number	0
0x09	Accessory incoming maximum payload size	0
0x0A-0xFF	Reserved	N/A

Table 5-51 `GetAccessoryInfo` packet with Accessory Info Type = 0x02

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x0A	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x27	Command ID: `GetAccessoryInfo`
5	0x02	Accessory Info Type (see Table 5-50 (page 98))
6	0x*NN*	iPod Model ID (bits 31:24)
7	0x*NN*	iPod Model ID (bits 23:16)

Byte number	Value	Comment
8	0x*NN*	iPod Model ID (bits 15:8)
9	0x*NN*	iPod Model ID (bits 7:0)
10	0x*NN*	iPod firmware major version number
11	0x*NN*	iPod firmware minor version number
12	0x*NN*	iPod firmware revision version number
13	0x*NN*	Checksum

Table 5-52 `GetAccessoryInfo` packet with Accessory Info Type = 0x03

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x04	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x27	Command ID: `GetAccessoryInfo`
5	0x03	Accessory Info Type (see Table 5-50 (page 98))
6	0x*NN*	Lingo Number
7	0x*NN*	Checksum

Command 0x28: RetAccessoryInfo

Direction: Device → iPod

The accessory device sends this command to the iPod in response to command 0x27, `GetAccessoryInfo`. The data contained in the packet depends on the Accessory Info Type requested by the iPod.

When the `RetAccessoryInfo` command is returning the accessory info capabilities, a bit-field is returned where every set bit represents a supported Accessory Info Type. See Table 5-49 (page 97) for a description of the Accessory Info Type; see Table 5-50 (page 98) for the meanings of its bytes.

Table 5-53 `RetAccessoryInfo` packet with Accessory Info Type = 0x00

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet

Byte number	Value	Comment
2	0x07	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x28	Command ID: `RetAccessoryInfo`
5	0x00	Accessory Info Type. See Table 5-50 (page 98)
6–9	0x*NNNNNNNN*	Accessory capabilities (32-bit field sent in big endian format; see Table 5-54 (page 100))
10	0x*NN*	Checksum

Table 5-54 Accessory Capabilities bit field

Bit number	Capability supported
0	Accessory info capabilities (must be set to 1)
1	Accessory name
2	Accessory minimum supported iPod firmware version
3	Accessory minimum supported lingo version
4	Accessory firmware version
5	Accessory hardware version
6	Accessory manufacturer
7	Accessory model number
8	Accessory serial number
9	Accessory incoming max packet size
10–31	Reserved

The accessory name, manufacturer, model number, and serial number are sent as UTF-8 character arrays and must be less than or equal to 64 bytes (including a null termination character). See Table 5-54 (page 100) for details. Note that even if this condition is met, the iPod may not be capable of displaying all the characters in the array in its About box.

Table 5-55 `RetAccessoryInfo` packet with Accessory Info Type = 0x01/0x06/0x07/0x08

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet

Byte number	Value	Comment
2	0x*NN*	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x28	Command ID: `RetAccessoryInfo`
5	0x*NN*	Accessory Info Type. See Table 5-50 (page 98)
6...	0x*NN*...	Null-terminated UTF-8 character array
(last byte)	0x*NN*	Checksum

When the accessory returns the 3-byte minimum supported iPod firmware major/minor/revision version it requires, it also returns the 4-byte iPod model ID. The accessory should therefore store all the model numbers of iPods that support it, along with their corresponding minimum supported iPod firmware versions.

If an unknown or unsupported iPod model ID is sent to the accessory, the accessory should return the `RetAccessoryInfo` command with the iPod Model ID and 0xFF as the iPod firmware major/minor/revision version numbers.

Table 5-56 `RetAccessoryInfo` packet with Accessory Info Type = 0x02

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x0A	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x28	Command ID: `RetAccessoryInfo`
5	0x02	Accessory Info Type. See Table 5-50 (page 98)
6	0x*NN*	iPod Model ID (bits 31:24)
7	0x*NN*	iPod Model ID (bits 23:16)
8	0x*NN*	iPod Model ID (bits 15:8)
9	0x*NN*	iPod Model ID (bits 7:0)
10	0x*NN*	minimum supported iPod firmware major version
11	0x*NN*	minimum supported iPod firmware minor version
12	0x*NN*	minimum supported iPod firmware revision version
13	0x*NN*	Checksum

If the accessory's minimum supported iPod firmware version is higher than the iPod firmware version, and one or more of the lingo version numbers is higher than that supported by the iPod, the iPod will display a message to the user indicating that the iPod firmware should be updated.

If the accessory's minimum supported iPod firmware version is smaller than or equal to the iPod firmware version or any of the major/minor/revision numbers are 0xFF, and one or more of the lingo version numbers is higher than that supported by the iPod, the iPod will display a message to the user indicating that the iPod does not support the accessory.

Table 5-57 `RetAccessoryInfo` packet with Accessory Info Type = 0x03

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x06	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x28	Command ID: `RetAccessoryInfo`
5	0x03	Accessory Info Type. See Table 5-50 (page 98)
6	0x*NN*	Lingo ID
7	0x*NN*	Major protocol version for lingo ID
8	0x*NN*	Minor protocol version for lingo ID
9	0x*NN*	Checksum

Table 5-58 `RetAccessoryInfo` packet with Accessory Info Type = 0x04/0x05

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x06	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x28	Command ID: `RetAccessoryInfo`
5	0x*NN*	Accessory Info Type. See Table 5-50 (page 98)
6	0x*NN*	Accessory major version number
7	0x*NN*	Accessory minor version number
8	0x*NN*	Accessory revision version number

Byte number	Value	Comment
9	0x*NN*	Checksum

The accessory's incoming maximum payload size indicates the maximum size of a packet from the iPod that the accessory can support. If the accessory does not return a value, the iPod assumes that the maximum payload size is 1024 bytes. The maximum payload size must be bigger or equal to 128 bytes and smaller or equal to 65536 bytes.

Only iPod packets that contain a UTF-8 character array can be larger than maximum payload size. In that case, the iPod will insert a null termination character into the UTF-8 array to force it to fit into the packet. This does not apply to commands that allow multipacket responses.

Note: The maximum payload size restriction does not apply to General lingo packets related to authentication.

Table 5-59 `RetAccessoryInfo` packet with Accessory Info Type = 0x09

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x06	Length of packet payload
3	0x00	Lingo ID: General lingo
4	0x28	Command ID: `RetAccessoryInfo`
5	0x09	Accessory Info Type. See Table 5-50 (page 98)
6	0x*NN*	Max payload size (bits 15:8)
7	0x*NN*	Max payload size (bits 7:0)
8	0x*NN*	Checksum

Command 0x29: GetiPodPreferences

Direction: Device → iPod

The accessory device sends this command to ask the iPod to return a specific class of preferences set on it, as defined by a preference class ID. In response, the iPod sends a `RetiPodPreferences` command.

Table 5-60 `GetiPodPreferences` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x03	Length of packet
3	0x00	Lingo ID: General lingo
4	0x29	Command ID: `GetiPodPreferences`
5	0x*NN*	Preference class ID (see Table 5-61 (page 104))
6	0x*NN*	Checksum

Table 5-61 iPod preference class and setting IDs

Class ID	Preference	Setting	Setting ID
0x00	Video out setting	Off	0x00
		On	0x01
		Ask	0x02
		Reserved	0x03–0xFF
0x01	Screen configuration	Fullscreen	0x00
		Widescreen	0x01
		Reserved	0x02–0xFF
0x02	Video format setting	NTSC	0x00
		PAL	0x01
		Reserved	0x02-0xFF
0x03	Line-out usage	Not used	0x00
		Used	0x01
		Reserved	0x02-0xFF
0x04-0x07	Reserved		
0x08	Video-out connection	None	0x00
		Composite	0x01
		S-video	0x02
		Component	0x03
		Reserved	0x04-0xFF

Class ID	Preference	Setting	Setting ID
0x09	Closed captioning	Off	0x00
		On	0x01
		Reserved	0x02-0xFF
0x0A-0xFF	Reserved		

Note: Class IDs 0x00-0x02 and 0x08-0x09, shown in Table 5-61 (page 104) , are available only if the iPod supports video playback. Class ID 0x03 is available only if the iPod supports line-out usage.

To receive video signals from most iPods, an accessory must be authenticated. Only authenticated accessories may get or set class IDs 0x00, 0x01, 0x02, and 0x08. The iPod photo and 5G iPod are exempt from this requirement.

Command 0x2A: RetiPodPreferences

Direction: iPod → Device

The iPod sends this command in response to a `GetiPodPreferences` command from the accessory. In byte 5 it echoes the requested preference class ID, and in byte 6 it sends the current preference setting for that class.

Table 5-62 `RetiPodPreferences` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet
3	0x00	Lingo ID: General lingo
4	0x2A	Command ID: `RetiPodPreferences`
5	0x*NN*	Preference class ID (see Table 5-61 (page 104))
6	0x*NN*	Preference setting ID (see Table 5-61 (page 104))
7	0x*NN*	Checksum

Command 0x2B: SetiPodPreferences

Direction: Device → iPod

The accessory device sends this command to the iPod to set a specific preference. It sends the preference class ID in byte 5 and the setting ID in byte 6. If byte 7, restore on exit, is set to 0x01, then the iPod restores the original setting for this preference when the accessory is disconnected; if it is 0x00, it does not perform the restore. Other values of byte 7 are illegal.

Table 5-63 `SetiPodPreferences` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x05	Length of packet
3	0x00	Lingo ID: General lingo
4	0x2B	Command ID: `SetiPodPreferences`
5	0x*NN*	Preference class ID (see Table 5-61 (page 104))
6	0x*NN*	Preference setting ID (see Table 5-61 (page 104))
7	0x*NN*	Restore on exit
8	0x*NN*	Checksum

Note: The iPod line-out state is always restored, regardless of the Restore-on-Exit setting. On some iPods, the video-out state is also always restored, regardless of the Restore-on-Exit setting.

Accessory Lingoes

The iPod Accessory Protocol defines a number of different accessory lingoes. This chapter describes these lingoes and their commands. Table 6-1 (page 107) shows the available accessory lingoes and their IDs.

Table 6-1 iPod accessory lingoes

Lingo	ID	Notes
Microphone	0x01	
Simple Remote	0x02	
Display Remote	0x03	
Extended Interface	0x04	See *iPod Extended Interface Specification*
RF Transmitter	0x05	
USB Host Control	0x06	
RF Tuner	0x07	
Accessory Equalizer	0x08	
Reserved	0x09	
Digital Audio	0x0A	
Reserved	0x0B	
Storage	0x0C	
Reserved	0x0D–0xFF	

Lingo 0x01: Microphone Lingo

The Microphone lingo enables combination microphone and speaker accessory devices to record and playback audio. Initial microphone devices supported one input mode (mono) and one sample rate (8 kHz). The increased iPod mass storage disk capacities enable the option of supporting a stereo input mode and higher audio sample rates. With these changes, iPods may be used for high-quality mobile audio recording.

Note: The iPod mini and the iPod nano do not support the Microphone lingo.

Older iPods, such as the 3G, mini, 4G, and photo, do not support line input on the 30-pin connector. Older microphones designed to connect to the 9-pin Audio/Remote connector will continue to work on these iPods but will be unusable on newer iPods that lack the 9-pin connector. Newer microphones plug into the 30-pin connector of the iPod and provide audio line-in signal levels. They use the General lingo (0x0) "Command 0x13: IdentifyDeviceLingoes" (page 84) and authenticate to get access to the microphone commands. The legacy `Identify` command cannot be used on the 30-pin connector to identify as a microphone device; all microphone lingo 0x01 commands on the 30-pin connector require authentication.

Legacy Microphone lingo commands 0x00–0x03 are disabled for devices using the 30-pin connector. They are superseded by "Command 0x06: iPodModeChange" (page 114), which returns an ACK.

The Microphone lingo is defined such that the iPod initiates commands and the accessory device responds to these commands; that is, the iPod sends commands to the device and the device responds with data or `ACK` commands.

Note: Unless otherwise specified, all data units larger than bytes must be transferred in big-endian order; that is, 32 bits must be sent as bits 31:24, followed by bits 23:16, and so forth. Similarly, 16 bits must be sent as bits 15:8, followed by bits 7:0.

When the iPod detects a device speaking the Microphone lingo, it may transition into a recorder application where it can create and manage recordings. Based on the microphone device capabilities, the iPod recording application may choose to change its appearance based on the presence or absence of certain microphone features. The device should indicate its capabilities to the iPod on request. These capabilities may include:

- Stereo line input source
- Stereo/mono control
- Recording level control
- Recording level limiter

Microphone accessory devices can draw power from the iPod or supply power to the iPod. Accessory device power management is important as iPods transition to a smaller physical size at the same time as trying to extend battery life. As a device using the General lingo `IdentifyDeviceLingoes` command, the microphone device will be notified of iPod state changes, such as transitioning to the Power On, Light Sleep, Hibernate, and Deep Sleep states. As with the previous Microphone lingo, accessory

power is in low mode by default and is raised to high power mode only during recording and playback states, if the device selected the intermittent high power option in the "Command 0x13: IdentifyDeviceLingoes" (page 84) command.

The microphone device is responsible for keeping its power consumption below the maximum allowed limits for each iPod state. Note that accessory power is completely shut off when the iPod enters the Hibernate and Deep Sleep states. When waking from a Light Sleep state, the microphone device is required to reidentify and reauthenticate itself, as with other devices using the `IdentifyDeviceLingoes` command and using authenticated lingoes or commands. On reset or power up, the accessory device should be in low power state (consuming less than 5 mA) with the amplifier off (that is, with audio input and output disabled).

The minimum time between an iPod notification of entering a power-off mode (Hibernate or Deep Sleep), and entry into that state is 100 ms. The minimum time between an iPod notification of ending recording or playback mode and the associated accessory power reduction is also 100 ms.

Microphone state information should be retained locally by the device while uninterrupted accessory power (either high or low power) is available. If accessory power is turned off, device state information may be lost. Devices are not expected to retain state information across accessory power down cycles (Hibernate or Deep Sleep modes).

iPod playback volume level changes may require the device to support Display Remote lingo (0x03) functionality.

Table 6-2 (page 109) lists the commands available as part of the Microphone lingo.

Table 6-2 Microphone lingo command summary

Command	ID	Data length	Protocol Version	Connector	Authentication Required
`BeginRecord`	0x00	0x00	All	9-pin Audio/Remote	No
`EndRecord`	0x01	0x00	All	9-pin Audio/Remote	No
`BeginPlayback`	0x02	0x00	All	9-pin Audio/Remote	No
`EndPlayback`	0x03	0x00	All	9-pin Audio/Remote	No
`ACK`	0x04	0x02	1.01	30-pin	Yes
`GetDevAck`	0x05	0x00	1.01	30-pin	Yes
`iPodModeChange`	0x06	0x01	1.01	30-pin	Yes
`GetDevCaps`	0x07	0x00	1.01	30-pin	Yes
`RetDevCaps`	0x08	0x04	1.01	30-pin	Yes
`GetDevCtrl`	0x09	0x00	1.01	30-pin	Yes
`RetDevCtrl`	0x0A	0x02	1.01	30-pin	Yes
`SetDevCtrl`	0x0B	0x02	1.01	30-pin	Yes
Reserved	0x0C–0xFF	N/A	N/A	N/A	N/A

Command History of the Microphone Lingo

Table 6-3 (page 110) shows the history of command changes in the Microphone lingo:

Table 6-3 Microphone lingo command history

Lingo version	Command changes	Features
No version	Add: 0x00–0x03	Microphone begin/end record/playback notification commands, 9-pin Audio/Remote connector only
1.00	None	Version number available through `RequestLingoProtocol-Version`
1.01	Add: 0x04–0x0B	ACK, mode change, capabilities, and control support (30-pin connector only)

9-Pin Audio/Remote Connector Commands

The commands in this section apply only to the 9-pin Audio/Remote connector. For the top connector microphone only, the iPod sends a `BeginRecord` command when recording is about to begin. The device microphone bias, if applicable, should already be present. iPod sends an `EndRecord` command when recording is completed. The device may remove microphone bias, if applicable, after the `EndRecord` command is received.

The iPod sends a `BeginPlayback` command when playback is about to begin. The device may then turn on its speaker amplifier, if present. Upon receipt of an `EndPlayback` command, the device must turn off its speaker amplifier. For all Microphone lingo commands sent by the iPod, no device response is expected.

Command 0x00: BeginRecord

Direction: iPod → Device

The iPod sends this command to notify the device that audio recording has started. The device does not return a packet to the iPod in response to this command. See "Command 0x06: iPodModeChange" (page 114) for more details.

Table 6-4 `BeginRecord` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x00	Command ID: `BeginRecord`

Byte number	Value	Comment
5	0xFD	Checksum

Command 0x01: EndRecord

Direction: iPod → Device

The iPod sends this command to notify the device that audio recording has ended. The device does not return a packet to the iPod in response to this command. See "Command 0x06: iPodModeChange" (page 114) for more details.

Table 6-5 `EndRecord` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x01	Command ID: `EndRecord`
5	0xFC	Checksum

Command 0x02: BeginPlayback

Direction: iPod → Device

The iPod sends this command to notify the device that audio playback has started. The device does not return a packet to the iPod in response to this command. See "Command 0x06: iPodModeChange" (page 114) for more details.

Table 6-6 `BeginPlayback` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x02	Command ID: `BeginPlayback`
5	0xFB	Checksum

Command 0x03: EndPlayback

Direction: iPod → Device

The iPod sends this command to notify the device that audio playback has ended. The device does not return a packet to the iPod in response to this command. See "Command 0x06: iPodModeChange" (page 114) for more details.

Table 6-7 `EndPlayback` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x03	Command ID: `EndPlayback`
5	0xFA	Checksum

30-pin Connector Commands

The commands described in this section apply only to the 30-pin connector.

Note: Devices must return responses to iPod commands in the order in which the commands were received and within the specified time limits. Failing to send responses in the order commands were received or exceeding the command timeout limits could cause a communications failure and result in the device being considered not present by the iPod.

Command 0x04: ACK

Direction: Device → iPod

The microphone device sends this command in response to a command sent from the iPod. Note that the commands 0x00–0x03 do not require an `ACK` response. The device sends an `ACK` response when a command that does not return any data has completed, a bad parameter is received, or an unsupported or invalid command is received.

Table 6-8 `ACK` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload

Byte number	Value	Comment
3	0x01	Lingo ID: Microphone lingo
4	0x04	Command ID: ACK
5	0x*NN*	The command result status. See Table 6-9 (page 113) for the possible values.
6	0x*NN*	The ID of the command for which the response is being sent.
7	0x*NN*	Checksum

Table 6-9 (page 113) shows the possible values of the command result status byte.

Table 6-9 Command result values

Byte	Meaning
0x00	Success (OK)
0x01	Not applicable: the Microphone lingo does not use this error value.
0x02	ERROR: Command failed. Sent in response to a valid command if that command did not succeed.
0x03	ERROR: Out of resources. This indicates that an iPod internal allocation failed.
0x04	ERROR: Bad parameter. The command or input parameters are invalid.
0x05	Not applicable: the Microphone lingo does not use this error value.
0x06	Reserved
0x07	ERROR: The device is not authenticated to use this lingo command.
0x08	ERROR: Mismatched authentication protocol version.
0x09 - 0xFF	Reserved

Command 0x05: GetDevAck

Direction: iPod → Device

The iPod sends this command to get an ACK response from a microphone device. The iPod uses this command to "ping" the device and determine that it is present and ready to accept commands. In response, the device sends the ACK command with command status OK.

The timeout for this command is 200 ms (0.2 second). The device must respond within the allotted time; the iPod will not retry the command. If the device does not respond within the specified time, the command will fail and the device may be considered not present by the iPod.

Table 6-10 `GetDevAck` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x05	Command ID: `GetDevAck`
5	0xF8	Checksum

Command 0x06: iPodModeChange

Direction: iPod → Device

The iPod sends this command to the microphone device when an audio recording or playback event occurs. The microphone device uses the `iPodModeChange` command to configure its inputs or outputs and power consumption level for the specified mode. In response, the device sends the `ACK` command with the command status OK. The device sends the `ACK` command when the device has completed its mode change.

The iPod does not wait to receive an `ACK` command from the device in response to the mode change. It may continue sending other commands to the device after it has sent the mode change command.

Table 6-11 `iPodModeChange` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x06	Command ID: `iPodModeChange`
5	0x*NN*	Mode. See Table 6-12 (page 115).
6	0x*NN*	Checksum

Table 6-12 (page 115) lists the possible values of the Mode byte.

Table 6-12 Mode values

Value	Meaning
0x00	Begin audio recording mode. When it receives this command, the device can activate its microphone recording inputs or outputs connected to the iPod line inputs. If the device has requested the intermittent high power option using General lingo "Command 0x13: IdentifyDeviceLingoes" (page 84), it must wait until after this command is received before consuming more that 5 mA of accessory supply current. By the time the device receives this command, accessory high power is enabled and ready to use during the recording process.
0x01	End audio recording mode. When it receives this command, the device can deactivate its microphone recording inputs or outputs and return to a quiescent state. If the device has requested the intermittent high power option, by using the General lingo command `IdentifyDeviceLingoes`, it may be in a high power consumption state. After receiving this command, the device must reduce its power consumption below 5 mA of accessory supply current within 100 ms.
0x02	Begin audio playback mode. When it receives this command, the device can activate its speaker playback inputs or outputs connected to the iPod line outputs, if present. If the device has requested the intermittent high power option, by using the General lingo command `IdentifyDeviceLingoes`, it must wait until after this command is received before consuming more that 5 mA of accessory supply current. By the time the device receives this command, accessory high power is enabled and ready to use during the playback process.
0x03	End audio playback mode. When it receives this command, the device can deactivate its speaker playback inputs or outputs and return to a quiescent state. If the device has requested the intermittent high power option using the General lingo command `IdentifyDeviceLingoes`, it may be in a high power consumption state. After receiving this command, the device must reduce its power consumption below 5 mA of accessory supply current within 100 ms.
0x04–0xFF	Reserved.

Note: Failure to wait for a mode change notification before increasing power consumption could result in an incompatibility with present and future iPods.

Failure to reduce power consumption within the stated time could result in an incompatibility with present and future iPods.

Command 0x07: GetDevCaps

Direction: iPod → Device

The iPod sends this command to the microphone device to determine the features present on the device. In response, the device sends "Command 0x08: RetDevCaps" (page 116) with the payload indicating the capabilities it supports.

The timeout for this command is 200 ms (0.2 second). The device must respond within the allotted time; the iPod will not retry the command. If the device does not respond within the specified time, the command will fail and the device may be considered not present by the iPod.

Table 6-13 `GetDevCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x07	Command ID: `GetDevCaps`
5	0xF6	Checksum

Command 0x08: RetDevCaps

Direction: Device → iPod

The device sends this command in response to the command "Command 0x07: GetDevCaps" (page 115) sent by the iPod. The microphone device returns the payload indicating which capabilities it supports.

Table 6-14 `RetDevCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x08	Command ID: `RetDevCaps`
5	0x*NN*	Device capabilities (bits 31: 24). See Table 6-15 (page 117).
6	0x*NN*	Device capabilities (bits 23: 16)
7	0x*NN*	Device capabilities (bits 15: 8)
8	0x*NN*	Device capabilities (bits 7: 0)
9	0x*NN*	Checksum

The capabilities bit ranges correspond to the microphone control commands. The iPod should not attempt to control device features, using "Command 0x0B: SetDevCtrl" (page 119), if the associated capabilities bits are not set. Table 6-15 (page 117) lists the meaning of these bits.

Table 6-15 Microphone capabilities bitmask

Bit	Meaning
00	Stereo line input. A value of 0 indicates the device is monophonic only.
01	Stereo or mono line input. This bit should be set only if the microphone supports stereo line input and can switch between stereo and mono modes.
02	Recording level is present and variable.
03	Recording level limit is present.
31:04	Reserved.

Command 0x09: GetDevCtrl

Direction: iPod → Device

The iPod sends this command to get the device control state for the specified control type. In response, the device sends "Command 0x0A: RetDevCtrl" (page 118) with its current control state. If this command is not supported by the device—that is, if the microphone does not have any configurable controls—it should return an `ACK` command with a bad parameter error status.

The timeout for this command is 200 ms (0.2 second). The device must respond within the allotted time; the iPod will not retry the command. If the device does not respond within the specified time, the command will fail and the device may be considered not present by the iPod.

Table 6-16 `GetDevCtrl` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x09	Command ID: `GetDevCtrl`
5	0x*NN*	The control type for which to get the state. The possible values are: 0x00: Reserved. 0x01: Stereo/mono line input. 0x02: Recording level control. 0x03: Recording level limiter control. 0x04–0xFF: Reserved.

Byte number	Value	Comment
6	0x*NN*	Checksum

Command 0x0A: RetDevCtrl

Direction: Device → iPod

The device sends this command in response to the command "Command 0x09: GetDevCtrl" (page 117) received from the iPod. The device returns the current control state for the specified control type. Control types are supported only if the associated capabilities bits are set in the command "Command 0x08: RetDevCaps" (page 116).

Table 6-17 `RetDevCtrl` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x0A	Command ID: `RetDevCtrl`
5	0x*NN*	The control type. See Table 6-18 (page 118).
6	0x*NN*	The control data. See Table 6-18 (page 118).
7	0x*NN*	Checksum

Table 6-18 (page 118) lists the different control types and the data associated with them.

Table 6-18 Control types and data

Control type value	Control type	Control data
0x00	Reserved	Stereo capability cannot be set.
0x01	Stereo/mono line input	Possible data values are: 0x00 = mono 0x01 = stereo 0x02–0xFF = reserved
0x02	Recording level control	Possible data values are in a range between: 0x00 = mute 0xFF = maximum gain

Control type value	Control type	Control data
0x03	Recording level limiter	Possible data values are: 0x00 = off 0x01 = on 0x02–0xFF = reserved
0x04–0xFF	Reserved	

Command 0x0B: SetDevCtrl

Direction: iPod → Device

The iPod sends this command to set the device control state for the specified control type. In response, the device sends the `ACK` command with the command status. If this command is not supported by the device—that is, if the microphone does not have any configurable controls—it should return an `ACK` command with a bad parameter error status.

The timeout for this command is 200 ms (0.2 second). The device must respond within the allotted time; the iPod will not retry the command. If the device does not respond within the specified time, the command will fail and the device may be considered not present by the iPod.

Table 6-19 `SetDevCtrl` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload
3	0x01	Lingo ID: Microphone lingo
4	0x0B	Command ID: `SetDevCtrl`
5	0x*NN*	The control type. The control type and data values are the same as for the `RetDevCtrl` (0x0A) command. See Table 6-18 (page 118) for more information.
6	0x*NN*	The control data. See Table 6-18 (page 118) for more information.
7	0x*NN*	Checksum

Lingo 0x02: Simple Remote lingo

This lingo is intended for a remote device that retains no state information about the iPod. Simple commands are sent to the iPod, and no acknowledgment or state information is sent back to the device. This lingo is used by the iPod standard in-line remote control.

History and Applicability

System software versions 2.0 through 2.2 on the 3G iPod and versions 1.0 and 1.1 of the iPod mini support only the first five button responses (0 through 4) in the `ContextButtonStatus` command. This is called the contextual button lingo and includes only command 0x00. An extended set of commands (0x01 through 0x04) is available in the dedicated media lingoes, as shown in Table 6-20 (page 120).

Table 6-20 Simple remote lingo command summary

Command	ID	Direction	Packet	Lingo version	Authentication
0x00	`ContextButtonStatus`	Device to iPod	`buttonStatus:4`	All	No
0x01	`ACK`	iPod to Device	`cmdStatus:1, cmdID:1`	1.01	Yes
0x02	`ImageButtonStatus`	Device to iPod	`buttonStatus:4`	1.01	Yes
0x03	`VideoButtonStatus`	Device to iPod	`buttonStatus:4`	1.01	Yes
0x04	`AudioButtonStatus`	Device to iPod	`buttonStatus:4`	1.01	Yes
0x05–0xFF	Reserved	N/A	N/A	N/A	N/A

Dedicated media lingoes are supported by version 1.01 of the Simple Remote lingo. By querying the version number, an accessory can determine the level of command support, as shown in Table 6-21 (page 120).

Table 6-21 Simple remote lingo support versions

Version	Support
No version	Command 0 supported, buttons 0–4
No version	Command 0 supported, buttons 0–25
1.00	Version number can be obtained
1.01	Command 0–4 supported, buttons 0–25, media controls
1.02	Bug fix for compatibility with some accessories: when a device identifies itself using the General lingo `Identify` command, the 200 ms time limit during which `ContextButtonStatus` must report that all buttons are up is removed.

Table 6-22 (page 120) shows the history of command changes in the Simple Remote lingo:

Table 6-22 Simple Remote lingo command history

Lingo version	Command changes	Features
No version	Add: 0x00	Context specific button status, buttons 0–4

Lingo version	Command changes	Features
No version	None	Context-specific button status, buttons 5-25 added (iPod 4G with firmware versions 3.0.0 and 3.0.1)
1.00	0x0F	Version number available through `RequestLingoProtocol-Version`
1.01	Add: 0x01–0x04	Image-, video-, and audio-specific media control button status
1.02	None	All-buttons-up timeout applied to all commands, except not to `ContextButtonStatus` when that command is used with the General lingo (0x00) `Identify` command (0x01)

Using the Contextual Button lingo

A simple remote device sends a `ContextButtonStatus` command to provide updated status on which buttons are held down or released. The data of the packet is a number of bytes indicating which buttons are currently held down. The bytes are constructed by ORing the masks of the buttons together. To indicate all buttons are released, the device must send a full data payload consisting of 0x00. While any buttons are held down, the device should periodically send an updated `ContextButtonStatus` packet on a 30 ms to 100 ms interval.

It is not necessary to transmit any trailing bytes in which no bits are set. If this option is exercised, the length of the packet in the header should be adjusted accordingly; that is, the packet payload length should be decreased to exclude the trailing zero byte(s) that will not be transmitted.

When the user presses and holds down a button, a simple remote device should generate the button status packet immediately and repeat it every 30 to 100 ms for as long as the button is pressed. If a second button is pressed while the first button is down, the button status packet sent by the device should include status for both buttons, and this packet should be repeated every 30 to 100 ms for as long as both buttons are held down. Table 6-24 (page 123) lists the possible iPod button states.

Some iPod button states are interpreted differently by the iPod when pressed and held down for 2 seconds or more. These are as follows:

- The Next Track button is treated as a Scan Forward button when pressed and held while a track is playing.
- The Previous Track button is treated as a Scan Backward button when pressed and held while a track is playing.
- The Play/Pause button is treated as a Power Off button when pressed and held.
- In iPods before the iPod photo, the Menu button acted as a Display Backlight On/Off button when pressed and held. Starting with the iPod photo, pressing and holding the Menu button causes a jump to the top level menu.
- If the iPod is in Browse mode, the Select button is treated as an Add Track to On-The-Go Playlist button when pressed and held.

Repeated Next Track and Previous Track commands (see Table 6-24 (page 123)), without an intervening button status packet indicating all buttons are up, are interpreted as Fast Forward and Rewind commands. For a locking Fast Forward or Rewind button, use the Begin Fast Forward or Begin Rewind commands to start the operation and a Play/Resume command to return to the play state.

The Next and Previous Album commands (see Table 6-24 (page 123)) have no effect if there is no next or previous album to go to in the Now Playing list.

Use the following steps to wake up an iPod when a simple remote button is pressed (UART serial port only):

1. Send a 0xFF sync byte.
2. Wait 20 ms.
3. Send the button status packet.
4. Wait 30 to 100 ms.
5. Repeat steps 3 and 4 for as long as any button is pressed.

Multiple button status packets cannot be sent back to back; otherwise, the repeated button status packets may be misinterpreted as being part of a corrupted packet. Repeated packets must always be separated by a gap of more than 25 ms, so the iPod knows that the new packet is not part of the previous packet (which may happen if the SYNC and SOP bytes in the first packet have been lost).

Using the Dedicated Media lingoes

The iAP contextual button protocol sends command packets with button messages that are interpreted based on the iPod user interface context. When an iPod is playing media types such as images and video, however, remote controls can become overloaded and their behavior may become confusing to the iPod user. In this case, the accessory device should use dedicated media control button commands.

The dedicated media lingoes include an `ACK` command, so devices know that a packet has been received, and dedicated button status commands for each media type currently supported by the iPod: images, slideshows, videos, and audio. Use of the dedicated media lingoes requires accessory device authentication.

Media control button status bits are organized so that the most frequently used buttons are assigned low bit positions. This reduces the button status packet sizes for frequently used buttons. Button status is maintained separately for all ports and all commands. This means that buttons can be in different states for different media control types.

Note: For a given port and media control type (except `ContextButtonStatus` from a device using the `Identify` command), if a command has not been received within approximately 200 ms after the last button status command, the button status will be reset to all buttons up.

Command 0x00: ContextButtonStatus

Direction: Device → iPod

The device sends this command to the iPod when a button event occurs. The button status is a bitmask representing each button that is currently pressed. The device should send the button status packet repeatedly at intervals between 30 and 100 ms, while one or more buttons are pressed. When all buttons are released, the device should send a button status packet with a 0x0 payload to indicate that no buttons are pressed. The iPod does not return a packet to the device in response to this command.

Table 6-23 `ContextButtonStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03–0x06	Length of packet payload
3	0x02	Lingo ID: Simple Remote lingo
4	0x00	Command ID: `ContextButtonStatus`
5…	0x*NN*…	1 to 4 `ButtonStatus` bytes. See Table 6-24 (page 123) for a list of the button states recognized by the iPod.
(last byte)	0x*NN*	Checksum

Table 6-24 (page 123) lists the available buttons and their bitmasks.

Table 6-24 Button states

Button name	Number	Byte index	Button bitmask
Play/Pause	0	0x0	0x01
Volume Up	1	0x0	0x02
Volume Down	2	0x0	0x04
Next Track	3	0x0	0x08
Previous Track	4	0x0	0x10
Next Album	5	0x0	0x20

Button name	Number	Byte index	Button bitmask
Previous Album	6	0x0	0x40
Stop	7	0x0	0x80
Play/Resume	8	0x1	0x01
Pause	9	0x1	0x02
Mute toggle	10	0x1	0x04
Next Chapter	11	0x1	0x08
Previous Chapter	12	0x1	0x10
Next Playlist	13	0x1	0x20
Previous Playlist	14	0x1	0x40
Shuffle Setting Advance	15	0x1	0x80
Repeat Setting Advance	16	0x2	0x01
Power On	17	0x2	0x02
Power Off	18	0x2	0x04
Backlight for 30 Seconds	19	0x2	0x08
Begin Fast Forward	20	0x2	0x10
Begin Rewind	21	0x2	0x20
Menu	22	0x2	0x40
Select	23	0x2	0x80
Up Arrow	24	0x3	0x01
Down Arrow	25	0x3	0x02
Backlight off	26	0x3	0x04
Reserved	27-31	0x3	0xF8

The “Backlight off” button does not have any effect in the iPod touch.

Command 0x01: ACK

Direction: iPod → Device

The iPod sends this command in response to any command sent from the device, except command 0x00. An ACK response is sent when a command that does not return any data has completed, when a bad parameter is received, or when an unsupported or invalid command is received.

Table 6-25 ACK packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload
3	0x02	Lingo ID: Simple Remote lingo
4	0x01	Command ID: ACK
5	0x*NN*	Status of command received (see Table 6-26 (page 125))
6	0x*NN*	ID of the command being acknowledged
7	0x*NN*	Checksum

Table 6-26 Command status codes

Code	Command status
0x00	Command OK
0x01	Unknown track category (not applicable)
0x02	Command failed (valid command, did not succeed)
0x03	Out of resources (iPod internal allocation failed)
0x04	Bad parameter (command or input parameters invalid)
0x05	Unknown track ID (not applicable)
0x06	Command pending (cmdPendTime parameter returned)
0x07	Not authenticated (not authenticated)
0x08	Mismatched authentication protocol version
0x09–0xFF	Reserved

Command 0x02: ImageButtonStatus

Direction: Device → iPod

The device sends this command to the iPod when an image-specific button event occurs. The button status is a bitmask representing each button that is currently pressed. The button status packet should be repeatedly sent by the device at intervals between 30 and 100 ms while one or more buttons are pressed. When all buttons are released, the device should send a button status packet with a 0x00 payload to indicate that no buttons are pressed. In response, the iPod will return an `ACK` packet containing the command status to the device.

Table 6-27 `ImageButtonStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x02	Lingo ID: Simple Remote lingo
4	0x02	Command ID: `ImageButtonStatus`
5	0x*NN*...	Image-specific button status bitmask (see Table 6-28 (page 126))
(last byte)	0x*NN*	Checksum

Table 6-28 Image-specific button values

Bit	Meaning
00	Play/Pause
01	Next image
02	Previous image
03	Stop
04	Play/resume
05	Pause
06	Shuffle advance
07	Repeat advance
31:08	Reserved

Command 0x03: VideoButtonStatus

Direction: Device → iPod

The device sends this command to the iPod when a video-specific button event occurs. The button status is a bitmask representing each button that is currently pressed. The button status packet should be repeatedly sent by the device at intervals between 30 ms and 100 ms while one or more buttons are pressed. When all buttons are released, the device should send a button status packet with a 0x00 payload to indicate that no buttons are pressed. In response, the iPod will return an `ACK` packet containing the command status to the device.

Table 6-29 `VideoButtonStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x02	Lingo ID: Simple Remote lingo
4	0x03	Command ID: `VideoButtonStatus`
5	0x*NN*...	Video-specific button status bitmask (see Table 6-30 (page 127))
(last byte)	0x*NN*	Checksum

Table 6-30 Video-specific button values

Bit	Meaning
00	Play/Pause
01	Next video
02	Previous video
03	Stop
04	Play/Resume
05	Pause
06	Begin FF
07	Begin REW
08	Next chapter
09	Previous chapter
10	Next frame
11	Previous frame
12	Caption advance

Bit	Meaning
31:13	Reserved

Command 0x04: AudioButtonStatus

Direction: Device → iPod

The device sends this command to the iPod when an audio-specific button event occurs. The button status is a bitmask representing each button that is currently pressed. The button status packet should be repeatedly sent by the device at intervals between 30 ms and 100 ms while one or more buttons are pressed. When all buttons are released, the device should send a button status packet with a 0x00 payload to indicate that no buttons are pressed. In response, the iPod will return to the device an `ACK` message containing the command status.

Table 6-31 `AudioButtonStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x02	Lingo ID: Simple Remote lingo
4	0x04	Command ID: `AudioButtonStatus`
5	0x*NN*...	Audio-specific button status bitmask (see Table 6-32 (page 128))
(last byte)	0x*NN*	Checksum

Table 6-32 Audio-specific button values

Bit	Meaning
00	Play/Pause
01	Volume up
02	Volume down
03	Next track
04	Previous track
05	Next album
06	Previous album
07	Stop

Bit	Meaning
08	Play/Resume
09	Pause
10	Mute toggle
11	Next chapter
12	Previous chapter
13	Next playlist
14	Previous playlist
15	Shuffle setting advance
16	Repeat setting advance
17	Begin FF
18	Begin Rew
19	Record
31:20	Reserved

Lingo 0x03: Display Remote Lingo

The Display Remote lingo is for accessory devices that need to control the state of the iPod, display information about the state of the iPod on a remote display, or control the state of the iPod equalizer. The Display Remote protocol can be used by simple inline-display remotes (remotes that have single-line display and play control buttons) and more complex devices that have full multiline graphical displays to show information about the track, artist, or album; current play or pause state; track position; battery; shuffle and time.

For example, the Display Remote protocol can be used in an automotive application to show currently playing track information on the in-vehicle display while allowing the user to browse the database stored in the iPod. Such an application could let the user choose where to browse the database (on the in-vehicle display or on the iPod) and switch between RemoteUI and Display Remote modes, depending on the setting.

By supporting multiple lingoes, an accessory can use the Display Remote lingo in combination with other lingoes to create a fully functional iPod/accessory system. Accessories can also use this lingo to control the state of the iPod equalizer. The Display Remote lingo supports serial accessories attached to the 9-pin Audio/Remote connector or to the 30-pin connector.

The Display Remote command set uses a single byte command format similar to the General and Simple Remote lingoes. Devices using the Display Remote lingo can identify using the General lingo (0x0), with either the `Identify` (0x00) single lingo or `IdentifyDeviceLingoes` (0x13) multiple lingo commands. See "Command 0x01: Identify" (page 69) and "Command 0x13: IdentifyDeviceLingoes" (page 84) for more information.

Note: The Display Remote lingo is an authenticated lingo, with some exceptions. USB accessories must authenticate before they can use the Display Remote lingo. Serial accessories have access only to the equalizer control, battery state, and sound check state commands (commands 0x01–0x07 and 0x1A–0x1E) if they do not authenticate. Please refer to Table 6-33 (page 130) to determine which commands require authentication.

The Display Remote lingo can operate in notification (interrupt) mode, where the iPod sends event notifications to the device, or in polled (non-interrupt) mode. In polled mode, the device should send requests for state change information to the iPod.

The Display Remote commands export text as UTF-8 characters. Graphics cannot be exported. Note that Chapter count information can be retrieved only from the currently playing track.

Table 6-33 Display Remote lingo command summary

Command	ID	Data length	Protocol version	Authentication required
`ACK`	0x00	0x02	1.00	No
`GetCurrentEQProfileIndex`	0x01	0x00	1.00	No
`RetCurrentEQProfileIndex`	0x02	0x04	1.00	No
`SetCurrentEQProfileIndex`	0x03	0x05	1.00	No
`GetNumEQProfiles`	0x04	0x00	1.00	No
`RetNumEQProfiles`	0x05	0x04	1.00	No
`GetIndexedEQProfileName`	0x06	0x04	1.00	No
`RetIndexedEQProfileName`	0x07	0x*NN*	1.00	No
`SetRemoteEventNotification`	0x08	0x04	1.02	Yes
`RemoteEventNotification`	0x09	0x*NN*	1.02	Yes
`GetRemoteEventStatus`	0x0A	0x00	1.02	Yes
`RetRemoteEventStatus`	0x0B	0x04	1.02	Yes
`GetiPodStateInfo`	0x0C	0x01	1.02	Yes
`RetiPodStateInfo`	0x0D	0x*NN*	1.02	Yes
`SetiPodStateInfo`	0x0E	0x*NN*	1.02	Yes

Command	ID	Data length	Protocol version	Authentication required
GetPlayStatus	0x0F	0x00	1.02	Yes
RetPlayStatus	0x10	0x0D	1.02	Yes
SetCurrentPlayingTrack	0x11	0x04	1.02	Yes
GetIndexedPlayingTrackInfo	0x12	0x07	1.02	Yes
RetIndexedPlayingTrackInfo	0x13	0x*NN*	1.02	Yes
GetNumPlayingTracks	0x14	0x00	1.02	Yes
RetNumPlayingTracks	0x15	0x04	1.02	Yes
GetArtworkFormats	0x16	0x02	1.04	Yes
RetArtworkFormats	0x17	0x*NN*	1.04	Yes
GetTrackArtworkData	0x18	0x0C	1.04	Yes
RetTrackArtworkData	0x19	0x*NN*	1.04	Yes
GetPowerBatteryState	0x1A	0x00	1.02	No
RetPowerBatteryState	0x1B	0x02	1.02	No
GetSoundCheckState	0x1C	0x00	1.02	No
RetSoundCheckState	0x1D	0x01	1.02	No
SetSoundCheckState	0x1E	0x02	1.02	No
GetTrackArtworkTimes	0x1F	0x0C	1.04	Yes
RetTrackArtworkTimes	0x20	0x*NN*	1.04	Yes
Reserved	0x21–0xFF	N/A	N/A	N/A

Command History of the Display Remote lingo

Table 6-34 (page 131) shows the history of command changes in the Display Remote lingo:

Table 6-34 Display Remote lingo command history

Lingo version	Command changes	Features
1.00	Add: 0x00–0x07	iPod Equalizer Setting save/restore control
1.01	None	BugFix: Equalizer state not restored on extended interface exit

Lingo version	Command changes	Features
1.02	Add: 0x08–0x15, 0x1A–0x1E	Event notifications, iPod state info, playback track info, sound check, power/battery support
1.03	None	BugFix: Fix intermittent UI hang when restoring on exit
1.04	Add: 0x16–0x19, 0x1F–0x20	Track artwork, lyrics, track time position in seconds
1.05	None	Video browsing support added to playback commands. Chapter information can be retrieved for all tracks in the Now Playing list, not just for the currently playing track.

Transferring Album Art

The Display Remote lingo includes several commands that support the transfer of album artwork from an iPod to an accessory:

- `GetArtworkFormats`
- `RetArtworkFormats`
- `GetTrackArtworkTimes`
- `RetTrackArtworkTimes`
- `GetTrackArtworkData`
- `RetTrackArtworkData`

All album art image encoding is RGB-565, which can be transferred in both big- and small-endian formats. All formats are fixed-size; scalable images are not supported.

Artwork retrieval takes place in the following steps:

1. Retrieve the number of formats available for artwork on the iPod using `GetArtworkFormats`. It is not necessary to call `GetArtworkFormats` more than once per session; these values will be static while the accessory is attached to the iPod. However, there are no guarantees about the number of formats and which ones are available on a particular model or firmware version. Each `formatID` in `RetArtworkFormats` specifies both a pixel encoding, such as RGB-565 little-endian, and the image dimensions.

2. When the accessory wants to retrieve the artwork for a given track, it calls `GetIndexedPlayingTrackInfo` with an `infoType` of 0x08. This returns the count of artwork available for each `formatID` associated with the track. It is possible that a track may not have artwork for a particular `formatID` or that the number of images will vary by `formatID`.

3. To retrieve the list of images associated with a given track and `formatID`, the accessory calls `GetTrackArtworkTimes`. This command tells the iPod to return the associated timestamp for each artwork. The timestamp indicates when the artwork should be displayed, expressed in milliseconds from the start of playback.

4. When the accessory wants to retrieve an individual piece of artwork, it sends `GetTrackArtworkData` to the iPod. This requires the accessory to specify a track, a formatID, and the timestamp of the desired image. The iPod returns the specified artwork and the accessory can display it whenever it chooses.

Command 0x00: ACK

Direction: iPod → Device

The iPod sends this command to acknowledge the receipt of a command from the device and return the command status. The command ID field indicates the device command for which the response is being sent. The command status indicates the result of the command (success or failure).

Table 6-35 `ACK` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x00	Command ID: `ACK`
5	0x*NN*	Command result status. See Table 6-36 (page 133).
6	0x*NN*	The ID for the command being acknowledged.
7	0x*NN*	Checksum

Table 6-36 (page 133) lists the possible result values for the command result status field.

Table 6-36 Command result values

Result	Meaning
0x00	Success (OK)
0x01	Reserved
0x02	ERROR: Command failed
0x03	ERROR: Out of resources
0x04	ERROR: Bad parameter
0x05	ERROR: Unknown ID
0x06	Reserved

Result	Meaning
0x07	ERROR: Accessory not authenticated
0x08–0xFF	Reserved

Command 0x01: GetCurrentEQProfileIndex

Direction: Device → iPod

The device requests the current Equalizer Profile setting index. In response, the iPod sends the "Command 0x02: RetCurrentEQProfileIndex" (page 134) packet.

Table 6-37 `GetCurrentEQProfileIndex` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x01	Command ID: `GetCurrentEQProfileIndex`
5	0xFA	Checksum

Command 0x02: RetCurrentEQProfileIndex

Direction: iPod → Device

The iPod sends this command, returning the current Equalizer Profile setting index, in response to the "Command 0x01: GetCurrentEQProfileIndex" (page 134) packet sent by the device. The profile index is returned in an unsigned 32-bit big-endian format (bits 31:24 followed by bits 23:16 and so forth). An Equalizer Index of 0x0 indicates that the equalizer is disabled.

Table 6-38 `RetCurrentEQProfileIndex` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x02	Command ID: `RetCurrentEQProfileIndex`

Byte number	Value	Comment
5	0x*NN*	Current Equalizer Index (bits 31:24)
6	0x*NN*	Current Equalizer Index (bits 23:16)
7	0x*NN*	Current Equalizer Index (bits 15:8)
8	0x*NN*	Current Equalizer Index (bits 7:0)
9	0x*NN*	Checksum

Command 0x03: SetCurrentEQProfileIndex

Direction: Device → iPod

The device sets the current Equalizer Profile setting index and optionally restores the original Equalizer Setting on accessory detach. The valid Equalizer Index range can be determined by sending "Command 0x04: GetNumEQProfiles" (page 136). The profile index should be sent in an unsigned 32-bit big-endian format (bits 31:24 followed by bits 23:16 and so forth). An Equalizer Index of 0x0 tells the iPod that the equalizer should be disabled; a valid nonzero index enables the equalizer.

In response to this command, the iPod returns an `ACK` packet with the status of this command.

Table 6-39 `SetCurrentEQProfileIndex` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x07	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x03	Command ID: `SetCurrentEQProfileIndex`
5	0x*NN*	Current Equalizer Index (bits 31:24)
6	0x*NN*	Current Equalizer Index (bits 23:16)
7	0x*NN*	Current Equalizer Index (bits 15:8)
8	0x*NN*	Current Equalizer Index (bits 7:0)
9	0x*NN*	`bRestoreOnExit`. Specifies whether to restore the previous Equalizer Setting on device exit or detach. See the discussion below.
10	0x*NN*	Checksum

The bRestoreOnExit Boolean byte flag determines the behavior of the iPod when the accessory device is detached from the connector. A value of 0x0 (false) indicates that the original Equalizer Setting should be discarded. A nonzero (true) value indicates that the previous Equalizer Setting should be restored when the device is detached from the iPod. Anytime the SetCurrentEQProfileIndex command is sent with bRestoreOnExit equal to false, the previous equalizer state is erased and lost. If SetCurrentEQProfileIndex is sent with bRestoreOnExit equal to true every time, the first SetCurrentEQProfileIndex command saves the original equalizer state; subsequent commands do not change the original saved equalizer state. On accessory detach, the original saved equalizer state is restored.

Command 0x04: GetNumEQProfiles

Direction: Device → iPod

The device requests the number of iPod Equalizer Profile settings. In response, the iPod sends the "Command 0x05: RetNumEQProfiles" (page 136) packet.

Table 6-40 GetNumEQProfiles packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x04	Command ID: GetNumEQProfiles
5	0xF7	Checksum

Command 0x05: RetNumEQProfiles

Direction: iPod → Device

The iPod returns the number of Equalizer Profiles in it. It sends this command in response to the "Command 0x04: GetNumEQProfiles" (page 136) packet sent by the device. The Equalizer Profile count is returned in unsigned 32-bit big-endian format (bits 31:24 followed by bits 23:16 and so forth). The valid profile index range for iPod equalizer commands accepting a profile index is 0x0 to profileCount-1.

Table 6-41 RetNumEQProfiles packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x06	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x05	Command ID: `RetNumEQProfiles`
5	0x*NN*	`profileCount`. The Equalizer Profile count (bits 31:24).
6	0x*NN*	Equalizer profile count (bits 23:16)
7	0x*NN*	Equalizer profile count (bits 15:8)
8	0x*NN*	Equalizer profile count (bits 7:0)
9	0x*NN*	Checksum

Command 0x06: GetIndexedEQProfileName

Direction: Device → iPod

The device requests the iPod Equalizer Profile setting name for a given Equalizer Profile index. In response, the iPod sends the "Command 0x07: RetIndexedEQProfileName" (page 138) packet. The profile index should be sent in an unsigned 32-bit big-endian format (bits 31:24 followed by bits 23:16 and so forth). The valid profile index range can be obtained by sending "Command 0x04: GetNumEQProfiles" (page 136).

Table 6-42 `GetIndexedEQProfileName` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x06	Command ID: `GetIndexedEQProfileName`
5	0x*NN*	Equalizer profile index (bits 31:24)
6	0x*NN*	Equalizer profile index (bits 23:16)
7	0x*NN*	Equalizer profile index (bits 15:8)
8	0x*NN*	Equalizer profile index (bits 7:0)
9	0x*NN*	Checksum

Command 0x07: RetIndexedEQProfileName

Direction: iPod → Device

The iPod returns its Equalizer Profile setting name for the specified Equalizer Profile index in response to "Command 0x06: GetIndexedEQProfileName" (page 137). The Equalizer Profile name is returned as a variable-length, null-terminated UTF-8 character array.

Note: The UTF-8 Equalizer Profile name string is not limited to 255 characters. It may be sent in either small or large packet format. The following table shows the small packet format.

Table 6-43 `RetIndexedEQProfileName` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x07	Command ID: `RetIndexedEQProfileName`
5 … *N*	0x*NN*…	The Equalizer Profile name, as a null-terminated UTF-8 character array.
(last byte)	0x*NN*	Checksum

Command 0x08: SetRemoteEventNotification

Direction: Device → iPod

The device requests that the iPod enable asynchronous remote event notification for specific iPod events. Notification for each event can be enabled by setting the associated bit in the remote event bitmask (`remEventMask`). By default, all event notifications are disabled and must be explicitly enabled using this command. In response, the iPod sends an `ACK` command indicating the command completion status. A remote event bitmask of 0x0 disables all remote event status notifications. On device detach, event notification is reset to the default disabled state.

Table 6-44 `SetRemoteEventNotification` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo

Byte number	Value	Comment
4	0x08	Command ID: `SetRemoteEventNotification`
5	0x*NN*	`remEventMask` (bits 31:24). See Table 6-45 (page 139) for a list of the events for which you can enable notification.
6	0x*NN*	`remEventMask` (bits 23:16)
7	0x*NN*	`remEventMask` (bits 15:8)
8	0x*NN*	`remEventMask` (bits 7:0)
9	0x*NN*	Checksum

Enable notifications for the events listed in Table 6-45 (page 139) by setting the bit for each event in the remote event bitmask. A value of 1 enables the notification of the iPod state change for that event and a value of 0 disables the notification.

Table 6-45 iPod events

Bit number	Remote Event
0	Track time position in milliseconds
1	Track playback index
2	Chapter index
3	Play status (play, pause, stop, FF, and RW)
4	Mute/volume
5	Power/battery
6	Equalizer setting
7	Shuffle setting
8	Repeat setting
9	Date and time setting
10	Alarm setting
11	Backlight state
12	Hold switch state
13	Sound check state
14	Audiobook speed
15	Track time position in seconds

Bit number	Remote Event
31:16	Reserved

Command 0x09: RemoteEventNotification

Direction: iPod → Device

The iPod sends this command asynchronously whenever an enabled event change has occurred. Use "Command 0x08: SetRemoteEventNotification" (page 138) to control which events are enabled. The notification packet formats are described in more detail below. Notifications for enabled events are sent every 500 ms, with the exception of volume change notifications, which are sent every 100 ms.

Note: Notifications are sent only when the iPod is awake; none are sent when the iPod is in Light Sleep, Deep Sleep, or Hibernate states.

This is the command packet for the `RemoteEventNotification` command. See Table 6-47 (page 141) to interpret the `eventNum` and `eventData` fields.

Table 6-46 `RemoteEventNotification` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x09	Command ID: `RemoteEventNotification`
5	0x*NN*	`eventNum`. A number indicating the type of event.
6 … *N*	0x*NN*	`eventData`. Additional information about the event. This field is variable length; see Table 6-47 (page 141).
(last byte)	0x*NN*	Checksum

Table 6-47 (page 141) shows the expected payload and length for each type of event notification. For example, if an accessory receives an event notification for event 0x02 (Chapter Info) then the payload is 8 bytes long and contains the track index (bytes 0–3), the chapter count (bytes 4 and 5) and the chapter index (bytes 6 and 7). If bytes 4 and 5 are all zeros, the currently playing track does not have chapters.

Table 6-47 Event notification data

Event number	Event	Event data	Data length in bytes
0x00	Track position	The new track position, in milliseconds. ■ Byte 0: Track Position (bits 31:24) ■ Byte 1: Track Position (bits 23:16) ■ Byte 2: Track Position (bits 15:8) ■ Byte 3: Track Position (bits 7:0)	0x04
0x01	Track index	The index of the currently playing track. ■ Byte 0: Track Index (bits 31:24) ■ Byte 1: Track Index (bits 23:16) ■ Byte 2: Track Index (bits 15:8) ■ Byte 3: Track Index (bits 7:0)	0x04
0x02	Chapter information	Chapter information, including the track index, chapter count, and chapter index. ■ Bytes 0–3 specify the currently playing track index, as follows: Byte 0: Track Index (bits 31:24) Byte 1: Track Index (bits 23:16) Byte 2: Track Index (bits 15:8) Byte 3: Track Index (bits 7:0) ■ Bytes 4-5 specify the chapter count of the track, as follows. A value of 0x0000 indicates that the track does not have chapters. Byte 4: Chapter Count (bits 15:8) Byte 5: Chapter Count (bits 7:0) ■ Bytes 6-7 specify the chapter index, as follows. A value of 0xFFFF indicates that the track does not have chapters. Byte 6: Chapter Index (bits 15:8) Byte 7: Chapter Index (bits 7:0)	0x08
0x03	Play status	The current play status of the iPod; that is, whether it is playing, paused, stopped, fast forwarding or rewinding. Possible values are described in Table 6-48 (page 144). ■ Byte 0: Play Status (bits 7:0)	0x01

Event number	Event	Event data	Data length in bytes
0x04	Mute/volume	The current state of the mute setting and volume information. ■ Byte 0: Mute State (bits 7:0) A value of 0 indicates that mute is off; a value of 1 indicates that mute is on. ■ Byte 1: Volume Level (bits 7:0) A value between 0 and 255, with 0 indicating minimum volume and 255 indicating maximum volume. Note that if the Mute State value is true (mute is on), the volume level field is not valid and is returned as 0.	0x02
0x05	Power/battery	Information about the power and battery status. ■ Byte 0: Power State (bits 7:0) A value indicating the current power source and its state. See Table 6-51 (page 145) for possible values. ■ Byte 1: Battery Level (bits 7:0) Specifies the current battery level. A value from 0 to 255, with 0 indicating a fully discharged battery and 255 indicating a battery that is fully charged. If an external power status is returned, the battery level is invalid and is returned as 0.	0x02
0x06	Equalizer state	The current Equalizer Setting index. ■ Byte 0: Equalizer index (bits 31:24) ■ Byte 1: Equalizer index (bits 23:16) ■ Byte 2: Equalizer index (bits 15:8) ■ Byte 3: Equalizer index (bits 7:0)	0x04
0x07	Shuffle	The state of the shuffle setting. See Table 6-49 (page 145) for a list of possible values. ■ Byte 0: Shuffle State (bits 7:0)	0x01
0x08	Repeat	The state of the repeat setting. See Table 6-50 (page 145) for a list of possible values. ■ Byte 0: Repeat State (bits 7:0)	0x01

Event number	Event	Event data	Data length in bytes
0x09	Date/time	The current date and time. ■ Bytes 0–1 specify the current year. A value of 2005 represents the year 2005 A.D. Byte 0: Year (bits 15:8) Byte 1: Year (bits 7:0) ■ Byte 2: Month (bits 7:0) A value between 1 and 12, where 1 = January and 12 = December. ■ Byte 3: Day of the month (bits 7:0) A value between 1 and 31. ■ Byte 4: Hour (bits 7:0) A value between 0 and 23, where 0 = 12:00 a.m. and 23 = 11:00 p.m. ■ Byte 5: Minute (bits 7:0) A value between 0 and 59.	0x06
0x0A	Alarm	Alarm information. ■ Byte 0: Alarm State (bits 7:0) A value indicating the current state of the alarm. Possible values are: 0 = off 1 = enabled 2 = triggered ■ Byte 1: Alarm Hour (bits 7:0) A value between 0 and 23, where 0 = 12:00 a.m. and 23 = 11:00 p.m. ■ Byte 2: Alarm Minute (bits 7:0) A value between 0 and 59.	0x03
0x0B	Backlight	The current backlight level. A value between 0 and 255, where 0 indicates the backlight is off and 255 indicates that the backlight is at full intensity. ■ Byte 0: Backlight Level (bits 7:0)	0x01

Event number	Event	Event data	Data length in bytes
0x0C	Hold switch	The current state of the hold switch. A value of 0 means the hold switch is off. A value of 1 indicates it is on. ■ Byte 0: Hold Switch State (bits 7:0)	0x01
0x0D	Sound check	The state of the sound check setting. A value of 0 means that sound check is off; a value of 1 indicates it is on. ■ Byte 0: Sound Check State (bits 7:0)	0x01
0x0E	Audiobook	The audiobook playback speed setting. See Table 6-52 (page 146) for a list of possible values. ■ Byte 0: Audiobook Playback Speed Setting (bits 7:0)	0x01
0x0F	Track position in seconds	The new track time position, in seconds. ■ Byte 0: Track Position (bits 15:8) ■ Byte 1: Track Position (bits 7:0)	0x02
0X10–0XFF	Reserved	N/A	N/A

The battery and volume levels are normalized across all iPod platforms so that 0 represents the minimum level and 255 represents the maximum level. The granularity of minimum to maximum range steps varies by iPod platform.

Table 6-48 (page 144) lists the possible values for the data associated with a Play Status event and their meanings.

Table 6-48 Play status values

Value	Meaning
0x00	Playback stopped
0x01	Playing (for "Command 0x0E: SetiPodStateInfo" (page 150), start or resume playback)
0x02	Playback paused
0x03	Fast forward (FF)
0x04	Fast rewind (REW)
0x05	End fast forward or rewind mode
0x06–0xFF	Reserved

Table 6-49 (page 145) lists the possible values for the data associated with a Shuffle event.

Table 6-49 Shuffle state

Value	Meaning
0x00	Shuffle off
0x01	Shuffle tracks and songs
0x02	Shuffle albums
0x03–0xFF	Reserved

Table 6-50 (page 145) lists the possible values for the data associated with a Repeat event.

Table 6-50 Repeat state

Value	Meaning
0x00	Repeat off
0x01	Repeat one track or song
0x02	Repeat all tracks
0x03–0xFF	Reserved

Table 6-51 (page 145) lists the possible values for the data associated with a Power/Battery event and their meanings.

Table 6-51 Power and battery state

Value	Meaning
0x00	Internal battery power, low power (< 30%)
0x01	Internal battery power
0x02	External power, battery pack, no charging
0x03	External power, no charging
0x04	External power, battery charging
0x05	External power, battery charged
0x06–0xFF	Reserved

Table 6-52 (page 146) lists the possible values for the data associated with an Audiobook event and their meanings.

Table 6-52 Audiobook playback speed

Value	Meaning
0xFD	Slowest (–3)
0xFE	Slower (–2)
0xFF	Slow (–1)
0x00	Normal
0x01	Fast (+1)
0x02	Faster (+2)
0x03	Fastest (+3)
0x04–0xFC	Reserved

Command 0x0A: GetRemoteEventStatus

Direction: Device → iPod

The device requests the status of state information that has changed on the iPod. In response, the iPod sends "Command 0x0B: RetRemoteEventStatus" (page 146), containing a bitmask of event states that changed since the last `GetRemoteEventStatus` command and clears all the remote event status bits. This command may be used to poll the iPod for event changes without enabling asynchronous remote event notification. For more information on asynchronous event notification, see "Command 0x08: SetRemoteEventNotification" (page 138).

Table 6-53 `GetRemoteEventStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0A	Command ID: `GetRemoteEventStatus`
5	0xF1	Checksum

Command 0x0B: RetRemoteEventStatus

Direction: iPod → Device

The iPod sends this command in response to "Command 0x0A: GetRemoteEventStatus" (page 146).

Table 6-54 `RetRemoteEventStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0B	Command ID: `RetRemoteEventStatus`
5	0x*NN*	`remEventStatus` (bits 31:24). The event status that has changed. See Table 6-45 (page 139) for a list of possible events.
6	0x*NN*	`remEventStatus` (bits 23:16)
7	0x*NN*	`remEventStatus` (bits 15:8)
8	0x*NN*	`remEventStatus` (bits 7:0)
9	0x*NN*	Checksum

The bits in Table 6-45 (page 139) represent the events whose status has changed on the iPod since the last `GetRemoteEventStatus` command was received. For example, if the returned `remEventStatus` field has bits 2 and 7 set, then the chapter index and shuffle states of the iPod have changed since the last `GetRemoteEventStatus` command was sent by the accessory. Accessories can use "Command 0x0C: GetiPodStateInfo" (page 147) to get the updated state information for those events.

Command 0x0C: GetiPodStateInfo

Direction: Device → iPod

The device obtains iPod state information. The information type (`infoType`) field specifies the type of information to get. In response, the iPod sends "Command 0x0D: RetiPodStateInfo" (page 149) with the requested state information.

Table 6-55 `GetiPodStateInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0C	Command ID: `GetiPodStateInfo`

Byte number	Value	Comment
5	0x*NN*	`infoType` (1 byte). The type of state information for which to query the iPod. See Table 6-56 (page 148).
6	0x*NN*	Checksum

Table 6-56 (page 148) lists the different types of information for which you can query the iPod.

Table 6-56 `infoType` values

Value	Type of information
0x00	Track time position in milliseconds
0x01	Track playback index
0x02	Chapter information
0x03	Play status (play, pause, stop, FF, and RW)
0x04	Mute and volume information
0x05	Power and battery status
0x06	Equalizer setting
0x07	Shuffle setting
0x08	Repeat setting
0x09	Date and time
0x0A	Alarm state and time
0x0B	Backlight state
0x0C	Hold switch state
0x0E	Audiobook speed
0x0F	Track time position in seconds
0x10–0xFF	Reserved

For example, to retrieve the iPod Equalizer Setting, an accessory could send the command shown in Table 6-57 (page 148).

Table 6-57 `GetiPodStateInfo` packet to retrieve the iPod Equalizer Setting

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)

Byte number	Value	Comment
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0C	Command ID: `GetiPodStateInfo`
5	0x06	`infoType` = 0x06 (Equalizer setting)
6	0xE8	Checksum

Command 0x0D: RetiPodStateInfo

Direction: iPod → Device

The iPod sends this command in response to "Command 0x0C: GetiPodStateInfo" (page 147). The format of the returned state information depends on the type of information. See Table 6-47 (page 141) for a description of the data returned for each information type.

Table 6-58 `RetiPodStateInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0D	Command ID: `RetiPodStateInfo`
5	0x*NN*	`infoType` (1 byte). The type of iPod state information returned.
6 … *N*	0x*NN*	`infoData` (variable length). The iPod state information.
(last byte)	0x*NN*	Checksum

For example, an accessory requesting the Chapter Info of the currently playing track would receive a `RetiPodStateInfo` command packet similar to this (assuming a track index of 10, a chapter count of 8 and a current chapter index of 3):

Table 6-59 `RetiPodStateInfo` packet for requesting chapter information

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x0B	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0D	Command ID: `RetiPodStateInfo`
5	0x02	`infoType` = 0x02 (Chapter Information)
6	0x00	`infoData` = Track Index (bits 31:24)
7	0x00	`infoData` = Track Index (bits 23:16)
8	0x00	`infoData` = Track Index (bits 15:8)
9	0x0A	`infoData` = Track Index (bits 7:0)
10	0x00	`infoData` = Chapter Count (bits 15:8)
11	0x08	`infoData` = Chapter Count (bits 7:0)
12	0x00	`infoData` = Chapter Index (bits 15:8)
13	0x03	`infoData` = Chapter Index (bits 7:0)
14	0xCE	Checksum

Command 0x0E: SetiPodStateInfo

Direction: Device → iPod

Sets the iPod state. The information type (`infoType`) field specifies the type of information to update. In response, the iPod sends an `ACK` command with the results of the operation. Some commands include a `bRestoreOnExit` parameter that optionally allows the original iPod setting to be restored on exit.

Table 6-60 `SetiPodStateInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0E	Command ID: `SetiPodStateInfo`
5	0x*NN*	`infoType` (1 byte). The type of iPod state information to set.
6 … *N*	0x*NN*	`infoData` (variable length). The data for the iPod state information to set.

Byte number	Value	Comment
(last byte)	0x*NN*	Checksum

Table 6-61 (page 151) lists the possible values of the `infoType` field and the corresponding data in the `infoData` field.

Table 6-61 iPod state data

Information type		Information data	
Value	**Name**	**Description**	**Length (bytes)**
0x00	Track position	The new position of the track, in milliseconds. ■ Byte 0: Track Position (bits 31:24) ■ Byte 1: Track Position (bits 23:16) ■ Byte 2: Track Position (bits 15:8) ■ Byte 3: Track Position (bits 7:0)	0x04
0x01	Track index	The index of the track to play. ■ Byte 0: Track Index (bits 31:24) ■ Byte 1: Track Index (bits 23:16) ■ Byte 2: Track Index (bits 15:8) ■ Byte 3: Track Index (bits 7:0)	0x04
0x02	Chapter index	The new chapter index. ■ Byte 0: Chapter Index (bits 15:8) ■ Byte 1: Chapter Index (bits 7:0)	0x02
0x03	Play status	The play status of the iPod (play, pause, stop, FF or REW). See Table 6-48 (page 144) for a list of possible values. Byte 0: Play Status (bits 7:0)	0x01

Information type		Information data	
Value	**Name**	**Description**	**Length (bytes)**
0x04	Mute/volume	The new mute setting or volume level. ■ Byte 0: Mute State (bits 7:0) A value of 0x00 turns mute off; a value of 0x01 turns on mute. ■ Byte 1: Volume Level (bits 7:0) A value between 0 and 255, with 0 indicating minimum volume and 255 indicating maximum volume. ■ Byte 2: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154). If the mute state is 0x01, the volume level field is ignored.	0x03
0x05	Power/battery	Reserved: Power and battery state cannot be set.	N/A
0x06	Equalizer state	The new Equalizer Setting. ■ Byte 0: Equalizer index (bits 31:24) ■ Byte 1: Equalizer index (bits 23:16) ■ Byte 2: Equalizer index (bits 15:8) ■ Byte 3: Equalizer index (bits 7:0) ■ Byte 4: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154).	0x05
0x07	Shuffle	The new state of the shuffle setting. ■ Byte 0: Shuffle State (bits 7:0) See Table 6-49 (page 145) for a list of possible values. ■ Byte 1: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154).	0x02
0x08	Repeat	The new state of the repeat setting ■ Byte 0: Repeat State (bits 7:0) See Table 6-50 (page 145) for a list of possible values. ■ Byte 1: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154).	0x02

Information type		Information data	
Value	**Name**	**Description**	**Length (bytes)**
0x09	Date/time	The new date and time. ■ Bytes 0–1 specify the current year. A value of 2005 represents the year 2005 A.D. Byte 0: Year (bits 15:8) Byte 1: Year (bits 7:0) ■ Byte 2: Month (bits 7:0) A value between 1 and 12, where 1 = January and 12 = December. ■ Byte 3: Day of the month (bits 7:0) A value between 1 and 31. ■ Byte 4: Hour (bits 7:0) A value between 0 and 23, where 0 = 12:00 a.m. and 23 = 11:00 p.m. ■ Byte 5: Minute (bits 7:0) A value between 0 and 59.	0x06
0x0A	Alarm	The alarm state and time. ■ Byte 0: Alarm State (bits 7:0) A value of 0 turns the alarm off, while a value of 1 enables the alarm. ■ Byte 1: Alarm Hour (bits 7:0) A value between 0 and 23, where 0 = 12:00 a.m. and 23 = 11:00 p.m. ■ Byte 2: Alarm Minute (bits 7:0) A value between 0 and 59. ■ Byte 3: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154).	0x04

Information type		Information data	
Value	**Name**	**Description**	**Length (bytes)**
0x0B	Backlight	The new state of the backlight. ■ Byte 0: Backlight Level (bits 7:0) The backlight control only supports two modes: on or off. A value of 0 turns the backlight off; any other value turns on the backlight. There is no support for dimming the iPod backlight setting. ■ Byte 1: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154).	0x02
0x0C	Hold switch	Reserved. The state of the Hold switch cannot be set.	N/A
0x0D	Sound check	The new sound check state. ■ Byte 0: Sound Check State (bits 7:0) A value of 0x00 turns sound check off; a value of 0x01 turns it on. ■ Byte 1: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154).	0x02
0x0E	Audiobook speed	The audiobook playback speed. ■ Byte 0: Audiobook Playback Speed Setting (bits 7:0) See Table 6-52 (page 146) for a list of possible values. ■ Byte 1: `bRestoreOnExit` (bits 7:0) See Table 6-62 (page 154).	0x01
0x0F	Track position in seconds	The current track time position, in seconds. ■ Byte 0: Track Time Position (bits 15:8) ■ Byte 1: Track Time Position (bits 7:0)	0x02
0X10–0XFF	Reserved	N/A	N/A

Table 6-62 (page 154) lists the possible values for the `bRestoreOnExit` parameter.

Table 6-62 Restore-on-exit values

Value	Meaning
0x00	Do not save the original state.

Value	Meaning
0x01	Save the original state and restore it on exit.

For example, an accessory could send the command shown in Table 6-63 (page 155) to set the iPod alarm to 8:15 a.m. and have the setting restored upon accessory detach.

Table 6-63 `SetiPodStateInfo` packet to set the alarm

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x07	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0E	Command ID: `SetiPodStateInfo`
5	0x0A	`infoType` = 0x0A (Alarm)
6	0x01	`infoData` = Alarm State set to On (0x01).
7	0x08	`infoData` = Alarm Hour set to 8.
8	0x0F	`infoData` = Alarm Minute set to 15.
9	0x01	`InfoData` = `bRestoreOnExit` set to true (0x01).
10	0xC5	Checksum

As another example, an accessory could send the command shown in Table 6-64 (page 155) to set the currently playing track on the iPod to track 1000.

Table 6-64 `SetiPodStateInfo` packet for setting the current track

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x07	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0E	Command ID: `SetiPodStateInfo`
5	0x01	`infoType` = 0x01 (Track Index)
6	0x00	`infoData` = Track Index (bits 31:24). Sets the playback track index to 1000.

Byte number	Value	Comment
7	0x00	`infoData` = Track Index (bits 23:16)
8	0x03	`infoData` = Track Index (bits 15:8)
9	0xE8	`infoData` = Track Index (bits 7:0)
10	0xFC	Checksum

Note: `SetiPodStateInfo` commands that use information types that have data fields larger than one byte must be sure to send the data in big-endian format. See the example in Table 6-64 (page 155).

Command 0x0F: GetPlayStatus

Direction: Device → iPod

The device requests the current iPod play status information. In response, the iPod sends "Command 0x10: RetPlayStatus" (page 156) with the current play state, track index, track position, and track length.

Table 6-65 `GetPlayStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x0F	Command ID: `GetPlayStatus`
5	0xEC	Checksum

Command 0x10: RetPlayStatus

Direction: iPod → Device

The iPod sends this command in response to "Command 0x0F: GetPlayStatus" (page 156) and returns the current iPod play status information. If the iPod is in a playing or paused state, the track index (`trackIndex`), track length (`trackTotMs`), and track position (`trackPosMs`) fields are valid. Otherwise, they should be ignored.

Table 6-66 `RetPlayStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x0F	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x10	Command ID: `RetPlayStatus`
5	0x*NN*	`playState` (1 byte). The iPod playback engine state. See Table 6-48 (page 144) for a list of possible values. If the value of this field is 0x00, all of the subsequent track fields are invalid. The value 0x05 is reserved.
6	0x*NN*	`trackIndex` (bits 31:24). Specifies the index of the currently playing track.
7	0x*NN*	`trackIndex` (bits 23:16)
8	0x*NN*	`trackIndex` (bits 15:8)
9	0x*NN*	`trackIndex` (bits 7:0)
10	0x*NN*	`trackTotMs` (bits 31:24). Specifies the total length of the track, in milliseconds.
11	0x*NN*	`trackTotMs` (bits 23:16)
12	0x*NN*	`trackTotMs` (bits 15:8)
13	0x*NN*	`trackTotMs` (bits 7:0)
14	0x*NN*	`trackPosMs` (bits 31:24). The current position of the track, in milliseconds.
15	0x*NN*	`trackPosMs` (bits 23:16)
16	0x*NN*	`trackPosMs` (bits 15:8)
17	0x*NN*	`trackPosMs` (bits 7:0)
18	0x*NN*	Checksum

Command 0x11: SetCurrentPlayingTrack

Direction: Device → iPod

The device sets the iPod's currently playing track to the track at the specified index. The total number of playing tracks can be obtained by sending "Command 0x14: GetNumPlayingTracks" (page 161). The playing track index is zero based, so the valid range is from 0x0 to `numPlayTracks`–1 (one less than the total count).

Table 6-67 `SetCurrentPlayingTrack` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x11	Command ID: `SetCurrentPlayingTrack`
5	0x*NN*	`trackIndex` (bits 31:24). The track index to begin playing.
6	0x*NN*	`trackIndex` (bits 23:16)
7	0x*NN*	`trackIndex` (bits 15:8)
8	0x*NN*	`trackIndex` (bits 7:0)
9	0x*NN*	Checksum

Command 0x12: GetIndexedPlayingTrackInfo

Direction: Device → iPod

The device requests track information for the specified playing track index. The `infoType` field specifies the type of information to be returned, such as track title, artist name, album name, track genre, and track chapter information. In response, the iPod sends "Command 0x13: RetIndexedPlayingTrackInfo" (page 159) with the requested track information.

Table 6-68 `GetIndexedPlayingTrackInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x09	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x12	Command ID: `GetIndexedPlayingTrackInfo`
5	0x*NN*	`infoType` (1 byte). The type of track information to retrieve. See Table 6-70 (page 160) for a list of the available types of information.
6	0x*NN*	`trackIndex` (bits 31:24). The index of the track for which to retrieve information.
7	0x*NN*	`trackIndex` (bits 23:16)

Byte number	Value	Comment
8	0x*NN*	`trackIndex` (bits 15:8)
9	0x*NN*	`trackIndex` (bits 7:0)
10	0x*NN*	`chapIndex` (bits 15:8). The index of the chapter for which to retrieve information. This field is valid only when `infoType` is chapter information (0x01) and the track at `trackIndex` has chapters. Set the `chapIndex` fields to 0x00 if `infoType` is not chapter information and `trackIndex` does not have chapters.
11	0x*NN*	`chapIndex` (bits 7:0)
12	0x*NN*	Checksum

Command 0x13: RetIndexedPlayingTrackInfo

Direction: iPod → Device

The iPod sends this command in response to "Command 0x12: GetIndexedPlayingTrackInfo" (page 158). It returns the requested type of information and data for the specified playing track. Data returned as strings are encoded as null-terminated UTF-8 character arrays. If the track information string does not exist, an empty null-terminated string is returned.

Table 6-69 `RetIndexedPlayingTrackInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x13	Command ID: `RetIndexedPlayingTrackInfo`
5	0x*NN*	`infoType`. The type of track information being returned. See Table 6-70 (page 160) for possible values.
6 … *N*	0x*NN*	`infoData` (variable length). The track information data.
(last byte)	0x*NN*	Checksum

Table 6-70 (page 160) shows the data returned for each type of track information.

Table 6-70 Track information data

Info type	Data type	Track information data	Data length
0x00	Track caps/info	Track capabilities and information. This contains 3 fields: ■ Bytes 0–3: Track Caps bitfields. Specifies the capabilities of the track. These bits have the following meanings: Bit 0: If equal to 1, the track is an audiobook. Bit 1: If equal to 1, the track has chapters. Bit 2: Set to 1 if album artwork is available, 0 otherwise Bit 3: If equal to 1, the track has song lyrics. Bit 6:4: Reserved Bit 7: Track contains video (a video podcast, music video, movie, or TV show) Bit 8: Track is currently queued to play as a video Bit 31:9: Reserved ■ Bytes 4–7: TrackTotMs. The total length of the track in milliseconds. ■ Bytes 8–9: ChapCount. If the track has chapters, the chapter count.	10
0x01	Chapter time/name	Chapter time and name, having two fields: ■ Bytes 0–3: Chapter Time The chapter time in milliseconds. A value of 0 indicates there are no chapters. ■ (variable length): Chapter Name The chapter name, as a UTF-8 string.	4 + Variable
0x02	Artist name	The artist name, as a UTF-8 string.	Variable
0x03	Album name	The album name, as a UTF-8 string.	Variable
0x04	Genre name	The genre name, as a UTF-8 string.	Variable
0x05	Track title	The title of the track, as a UTF-8 string.	Variable
0x06	Composer name	The composer name, as a UTF-8 string.	Variable

Info type	Data type	Track information data	Data length
0x07	Lyrics	Track lyrics data, consisting of 3 fields: ■ Byte 0: Packet information bits. If set, these bits have the following meanings: Bit 0: If equal to 1, indicates that this is one of multiple packets. Bit 1: If equal to 1, this is the last packet (applicable only if bit 0 is equal to 1). Bits 7:2: Reserved; set to 0. ■ Bytes 1–2: Packet index (16-bit big-endian format) ■ Bytes 3–*NN*: Track lyrics as a UTF-8 string.	Variable
0x08	Artwork count	Artwork count data. The artwork count is a sequence of 4-byte records; each record consists of a 2-byte `formatID` value followed by a 2-byte count of images in that format for this track. For information about `formatID` values, see "Command 0x17: RetArtworkFormats" (page 163).	Variable
0x09–0xFF	Reserved	N/A	N/A

Command 0x14: GetNumPlayingTracks

Direction: Device → iPod

The device requests the total number of tracks playing in the iPod playback engine. The count can be used to select a different playing track or obtain information for a specific track. In response, the iPod sends "Command 0x15: RetNumPlayingTracks" (page 162).

Table 6-71 `GetNumPlayingTracks` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x14	Command = 0x14 (`GetNumPlayingTracks`)
5	0xE7	Checksum

Command 0x15: RetNumPlayingTracks

Direction: iPod → Device

The iPod sends this command in response to "Command 0x14: GetNumPlayingTracks" (page 161) received from the device. It returns the total number of tracks queued in the playback engine.

Table 6-72 `RetNumPlayingTracks` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x15	Command ID: `RetNumPlayingTracks`
5	0x*NN*	`numPlayTracks` (bits 31:24). The total number of queued playing tracks.
6	0x*NN*	`numPlayTracks` (bits 23:16)
7	0x*NN*	`numPlayTracks` (bits 15:8)
8	0x*NN*	`numPlayTracks` (bits 7:0)
9	0x*NN*	Checksum

Command 0x16: GetArtworkFormats

Direction: Device → iPod

The device sends this command to obtain the list of supported artwork formats on the iPod. No parameters are sent. See "Transferring Album Art" (page 132).

Table 6-73 `GetArtworkFormats` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x03	Lingo ID: Display Remote lingo
4	0x16	Command ID: `GetArtworkFormats`
5	0xE5	Checksum

Command 0x17: RetArtworkFormats

Direction: iPod → Device

The iPod sends this command to the device in response to "Command 0x16: GetArtworkFormats" (page 162). Each format is described in a 7-byte record (`formatID:2`, `pixelFormat:1`, `width:2`, `height:2`). The `formatID` is used when sending `GetTrackArtworkTimes`. The device may return zero records if the iPod does not contain any artwork. See "Transferring Album Art" (page 132).

Table 6-74 `RetArtworkFormats` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x03	Lingo ID: Display Remote lingo
4	0x17	Command ID: `RetArtworkFormats`
NN	0x*NN*	`formatID` (bits 15:8) iPod-assigned value for this format
NN	0x*NN*	`formatID` (bits 7:0)
NN	0x*NN*	`pixelFormat`. See Table 6-75 (page 163).
NN	0x*NN*	`imageWidth` (bits 15:8). Number of pixels wide for each image.
NN	0x*NN*	`imageWidth` (bits 7:0)
NN	0x*NN*	`imageHeight` (bits 15:8). Number of pixels high for each image.
NN	0x*NN*	`imageHeight` (bits 7:0)
Previous 7 bytes may be repeated *NN* times		
NN	0x*NN*	Checksum

Table 6-75 Display pixel format codes

Display pixel format	Code
Reserved	0x00
Monochrome, 2 bits per pixel	0x01
RGB 565 color, little-endian, 16 bpp	0x02
RGB 565 color, big-endian, 16 bpp	0x03
Reserved	0x04–0xFF

Command 0x18: GetTrackArtworkData

Direction: Device → iPod

The device sends this command to the iPod to request the artwork image bitmap data for a given trackIndex, formatID, and artworkIndex. See "Transferring Album Art" (page 132).

Table 6-76 GetTrackArtworkData packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x0C	Length of packet
3	0x03	Lingo ID: Display Remote lingo
4	0x18	Command ID: GetTrackArtworkData
5	0x*NN*	trackIndex (bits 31:24).
6	0x*NN*	trackIndex (bits 23:16)
7	0x*NN*	trackIndex (bits 15:8)
8	0x*NN*	trackIndex (bits 7:0)
9	0x*NN*	formatID (bits 15:8)
10	0x*NN*	formatID (bits 7:0)
11	0x*NN*	Time offset in milliseconds (bits 31:24)
12	0x*NN*	Time offset in milliseconds (bits 23:16)
13	0x*NN*	Time offset in milliseconds (bits 15:8)
14	0x*NN*	Time offset in milliseconds (bits 7:0)
15	0x*NN*	Checksum

Command 0x19: RetTrackArtworkData

Direction: iPod → Device

The iPod sends this command in response to a "Command 0x18: GetTrackArtworkData" (page 164) command received from a device. Multiple RetTrackArtworkData commands may be necessary to transfer all the data because it will be too much to fit into a single packet. See "Transferring Album Art" (page 132).

This command returns the overall image width and height in pixels, the image row size in bytes, and the inset rectangle that contains the actual artwork content. The inset rectangle coordinates consist of two *x*,*y* pairs. Each *x* or *y* value is 2 bytes, so the total size of the inset rectangle coordinate set is 8 bytes.

The total image size in bytes is given by multiplying the row size by the image height. Padding may be inserted at the top, bottom, left, or right of the artwork to fit it to the iPod screen.

Table 6-77 `RetTrackArtworkData` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x03	Lingo ID: Display Remote lingo
4	0x19	Command ID: `RetTrackArtworkData`
5	0x*NN*	Descriptor telegram index (15:8). These fields uniquely identify each packet in the `RetTrackArtworkData` transaction. The first telegram is the descriptor telegram, which always starts with an index of 0x0000.
6	0x*NN*	Descriptor telegram index (7:0)
7	0x*NN*	Display pixel format code. See Table 6-75 (page 163).
8	0x*NN*	Image width in pixels (15:8)
9	0x*NN*	Image width in pixels (7:0)
10	0x*NN*	Image height in pixels (15:8)
11	0x*NN*	Image height in pixels (7:0)
12	0x*NN*	Inset rectangle, top-left point, *x* value (15:8)
13	0x*NN*	Inset rectangle, top-left point, *x* value (7:0)
14	0x*NN*	Inset rectangle, top-left point, *y* value (15:8)
15	0x*NN*	Inset rectangle, top-left point, *y* value (7:0)
16	0x*NN*	Inset rectangle, bottom-right point, *x* value (15:8)
17	0x*NN*	Inset rectangle, bottom-right point, *x* value (7:0)
18	0x*NN*	Inset rectangle, bottom-right point, *y* value (15:8)
19	0x*NN*	Inset rectangle, bottom-right point, *y* value (7:0)
20	0x*NN*	Row size in bytes (bits 31:24)

Byte number	Value	Comment
21	0x*NN*	Row size in bytes (bits 23:16)
22	0x*NN*	Row size in bytes (bits 15:8)
23	0x*NN*	Row size in bytes (bits 7:0)
24	0x*NN*...	Image pixel data (variable length)
NN	0x*NN*	Checksum

Note: In subsequent packets in the sequence (packets with a descriptor telegram index greater than 0x0000), bytes 7 through 23 are omitted.

Command 0x1A: GetPowerBatteryState

Direction: Device → iPod

The device sends this command to obtain the power and battery level state of the iPod. In response, the iPod sends "Command 0x1B: RetPowerBatteryState" (page 166) with the power and battery information.

Table 6-78 `GetPowerBatteryState` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x1A	Command ID: `GetPowerBatteryState`
5	0xE1	Checksum

Command 0x1B: RetPowerBatteryState

Direction: iPod → Device

The iPod sends this command in response to "Command 0x1A: GetPowerBatteryState" (page 166), returning the current iPod power state and battery level.

Note: If an external power status (indicated by the value of the `powerStat` field) is returned, the battery level is invalid and is 0 on return.

Table 6-79 `RetPowerBatteryState` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x1B	Command ID: `RetPowerBatteryState`
5	0x*NN*	`powerStat` (1 byte). The battery power state. See Table 6-51 (page 145).
6	0x*NN*	`battLevel` (1 byte). The battery power level. This is a value between 00 (the battery is fully discharged, no power remains) and 255 (the battery is fully charged). This field is valid only if `powerStat` is Internal battery (0x00 or 0x01)
7	0x*NN*	Checksum

Command 0x1C: GetSoundCheckState

Direction: Device → iPod

The device requests the iPod's current sound check setting. When enabled, sound check adjusts track playback volume to the same level. In response, the iPod sends "Command 0x1D: RetSoundCheckState" (page 168) with the current sound check state.

Table 6-80 `GetSoundCheckState` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x1C	Command ID: `GetSoundCheckState`
5	0xDF	Checksum

Command 0x1D: RetSoundCheckState

Direction: iPod → Device

The iPod sends this command in response to "Command 0x1C: GetSoundCheckState" (page 167) and returns the current state of the sound check setting.

Table 6-81 `RetSoundCheckState` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x1D	Command ID: `RetSoundCheckState`
5	0x*NN*	`sndChkState` (1 byte). The current state of the sound check setting. A value of 0x00 indicates that sound check is off; 0x01 indicates that it is on.
6	0x*NN*	Checksum

Command 0x1E: SetSoundCheckState

Direction: Device → iPod

The device sets the state of the iPod's sound check setting and optionally saves the previous sound check state to be restored on device detach. In response to this command, the iPod sends an `ACK` packet with the status of the command.

Table 6-82 `SetSoundCheckState` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet payload
3	0x03	Lingo ID: Display Remote lingo
4	0x1E	Command ID: `SetSoundCheckState`
5	0x*NN*	`sndChkState` (1 byte). The new state of the sound check setting. A value of 0x00 turns off sound check off; 0x01 turns on sound check.
6	0x*NN*	`bRestoreOnExit` (1 byte). Specifies whether the original sound check state is saved and restored on exit. See Table 6-62 (page 154)

Byte number	Value	Comment
7	0x*NN*	Checksum

Command 0x1F: GetTrackArtworkTimes

Direction: Device → iPod

The device sends this command to the iPod to request the list of artwork time offsets for a track. A 4-byte `trackIndex` specifies which track is to be selected. A 2-byte `formatID` indicates which type of artwork is desired. The format IDs that the iPod supports can be obtained using the "Command 0x16: GetArtworkFormats" (page 162) command.

The 2-byte `artworkIndex` specifies where to begin searching for artwork. A value of 0 indicates that the iPod should start with the first available artwork.

The 2-byte `artworkCount` specifies the maximum number of times to be returned. A value of -1 (0xFFFF) indicates that all artwork times from the specified index should be returned.

Table 6-83 `GetTrackArtworkTimes` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x0C	Length of packet
3	0x03	Lingo ID: Display Remote lingo
4	0x1F	Command ID: `GetTrackArtworkTimes`
5	0x*NN*	`trackIndex` (bits 31:24)
6	0x*NN*	`trackIndex` (bits 23:16)
7	0x*NN*	`trackIndex` (bits 15:8)
8	0x*NN*	`trackIndex` (bits 7:0)
9	0x*NN*	`formatID` (bits 15:8)
10	0x*NN*	`formatID` (bits 7:0)
11	0x*NN*	`artworkIndex` (bits 15:8)
12	0x*NN*	`artworkIndex` (bits 7:0)
13	0x*NN*	`artworkCount` (bits 15:8)
14	0x*NN*	`artworkCount` (bits 7:0)

Byte number	Value	Comment
15	0x*NN*	Checksum

Command 0x20: RetTrackArtworkTimes

Direction: iPod → Device

The iPod sends this command to the device in response to a "Command 0x1F: GetTrackArtworkTimes" (page 169) command. The device returns zero or more 4-byte records, one for each piece of artwork associated with the track and format specified by `GetTrackArtworkTimes`. Artwork times are expressed as offsets, in milliseconds, from the beginning of the track.

The number of records returned will be no greater than the number specified in the `GetTrackArtworkTimes` command. It may, however, be less than requested. This can happen if there are fewer pieces of artwork available than were requested, or if the iPod is unable to place the full number in a single packet. Check the number of records returned against the results of `RetIndexedPlayingTrackInfo` with `infoType` 0x08 to ensure that all artwork has been received.

Table 6-84 `RetTrackArtworkTimes` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x03	Lingo ID: Display Remote lingo
4	0x20	Command ID: `RetTrackArtworkTimes`
5	0x*NN*	Time offset in milliseconds (bits 31:24)
6	0x*NN*	Time offset in milliseconds (bits 23:16)
7	0x*NN*	Time offset in milliseconds (bits 15:8)
8	0x*NN*	Time offset in milliseconds (bits 7:0)
The preceding 4 bytes may be repeated *NN* times.		
NN	0x*NN*	Checksum

Lingo 0x05: RF Transmitter Lingo

The RF Transmitter lingo is used for devices that transmit the iPod analog audio over radio frequencies, typically over an unused frequency in the FM band. The Begin Transmission command packet notifies the external RF transmitter device that the iPod is entering playback mode. The End Transmission command packet notifies the RF transmitter that the iPod is exiting playback mode (that is, it is stopped, entering Light Sleep mode, and so forth).

This specification has facilities for devices that draw more than 5 mA power from the iPod. The conditions under which the device can draw more than 5 mA are listed in the options field of the General lingo "Command 0x01: Identify" (page 69).

Table 6-85 RF Transmitter lingo command summary

Command	ID	Data length	Protocol version	Authentication required
Reserved	0x00–0x01	N/A	N/A	N/A
Begin transmission	0x02	0x00	All	No
End transmission	0x03	0x00	All	No
Reserved	0x04–0xFF	N/A	N/A	N/A

Command History of the RF Transmitter lingo

Table 6-86 (page 171) shows the history of command changes in the RF Transmitter lingo:

Table 6-86 RF Transmitter lingo command history

Lingo version	Command changes	Features
(0.00)	Add: 0x02–0x03	Begin/end transmission support
1.00	0x04	Version number available through `RequestLingoProtocolVersion`
1.01	None	BugFix: `BeginTransmission` command sent after device inserted while iPod is playing

Command 0x02: Begin Transmission

Direction: iPod → Device

The iPod sends this command to notify the device that high power may be used and that it should begin transmitting. Upon receipt of the this command, the device may begin drawing more than 5 mA power, up to a maximum of 100 mA, if it has previously requested high power in the "Command 0x01: Identify" (page 69) packet options.

Table 6-87 Begin Transmission packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x05	Lingo ID: RF Transmitter lingo
4	0x02	Command ID: Begin Transmission
5	0xF7	Checksum

Command 0x03: End Transmission

Direction: iPod → Device

The iPod sends this command to notify the device to stop transmitting and to stop using accessory high power. The device must reduce its power usage to less than 5 mA within 1 second of receiving an End Transmission packet.

Note: Failure to reduce device power consumption to below 5 mA within 1 second after receiving this notification could damage the iPod.

Table 6-88 End Transmission packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet payload
3	0x05	Lingo ID: RF Transmitter lingo
4	0x03	Command ID: End Transmission
5	0xF6	Checksum

Lingo 0x06: USB Host Control Lingo

When an accessory is attached to an iPod, the iPod ordinarily acts as the USB device and the accessory acts as the USB host. The USB Host Control lingo lets the accessory switch roles with the iPod, becoming the USB device to the iPod's host. The USB Host Control lingo also automatically launches the Photo Import application on the iPod.

When an accessory identifies itself as using this lingo, the roles are switched automatically. If authentication fails or if the accessory is detached from the iPod, the iPod reverts to being the USB device.

Note: This lingo may be used only over the serial port link, not over the USB port link.

The accessory developer may choose to build a self-powered accessory that communicates with the iPod over USB or may choose to build a USB dongle similar to the Apple iPod Camera Connector.

Table 6-89 (page 173) summarizes the USB Host Control commands.

Table 6-89 USB Host Control lingo command summary

Command	ID	Direction	Data length	Protocol version	Authentication required
`ACK` (command/status)	0x00	Dev → iPod	4	1.00	Yes
`GetUSBPowerState`	0x01	iPod → Dev	2	1.00	Yes
`RetUSBPowerState`	0x02	Dev → iPod	3	1.00	Yes
`SetUSBPowerState`	0x03	iPod → Dev	3	1.00	Yes
Reserved	0x04–0xFF	N/A	N/A	N/A	N/A

Command History of the USB Host Control Lingo

Table 6-90 (page 173) shows the history of changes to the USB Host Control lingo.

Table 6-90 USB Host Control lingo command history

Lingo version	Command changes	Features
1.00	Add: 0x00–0x03	USB power state support

Command 0x00: ACK

Direction: Device → iPod

This command is sent by the device in response to a command received from the iPod. It reports the original command number and the status of the command.

Table 6-91 `ACK` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x04	Length of packet
3	0x06	Lingo ID: USB Host Control lingo
4	0x00	Command ID: ACK
5	0x*NN*	Command status (see Table 6-26 (page 125))
6	0x*NN*	ID of the command being acknowledged
7	0x*NN*	Checksum

Command 0x01: GetUSBPowerState

Direction: iPod → Device

This command is sent by the iPod to obtain the device's current USB power state. In response, the device must send a RetUSBPowerState command with the current USB power setting.

Table 6-92 GetUSBPowerState packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x06	Lingo ID: USB Host Control lingo
4	0x01	Command ID: GetUSBPowerState
5	0xF7	Checksum

Command 0x02: RetUSBPowerState

Direction: Device → iPod

This command is sent by the device in response to a GetUSBPowerState command received from the iPod. It returns the current state of the USB power supply.

Table 6-93 RetUSBPowerState packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x03	Length of packet
3	0x06	Lingo ID: USB Host Control lingo
4	0x02	Command ID: `RetUSBPowerState`
5	0x*NN*	Current power state (zero for off, nonzero for on)
6	0x*NN*	Checksum

Command 0x03: SetUSBPowerState

Direction: iPod → Device

This command is sent by the iPod to set the device's USB power state. The device must set the USB power state and respond with an `ACK` command indicating command completion.

Table 6-94 `SetUSBPowerState` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x06	Lingo ID: USB Host Control lingo
4	0x03	Command ID: `SetUSBPowerState`
5	0x*NN*	Power state to be set (zero for off, nonzero for on)
6	0x*NN*	Checksum

Lingo 0x07: RF Tuner Lingo

An external radio frequency tuner (RFT) accessory can be attached to an iPod to provide radio reception. When the iPod detects the RFT device, it adds a new menu item titled "Radio" to its top-level menu. Choosing this menu item displays the iPod tuner application. The application lets the user change the iPod's music source and control the accessory's RF band and tuner frequency.

The RF Tuner lingo is used to pass control and state information between an iPod and an RFT device. Normally the iPod is the master and the accessory responds to commands from the iPod. This means that the iPod can initiate actions such as controlling tuner power, setting the tuner's band and frequency, initiating up or down frequency scans, and so on. An iPod attached to an RFT device also stores station frequencies and other tuner state information.

Every RFT device must report a set of capabilities back to its attached iPod. Based on the capabilities reported by the RFT device, the iPod tuner application may choose to change its appearance to reflect the presence or absence of certain RFT features.

RFT Accessory Design

RFT accessory devices for iPods must meet the following requirements:

- All RFT lingo commands require authentication.
- An iPod can support only one RFT device at any time, and that device must be attached using the 30-pin connector.
- On reset or power up, the RFT device should be in its low power state (<5 mA) with the tuner off and audio output disabled.
- US devices must support the Europe/US FM band (87.5–108.0 MHz), with 200 KHz channel spacing and default 75 µs deemphasis.
- EU/US devices must support the Europe/US FM band (87.5–108.0 MHz), with 100 KHz channel spacing and selectable 50 µs or 75 µs deemphasis.
- JP devices must support the Japan FM band (76.0–90.0 MHz), with 100 KHz channel spacing and selectable 50 µs or 75 µs deemphasis.

An RFT device may support any combination of US, EU, and JP requirements.

The following options apply to RFT devices:

- An RFT device may send asynchronous notification of state changes to an attached iPod, but such notifications are not required.
- RFT devices are not expected to retain state information across accessory power down cycles resulting from the invocation of Hibernate and Deep Sleep modes.

RFT Power

Accessory device power management is important. As a device using the General lingo `IdentifyDeviceLingoes` command, an attached RFT device will be notified of iPod state changes such as transitions between Power On, Light Sleep, Hibernate, and Deep Sleep states. The RFT device is responsible for keeping its power consumption below the maximum allowed limits for each iPod state. Note that accessory power will be completely shut off when Hibernate or Deep Sleep states begin. When waking from a Light Sleep state the RFT device will be required to reidentify and reauthenticate itself, as is the case with other devices that use the `IdentifyDeviceLingoes` command and support authenticated lingoes.

Note: Currently, iPods send iAP notifications only for transitions between Light Sleep, Hibernate, and Power On states.

RFT Lingo Commands

Table 6-95 (page 177) summarizes the RF Tuner commands (lingo 0x07).

Table 6-95 RF Tuner lingo command summary

Command	ID	Direction	Data length	Protocol version	Authentication required
ACK	0x00	Dev → iPod	4 or 8	1.00	Yes
GetTunerCaps	0x01	iPod → Dev	2	1.00	Yes
RetTunerCaps	0x02	Dev → iPod	8	1.00	Yes
GetTunerCtrl	0x03	iPod → Dev	2	1.00	Yes
RetTunerCtrl	0x04	Dev → iPod	3	1.00	Yes
SetTunerCtrl	0x05	iPod → Dev	3	1.00	Yes
GetTunerBand	0x06	iPod → Dev	2	1.00	Yes
RetTunerBand	0x07	Dev → iPod	3	1.00	Yes
SetTunerBand	0x08	iPod → Dev	3	1.00	Yes
GetTunerFreq	0x09	iPod → Dev	2	1.00	Yes
RetTunerFreq	0x0A	Dev → iPod	7	1.00	Yes
SetTunerFreq	0x0B	iPod → Dev	6	1.00	Yes
GetTunerMode	0x0C	iPod → Dev	2	1.00	Yes
RetTunerMode	0x0D	Dev → iPod	3	1.00	Yes
SetTunerMode	0x0E	iPod → Dev	3	1.00	Yes
GetTunerSeekRssi	0x0F	iPod → Dev	2	1.00	Yes
RetTunerSeekRssi	0x10	Dev → iPod	3	1.00	Yes
SetTunerSeekRssi	0x11	iPod → Dev	3	1.00	Yes
TunerSeekStart	0x12	iPod → Dev	3	1.00	Yes
TunerSeekDone	0x13	Dev → iPod	7	1.00	Yes
GetTunerStatus	0x14	iPod → Dev	2	1.00	Yes

Command	ID	Direction	Data length	Protocol version	Authentication required
RetTunerStatus	0x15	Dev → iPod	3	1.00	Yes
GetStatusNotifyMask	0x16	iPod → Dev	2	1.00	Yes
RetStatusNotifyMask	0x17	Dev → iPod	3	1.00	Yes
SetStatusNotifyMask	0x18	iPod → Dev	3	1.00	Yes
StatusChangeNotify	0x19	Dev → iPod	3	1.00	Yes
GetRdsReadyStatus	0x1A	iPod → Dev	2	1.00	Yes
RetRdsReadyStatus	0x1B	Dev → iPod	6	1.00	Yes
GetRdsData	0x1C	iPod → Dev	3	1.00	Yes
RetRdsData	0x1D	Dev → iPod	0x*NN*	1.00	Yes
GetRdsNotifyMask	0x1E	iPod → Dev	2	1.00	Yes
RetRdsNotifyMask	0x1F	Dev → iPod	6	1.00	Yes
SetRdsNotifyMask	0x20	iPod → Dev	6	1.00	Yes
RdsReadyNotify	0x21	Dev → iPod	0x*NN*	1.00	Yes
Reserved	0x22–0xFF	N/A	N/A	N/A	N/A

Command History of the RFT Lingo

Table 6-96 (page 178) shows the history of changes to the RF Tuner lingo.

Table 6-96 RF Tuner lingo command history

Lingo version	Command changes	Features
1.00	Add: 0x00–0x21	RF tuner control and support

Command 0x00: ACK

Direction: Device → iPod

This command is sent by an RFT device on completion of a command received from an iPod. An ACK response is also sent if a bad parameter is received, an unsupported or invalid command is received, or a command that does not return any data has finished.

Note: Certain bytes of the RFT lingo `ACK` command packet are included only if the value of `cmdStatus` is 0x06. See Table 6-97 (page 179) and Table 6-98 (page 179).

Table 6-97 RFT lingo `ACK` packet with `cmdStatus` not 0x06

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x00	Command ID: `ACK`
5	0x*NN*	`cmdStatus`: Status of command received (see Table 6-99 (page 180))
6	0x*NN*	`cmdIDOrig`: ID of the command for which the response is being sent
7	0x*NN*	Checksum

Table 6-98 RFT lingo `ACK` packet with `cmdStatus` equal to 0x06

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x00	Command ID: `ACK`
5	0x06	`cmdStatus`: Command pending
6	0x*NN*	`cmdIDOrig`: ID of the command for which the response is being sent
7	0x*NN*	`cmdPendTime` (bits 31:24). Total timeout in milliseconds for the pending command to complete.
8	0x*NN*	`cmdPendTime` (bits 23:16)
9	0x*NN*	`cmdPendTime` (bits 15:8)
10	0x*NN*	`cmdPendTime` (bits 7:0)
11	0x*NN*	Checksum

Table 6-99 RFT lingo `ACK` command status values

Value	Meaning
0x00	Command OK
0x01	Unknown track category (not applicable)
0x02	Command failed (command valid but did not succeed)
0x03	Out of resources (iPod internal allocation failed)
0x04	Bad parameter (command or input parameters invalid)
0x05	Unknown track ID (not applicable)
0x06	Command pending (`cmdPendTime` parameter returned)
0x07	Not authenticated (iPod not authenticated)

Note: A value of 0x06 is usually returned in the `cmdStatus` byte of the RFT `ACK` packet for commands that require more than 100 ms to complete. In this case, the `cmdPendTime` parameter is included in bytes 7–10 of the `ACK` packet. If the completion status is not returned by the total number of milliseconds specified by these bytes, the iPod will assume that a command failure has occurred and will retry the command or otherwise recover from the error. The RFT device should not return any response to the command after this timeout period has expired.

Command 0x01: GetTunerCaps

Direction: iPod → Device

This command is sent by an iPod to query an attached RFT device's capabilities and determine what features the device supports. In response, the device must send a `RetTunerCaps` command with a payload specifying its capabilities.

Table 6-100 `GetTunerCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x01	Command ID: `GetTunerCaps`
5	0xF6	Checksum

Command 0x02: RetTunerCaps

Direction: Device → iPod

This command is sent by an RFT device in response to a `GetTunerCaps` command sent by an iPod. The command transmits a payload indicating which RFT capabilities it supports.

Table 6-101 `RetTunerCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x08	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x02	Command ID: `RetTunerCaps`
5	0x*NN*	Tuner capabilities (bits 31:24) (See Table 6-102 (page 181))
6	0x*NN*	Tuner capabilities (bits 23:16)
7	0x*NN*	Tuner capabilities (bits 15:8)
8	0x*NN*	Tuner capabilities (bits 7:0)
9	0x00	Reserved; set to 0x00
10	0x00	Reserved; set to 0x00
11	0x*NN*	Checksum

Note: The capabilities bits returned in the `RetTunerCaps` payload (bytes 5-8) correspond to the band, control, mode, and status commands of the RFT lingo. The iPod will enable features or use commands only if the RFT device sets the associated capabilities bits when sending this command.

Table 6-102 RFT device capabilities payload

Bits	Capability
00	Reserved; must be set to 0
01	FM band, Europe/US (87.5–108.0 MHz) capable
02	FM band, Japan (76.0–90.0 MHz) capable
07:03	Reserved; must be set to 00000
08	Tuner power on/off control capable

Bits	Capability
09	Status change notification capable
15:10	Reserved; must be set to 000000
17:16	Minimum FM resolution ID bits (see Table 6-103 (page 182))
18	Tuner seek up/down capable
19	Tuner seek RSSI threshold capable (should not be set if bit 18 is 0)
20	Force monophonic mode capable
21	Stereo blend capable
22	FM Tuner deemphasis select capable
23	Reserved; must be set to 0
24	Radio Data System (RDS/RBDS) data capable
25	Tuner channel RSSI indication capable
26	Stereo source indicator capable
31:27	Reserved; must be set to 00000

Table 6-103 Minimum FM resolution ID bits

ID bits	Description
00	200 kHz capable
01	100 kHz capable
10	50 kHz capable
11	Reserved

Note: The minimum FM resolution bits (bits 17-16) should report the smallest FM tuner resolution that the RFT device supports.

Command 0x03: GetTunerCtrl

Direction: iPod → Dev

An iPod sends this command to an RFT device to get the device's control state. In response, the device must send a `RetTunerCtrl` command with its current control state.

Table 6-104 `GetTunerCtrl` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x03	Command ID: `GetTunerCtrl`
5	0xF4	Checksum

Command 0x04: RetTunerCtrl

Direction: Device → iPod

An RFT device uses this command to send its current device control state in response to a `GetTunerCtrl` command from an iPod. Tuner control bits should be set only if the corresponding capabilities bits have been set in a previous `RetTunerCaps` command.

Table 6-105 `RetTunerCtrl` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x04	Command ID: `RetTunerCtrl`
5	0x*NN*	Tuner control state (see Table 6-106 (page 183))
6	0x*NN*	Checksum

Table 6-106 Tuner control state bits

Bit	Description
0	RFT device power is on (1) or off (0). When RFT device power is off, the accessory should rest in the lowest power state that still allows iAP commands to be received and processed.
1	Status change notification is enabled (1) or disabled (0). When status change notification is disabled, the iPod assumes that the device will send no asynchronous `StatusChangeNotify` commands. The iPod may poll the device for changes.

Bit	Description
7:2	Reserved; must be set to 000000

Command 0x05: SetTunerCtrl

Direction: iPod → Dev

This command is sent by an iPod to control an RFT device's state. In response, the device must return an ACK command with the command status. RFT state information, such as tuner frequency, band, and so on, must be preserved by the device across tuner on and off cycles, assuming accessory power is available.

When the tuner power is turned off, it must disable its audio output and reduce its power consumption to less than 5 mA. When tuner power is switched on, its power consumption may rise to the maximum declared by the options bits in a previous IdentifyDeviceLingoes command (see Table 5-32 (page 86)). If a device has specified that it requires intermittent accessory high power (up to 100 mA while on), the iPod will enable its accessory high power before sending a SetTunerCtrl command to turn on tuner power. The iPod accessory high power will remain on until a subsequent SetTunerCtrl command to turn off tuner power has finished.

Note: The iPod NotifyiPodStateChange command overrides this command for iPod transitions to a low power states (Light Sleep, Hibernate, and Deep Sleep). Even if this command has enabled RFT device power, the device must reduce its power consumption below 5 mA on receiving an iPod state change notification that specifies a transition to a low power state. The only iPod power state in which an RFT device requesting high power can consume more than 5 mA is the Power On state.

Table 6-107 SetTunerCtrl packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x05	Command ID: SetTunerCtrl
5	0x*NN*	Tuner control bits (see Table 6-108 (page 185))
6	0x*NN*	Checksum

Table 6-108 Tuner control bits

Bit	Description
0	Turn RFT device power on (1) or off (0). When RFT device power is turned off, the accessory should rest in the lowest power state that still allows iAP commands to be received and processed.
1	Enable (1) or disable (0) status change notification. When status change notification is disabled, the iPod assumes that the device will send no asynchronous `StatusChangeNotify` commands. The iPod may poll the device for changes.
7:2	Reserved

Command 0x06: GetTunerBand

Direction: iPod → Dev

This command is sent by an iPod to get an RFT device's RF band information. The device must respond by returning a `RetTunerBand` command.

Table 6-109 `GetTunerBand` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x06	Command ID: `GetTunerBand`
5	0xF1	Checksum

Command 0x07: RetTunerBand

Direction: Device → iPod

This command is sent by an RFT device to report its tuner band state in response to an iPod's `GetTunerBand` command. If the RFT device power is off, it should return its last active band state.

Note: Tuner band state information should be consistent with the device's capabilities, as previously reported by a `RetTunerCaps` command.

Table 6-110 `RetTunerBand` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x07	Command ID: `RetTunerBand`
5	0x*NN*	Tuner band state information (see Table 6-111 (page 186)):
6	0x*NN*	Checksum

Table 6-111 Tuner band state IDs

ID	Description
0x00	Reserved
0x01	Europe/US FM band (87.5–108.0 MHz)
0x02	Japan FM band (76.0–90.0 MHz)
0x03–0xFF	Reserved

Command 0x08: SetTunerBand

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to set its tuner band. In response, the device must send the iPod an `ACK` command with the command status. The command status must be reported as command failed (command status 0x02) if RFT device power is not on.

Table 6-112 `SetTunerBand` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet

Byte number	Value	Comment
3	0x07	Lingo ID: RF Tuner lingo
4	0x08	Command ID: SetTunerBand
5	0x*NN*	Tuner band to be set (see Table 6-111 (page 186))
6	0x*NN*	Checksum

Note: The tuner band setting requested by the iPod will be consistent with the RFT device's capabilities, as previously reported by a RetTunerCaps command.

Command 0x09: GetTunerFreq

Direction: iPod → Dev

This command is sent by an iPod to get an RFT device's current device tuner frequency and signal strength level. In response, the device must send the iPod a RetTunerFreq command.

Table 6-113 GetTunerFreq packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x09	Command ID: GetTunerFreq
5	0xEE	Checksum

Command 0x0A: RetTunerFreq

Direction: Device → iPod

This command is sent by an RFT device in response to a GetTunerFreq or SetTunerFreq command received from an iPod. The tuner frequency is expressed in kilohertz: for example, 76000 for 76.0 MHz, 87500 for 87.5 MHz, or 107900 for 107.9 MHz. Valid ranges are 87500 to 107900 for the European/US FM band and 76000 to 89900 for the Japanese FM band.

If the RFT device's capabilities includes tuner channel RSSI indication, byte 9 of the command packet may be a number greater than zero; otherwise it must be set to 0x00. The tuner RSSI level must be normalized to a value between 0 (minimum) and 255 (maximum). If RFT device power is off, the last active tuner frequency and signal strength (if applicable) should be sent.

Table 6-114 `RetTunerFreq` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x07	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x0A	Command ID: `RetTunerFreq`
5	0x*NN*	Tuner frequency in kilohertz (bits 31:24)
6	0x*NN*	Tuner frequency (bits 23:16)
7	0x*NN*	Tuner frequency (bits 15:8)
8	0x*NN*	Tuner frequency (bits 7:0)
9	0x*NN*	Tuner channel received signal strength (`rssiLevel`); may range from 0x00 (no signal present or RSSI indication not supported) to 0xFF (maximum RSSI signal strength).
10	0x*NN*	Checksum

Note: The tuner channel received signal strength value is valid only if the tuner RSSI bit was set in a previous `RetTunerCaps` command.

Command 0x0B: SetTunerFreq

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to select a specific tuner frequency within the current tuner band. The tuner frequency is expressed in kilohertz; valid ranges are 87500 to 107900 for the European/US FM band and 76000 to 89900 for the Japanese FM band. In response, the device must send a `RetTunerFreq` command with the new tuner frequency and RSSI level (if the device supports RSSI). The requested frequency should be within the currently selected tuner band. The command status must be reported as command failed (command status 0x02) if RFT device power is not on.

Table 6-115 `SetTunerFreq` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet

Byte number	Value	Comment
3	0x07	Lingo ID: RF Tuner lingo
4	0x0B	Command ID: SetTunerFreq
5	0x*NN*	Tuner frequency in kilohertz (bits 31:24)
6	0x*NN*	Tuner frequency (bits 23:16)
7	0x*NN*	Tuner frequency (bits 15:8)
8	0x*NN*	Tuner frequency (bits 7:0)
9	0x*NN*	Checksum

Command 0x0C: GetTunerMode

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to get the current tuner mode state from the device. In response, the device must send the iPod a RetTunerMode command.

Table 6-116 GetTunerMode packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x0C	Command ID: GetTunerMode
5	0xEB	Checksum

Command 0x0D: RetTunerMode

Direction: Device → iPod

This command is sent by a device in response to a GetTunerMode command received from an iPod. The tuner mode bits returned are valid only if the associated mode bits returned by a previous RetTunerCaps command were set. If tuner power is off, the last active tuner mode information should be returned.

Table 6-117 RetTunerMode packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x0D	Command ID: RetTunerMode
5	0x*NN*	Tuner mode status (see Table 6-118 (page 190))
6	0x*NN*	Checksum

Table 6-118 RF Tuner mode status bits

Bits	Description
1:0	FM tuner resolution (see Table 6-103 (page 182))
2	Tuner is currently seeking up or down
3	Tuner is currently seeking with an RSSI minumum threshold enabled
4	Monophonic mode is forced (1) or stereo is allowed (0)
5	Stereo blend is enabled (valid only if bit 4 is 0)
6	FM Tuner deemphasis is 50 µs (1) or 75 µs (0)
7	Reserved; must be set to 0

Command 0x0E: SetTunerMode

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to set the current tuner mode. In response, the device must send an RFT lingo ACK command with the command status. The command status must be reported as command failed (command status 0x02) if RFT device power is not on.

Table 6-119 SetTunerMode packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet

Byte number	Value	Comment
3	0x07	Lingo ID: RF Tuner lingo
4	0x0E	Command ID: SetTunerMode
5	0x*NN*	Tuner mode to be set (see Table 6-120 (page 191))
6	0x*NN*	Checksum

Table 6-120 Set RF Tuner mode bits

Bits	Description
1:0	Set FM tuner resolution (see Table 6-103 (page 182))
2	Reserved (use TunerSeekStart to start tuner seek)
3	Reserved (use TunerSeekStart to set tuner seek type)
4	Force monophonic mode (1) or allow stereo (0)
5	Enable stereo blend (valid only if bit 4 is 0); this blends the source left and right channels to improve the perceived signal quality while still retaining some stereo image separation
6	Set FM Tuner deemphasis to 50 µs (1) or 75 µs (0)
7	Reserved; set to 0

Command 0x0F: GetTunerSeekRssi

Direction: iPod → Dev

This command is sent by an iPod to get the current tuner seek threshold value from an RFT device. In response, the device must send a RetTunerSeekRssi command with the seek RSSI threshold, if supported by the device. If seek RSSI is not supported, the RFT device must return an ACK command with failure status.

Table 6-121 GetTunerSeekRssi packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x0F	Command ID: GetTunerSeekRssi
5	0xE8	Checksum

Command 0x10: RetTunerSeekRssi

Direction: Device → iPod

This command is sent by a device in response to a `GetTunerSeekRssi` command received from an iPod, assuming the device supports the seek RSSI capability. Its threshold value represents the minimum signal strength that allows a tuner channel to be recognized during the seek process; it is normalized to a value between 0 (minimum) and 255 (maximum). If device power is off, the last active RSSI threshold value should be sent.

Table 6-122 `RetTunerSeekRssi` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x10	Command ID: `RetTunerSeekRssi`
5	0x*NN*	RSSI threshold for seeking action
6	0x*NN*	Checksum

Command 0x11: SetTunerSeekRssi

Direction: iPod → Dev

This command is sent by an iPod to an RFT device that supports the seek RSSI capability, to set the device's seek RSSI signal strength threshold. The threshold value is the minimum signal strength that allows a tuner channel to be recognized during the seek process; it is normalized to a range between 0 (minimum) and 255 (maximum).

Table 6-123 `SetTunerSeekRssi` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x11	Command ID: `SetTunerSeekRssi`
5	0x*NN*	RSSI threshold for seeking action

Byte number	Value	Comment
6	0x*NN*	Checksum

Command 0x12: TunerSeekStart

Direction: iPod → Dev

This command is sent by an iPod to an RFT device that supports the tuner seek up/down capability (as reported by a previous `RetTunerCaps` command), to initiate seeking of a specified type. The returned `ACK` command status must be reported as command failed (command status 0x02) if RFT device power is not on. Seeking operations that use an RSSI threshold are initiated only if the RFT device's seek RSSI threshold capability is also supported, as reported by the `RetTunerCaps` command.

Table 6-124 `TunerSeekStart` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x12	Command ID: `TunerSeekStart`
5	0x*NN*	ID of tuner seeking operation (see Table 6-125 (page 193))
6	0x*NN*	Checksum

Table 6-125 Tuner seeking operations

ID	Operation	Use RSSI threshold
0x00	No seek operation (cancel seek operation, if active)	N/A
0x01	Seek up from beginning of band	No
0x02	Seek down from end of band	No
0x03	Seek up from current frequency	No
0x04	Seek down from current frequency	No
0x05	Seek up from beginning of band	Yes
0x06	Seek down from end of band	Yes
0x07	Seek up from current frequency	Yes
0x08	Seek down from current frequency	Yes

ID	Operation	Use RSSI threshold
0x09–0xFF	Reserved	N/A

A seek operation using an RSSI threshold completes when either of two conditions is satisfied:

- A channel was located within the band that satisfies the minimum RSSI threshold level.
- No channel was located within the band that satisfies the minimum the RSSI threshold level. The seek has traversed the entire band and wrapped back to the beginning tuner frequency without locating a valid channel. If no channel is found, it may indicate that the threshold is too high for the current radio reception area.

A seek operation using no RSSI threshold completes when either of two conditions is satisfied:

- A channel was located within the band that satisfies the criteria of the tuner's seek function. This may result in moving one or more channel spacings and wrapping around at the band ends.
- No channel was located within the band that satisfies the criteria of the tuner's seek function and the seek has traversed the entire band and wrapped back to the beginning tuner frequency without locating a valid channel.

An `ACK` command with command pending status is returned for seek operations requiring more than 100 ms to complete. It indicates the maximum time required for the device to complete the requested scan type. When the requested seek operation is completed (either successfully or unsuccessfully), the device must respond with a `TunerSeekDone` command indicating the seek operation status. If the device does not support tuner seek (with RSSI) operations or if tuner power is off, an `ACK` command with error status is returned.

When a cancel seek operation is requested and a seek operation is active, the seek operation stops at the current seek channel and the device responds with a `TunerSeekDone` command indicating the frequency and RSSI of the channel.

Command 0x13: TunerSeekDone

Direction: Device → iPod

In response to a `TunerSeekStart` command from an iPod, an RFT device must send this command to the iPod after the seek operation has finished. It reports the current tuner frequency, which is assumed to be the result of the seek operation. If the device supports a tuner channel RSSI indication capability (as reported by a previous `RetTunerCaps` command), the `rssiLevel` value shows the current channel's RSSI signal strength level. If no channel was found, an unsigned tuner frequency value of 0xFFFFFFFF should be reported. The received signal strength value must be normalized to a range from 0x00 (no signal or device does not support RSSI indication) to 0xFF (maximum signal strength).

Table 6-126 `TunerSeekDone` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)

Byte number	Value	Comment
1	0x55	Start of packet (SOP)
2	0x07	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x13	Command ID: `TunerSeekDone`
5	0x*NN*	Tuner frequency in kilohertz (bits 31:24)
6	0x*NN*	Tuner frequency in kilohertz (bits 23:16)
7	0x*NN*	Tuner frequency in kilohertz (bits 15:8)
8	0x*NN*	Tuner frequency in kilohertz (bits 7:0)
9	0x*NN*	Tuner channel RSSI received signal strength value
10	0x*NN*	Checksum

Command 0x14: GetTunerStatus

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to get its current tuner status state; in response, the device must send a `RetTunerStatus` command. The command can be used to poll the device's status if the device does not support status change notifications. After the device's status is reported its status bits must be cleared, so that an immediate reread will return no active status.

Table 6-127 `GetTunerStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x14	Command ID: `GetTunerStatus`
5	0xE3	Checksum

Command 0x15: RetTunerStatus

Direction: Device → iPod

This command is sent by an RFT device in response to a GetTunerStatus command received from an iPod. The tuner status bits returned are valid only if the status capability bits returned by a previous RetTunerCaps command were set. If the tuner power is off, the last active tuner status information should be returned.

Table 6-128 RetTunerStatus packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x15	Command ID: RetTunerStatus
5	0x*NN*	Tuner status bits (see Table 6-129 (page 196))
6	0x*NN*	Checksum

Table 6-129 RF tuner status bits

Bit	Description
0	RDS/RBDS data is received, ready to read
1	Tuner channel RSSI level has changed
2	Stereo source indicator state; 1 for a stereo signal source, 0 for monophonic
7:3	Reserved (must be set to 00000)

Command 0x16: GetStatusNotifyMask

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to get the status notification mask from the device. This mask indicates which state changes will invoke a notification change command from the device. In response, the device must send the iPod a RetStatusNotifyMask command.

Table 6-130 GetStatusNotifyMask packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet

Byte number	Value	Comment
3	0x07	Lingo ID: RF Tuner lingo
4	0x16	Command ID: `GetStatusNotifyMask`
5	0xE1	Checksum

Command 0x17: RetStatusNotifyMask

Direction: Device → iPod

This command must be returned by the device in response to the `GetStatusNotifyMask` command. The status notification mask indicates which state changes will invoke a notification change command from the device. However, its bit values are valid only if the corresponding capabilities bits returned by a previous `RetTunerCaps` command were set. A mask bit value of 1 indicates that notification is enabled; a value of 0 indicates notification is disabled or not supported.

Table 6-131 `RetStatusNotifyMask` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x17	Command ID: `RetStatusNotifyMask`
5	0x*NN*	Status notification mask (see Table 6-132 (page 197))
6	0x*NN*	Checksum

Table 6-132 Status notification mask bits

Bit	Description
0	RDS/RBDS data-ready change notify enabled (ignored if `SetRdsNotifyMask` sets `rdsMask` to a value other than zero)
1	Tuner channel RSSI-level change notify enabled
2	Stereo indicator state change notify enabled
7:3	Reserved (must be set to 00000)

Command 0x18: SetStatusNotifyMask

Direction: iPod → Dev

The iPod sends this command to an RFT device to set the status change notification mask. The status notification mask indicates which state changes will invoke a notification change command from the device. However, its bit values are valid only if the corresponding capabilities bits returned by a previous `RetTunerCaps` command were set. A mask bit value of 1 indicates that notification is enabled; a value of 0 indicates notification is disabled or not supported.

Note: For RDS/RBDS data-ready changes, the `SetRdsNotifyMask` command can override the notification status mask bit. If the `SetRdsNotifyMask` notification mask is set with a nonzero mask, then any enabled RDS/RBDS data notifications will be sent using the `RdsReadyNotify` command instead of the `StatusChangeNotify` command.

Table 6-133 `SetStatusNotifyMask` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x18	Command ID: `SetStatusNotifyMask`
5	0x*NN*	Status notification mask (see Table 6-134 (page 198))
6	0x*NN*	Checksum

Table 6-134 Status notification mask setting bits

Bit	Description
0	Enable RDS/RBDS data-ready change notification (ignored if `SetRdsNotifyMask` sets `rdsMask` to a value other than zero)
1	Enable tuner channel RSSI-level change notification
2	Enable stereo-indicator state change notification
7:3	Reserved (must be set to 00000)

Command 0x19: StatusChangeNotify

Direction: Device → iPod

This command must be sent asynchronously by an RFT device to an attached iPod to report each enabled status change. The iPod enables specific RFT status change notifications using the `SetStatusNotifyMask` command. After a notification has been sent, the status bits should be automatically cleared so they are ready to receive the next status change.

Note: Tuner channel RSSI-level change and stereo-indicator state change notifications should not occur more than 10 times per second.

Table 6-135 `StatusChangeNotify` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x19	Command ID: `StatusChangeNotify`
5	0x*NN*	Status change ID bits (see Table 6-136 (page 199))
6	0x*NN*	Checksum

Note: For RDS/RBDS data-ready changes, the `SetRdsNotifyMask` command can override the notification status mask bit. If the `SetRdsNotifyMask` notification mask is set with a nonzero mask, then any enabled RDS/RBDS data notifications will be sent using the `RdsReadyNotify` command instead of the `StatusChangeNotify` command.

Table 6-136 Status change ID bits

Bit	Description
0	RDS/RBDS data-ready change (valid only if bit 0 of `statusMask` is 1 and `SetRdsNotifyMask` sets `rdsMask` to 0x0)
1	Tuner channel RSSI-level change
2	Stereo indicator state change
7:3	Reserved (must be set to 00000)

Command 0x1A: GetRdsReadyStatus

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to get the device's current RDS/RBDS data-ready status. It can be used to poll the device's RDS/RBDS data-ready status without having to enable RDS/RBDS data-ready notifications. This command is usable only if the `RetTunerCaps` capability bits report has indicated that the device supports RDS/RBDS data reception and parsing. In response, the device must send a `RetRdsReadyStatus` command.

Table 6-137 `GetRdsReadyStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x1A	Command ID: `GetRdsReadyStatus`
5	0xDD	Checksum

Command 0x1B: RetRdsReadyStatus

Direction: Device → iPod

This command must be sent by an RFT device in response to a `GetRdsReadyStatus` command received from an iPod. Its status value indicates which RDS/RBDS data values are available to be read. All status bits must remain set on the device until the iPod sends a `GetRdsData` command to read the data. If the device does not support RDS/RBDS data, it must send an `rdsReady` data-ready status of 0 to indicate that no data is ready.

Table 6-138 `RetRdsReadyStatus` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x1B	Command ID: `RetRdsReadyStatus`
5	0x*NN*	RDS/RBDS data-ready status (bits 31:24) (see Table 6-139 (page 201))
6	0x*NN*	RDS/RBDS data-ready status (bits 23:16)
7	0x*NN*	RDS/RBDS data-ready status (bits 15:8)
8	0x*NN*	RDS/RBDS data-ready status (bits 7:0)

Byte number	Value	Comment
9	0x*NN*	Checksum

Table 6-139 RDS/RBDS data-ready status bits

Bit	Description
03-00	Reserved; set to zeros
04	RadioText (RT) data-ready
29-05	Reserved; set to zeros
30	Program Service Name (PSN) data-ready
31	Reserved; set to zero

Command 0x1C: GetRdsData

Direction: iPod → Dev

This command is sent by an iPod to get raw or unprocessed RDS/RBDS data from an RFT device that supports the RDS/RBDS data capability. If the device supports RDS/RBDS and has data-ready, it must send a `RetRdsData` command. If the device does not support RDS/RBDS or does not have the specified data type ready, it must return an `ACK` command with command failure status.

Table 6-140 `GetRdsData` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x1C	Command ID: `GetRdsData`
5	0x*NN*	RDS/RBDS data type (`rdsDataType`) ID (see Table 6-141 (page 201))
6	0x*NN*	Checksum

Table 6-141 RDS/RBDS data type IDs

ID Bytes	Description
0x00–0x03	Reserved
0x04	RadioText (RT)

ID Bytes	Description
0x05–0x1D	Reserved
0x1E	Program Service Name (PSN)
0x1F–0xFF	Reserved

Command 0x1D: RetRdsData

Direction: Device → iPod

This command is sent by an RFT device that has an RDS/RBDS data group ready in response to a `GetRdsData` command received from an iPod. If the device does not support the RDS/RBDS capability or does not have the specified data type ready, it must return an `ACK` command with command failure status (command status 0x02).

Table 6-142 `RetRdsData` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x1D	Command ID: `RetRdsData`
5	0x*NN*	`rdsDataType` (see Table 6-143 (page 202))
6	0x*NN*	`rdsData` (see Table 6-143 (page 202))
NN	0x*NN*	Checksum

Table 6-143 `rdsDataType` bytes and `rdsData` formats

rdsDataType bytes	Description	rdsData format
0x00–0x03	Reserved	N/A
0x04	RadioText (RT)	`charSet`:1 (see Table 6-144 (page 203)), `radioText`:64
0x05–0x1D	Reserved	N/A
0x1E	Program Service Name (PSN)	`charSet`:1 (see Table 6-144 (page 203)), `progSvcName`:8
0x1F–0xFF	Reserved	N/A

Note: The maximum length of a Radio Text is 64 bytes and that of a Program Service Name is 8 bytes.

Table 6-144 `rdsData` character set IDs

ID	Character set
0x00	Latin-based languages
0x01	Cyrillic- and Greek-based languages
0x02	Arabic- and Hebrew-based languages
0x03–0xFF	Reserved

Command 0x1E: GetRdsNotifyMask

Direction: iPod → Dev

This command is sent by an iPod to get an RFT device's current RDS/RBDS data notification mask (`rdsMask()`. In response, the device must send a `RetRdsNotifyMask` command with the RDS/RBDS data notification mask. This command is valid only if the RDS/RBDS data-ready capability bit was set in a previous `RetTunerCaps` command.

Table 6-145 `GetRdsNotifyMask` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x1E	Command ID: `GetRdsNotifyMask`
5	0xD9	Checksum

Command 0x1F: RetRdsNotifyMask

Direction: Device → iPod

This command is sent by an RFT device in response to a `GetRdsNotifyMask` command sent by an iPod. The RDS/RBDS mask (`rdsMask)` indicates which asychronous data notifications should be sent by the device when data is ready. For each data type bit, 1 means that data-ready notification is enabled and 0 means that notification is disabled. For enabled RDS/RBDS data types, the device must send `RdsReadyNotify` commands to the iPod when the associated data becomes available.

Table 6-146 `RetRdsNotifyMask` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x1F	Command ID: `RetRdsNotifyMask`
5	0x*NN*	RDS/RBDS data change notification mask (bits 31:24) (see Table 6-147 (page 204))
6	0x*NN*	RDS/RBDS data change notification mask (bits 23:16)
7	0x*NN*	RDS/RBDS data change notification mask (bits 15:8)
8	0x*NN*	RDS/RBDS data change notification mask (bits 7:0)
9	0x*NN*	Checksum

Table 6-147 RDS/RBDS data change notification mask bits

Bit	Description
03-00	Reserved (must be set to 0000)
04	RadioText (RT)
29-05	Reserved (must be set to zeros)
30	Program Service Name (PSN)
31	Reserved (must be set to 0)

Command 0x20: SetRdsNotifyMask

Direction: iPod → Dev

This command is sent by an iPod to an RFT device to set the device's RDS/RBDS data-ready notification mask (`rdsMask`). When notification bits are enabled, the device must send an `RdsReadyNotify` command with the associated RDS/RBDS data. For each RDS/RBDS data type bit, 1 enables data-ready notification and 0 disables notification.

Note: For RDS/RBDS data-ready changes, the SetRdsNotifyMask command can override the notification status mask bit. If the SetRdsNotifyMask notification mask is set with a nonzero mask, then any enabled RDS/RBDS data notifications will be sent using the RdsReadyNotify command instead of the StatusChangeNotify command.

Table 6-148 SetRdsNotifyMask packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x06	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x20	Command ID: SetRdsNotifyMask
5	0x*NN*	RDS/RBDS data change notification mask (bits 31:24) (see Table 6-149 (page 205))
6	0x*NN*	RDS/RBDS data change notification mask (bits 23:16)
7	0x*NN*	RDS/RBDS data change notification mask (bits 15:8)
8	0x*NN*	RDS/RBDS data change notification mask (bits 7:0)
9	0x*NN*	Checksum

Table 6-149 RDS/RBDS data change notification mask setting bits

Bit	Description
03-00	Reserved
04	RadioText (RT)
29-05	Reserved
30	Program Service Name (PSN)
31	Reserved

Command 0x21: RdsReadyNotify

Direction: Device → iPod

This command must be sent by an RFT device when RDS/RBDS data is ready, the associated SetRdsNotifyMask bit is set, and the device's capability bit from a previous RetTunerCaps command indicates that the device supports RDS/RBDS status notifications. After a notification command is sent, the device's associated RDS/RBDS data-ready status bit must cleared.

Table 6-150 `RdsReadyNotify` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x07	Lingo ID: RF Tuner lingo
4	0x21	Command ID: `RdsReadyNotify`
5	0x*NN*	`rdsDataType` (see Table 6-143 (page 202))
6	0x*NN*	`rdsData` (see Table 6-143 (page 202))
NN	0x*NN*	Checksum

Lingo 0x08: Accessory Equalizer Lingo

iPods may be connected to accessory devices such as boom boxes or amplifiers that are capable of frequency response equalization. The Accessory Equalizer lingo, number 0x08, lets the iPod control the Equalizer Settings of the accessory device.

Note: The Accessory Equalizer lingo requires device authentication for all communication transports (serial or USB).

The Accessory Equalizer lingo is similar to the Display Remote Equalizer lingo, but the command direction is reversed: the iPod is the master, initiating commands to which the device responds. The iPod can query the total count of device Equalizer Settings and the UTF-8 name for each one; it can then get or set each Equalizer Setting on the accessory device.

To support this lingo, the iPod application software will add an iPod menu item for accessory equalizer selection and control.

Equalizer Setting Requirements

To be compatible with the Accessory Equalizer lingo, an accessory device must observe these rules:

- The accessory must have an Equalizer Setting with an index of 0x00 and this must be the accessory equalizer's off/none setting.
- The Equalizer Index and Equalizer Setting count fields are 1 byte in size, limiting the setting count to a maximum of 255, including the required index 0x00 setting.
- Devices supporting this lingo must support at least two Equalizer Settings: off/none and at least one other setting. If fewer than two settings are returned by the device, the iPod will not present equalizer menu options to the user.

Accessory Equalizer Lingo Commands

Table 6-151 (page 207) lists the commands included in the Accessory Equalizer lingo.

Table 6-151 Accessory Equalizer lingo command summary

Command	ID	Direction	Data length	Protocol version	Authentication required
ACK	0x00	Dev → iPod	4	1.00	Yes
GetCurrentEQIndex	0x01	iPod → Dev	2	1.00	Yes
RetCurrentEQIndex	0x02	Dev → iPod	3	1.00	Yes
SetCurrentEQIndex	0x03	iPod → Dev	3	1.00	Yes
GetEQSettingCount	0x04	iPod → Dev	2	1.00	Yes
RetEQSettingCount	0x05	Dev → iPod	3	1.00	Yes
GetEQIndexName	0x06	iPod → Dev	3	1.00	Yes
RetEQIndexName	0x07	Dev → iPod	0x*NN*	1.00	Yes
Reserved	0x08–0xFF	N/A	N/A	N/A	N/A

Command History of the Accessory Equalizer Lingo

Table 6-152 (page 207) shows the history of command changes in the accessory equalizer lingo:

Table 6-152 Accessory Equalizer lingo command history

Lingo version	Command changes	Features
1.00	Add: 0x00–0x07	Accessory device Equalizer Index, count, and name support

Command 0x00: ACK

Direction: Device → iPod

This command is sent by the device in response to a command sent from the iPod. An ACK response is sent when a bad parameter is received, an unsupported/invalid command is received, or a command completed but did not return any data. See Table 6-153 (page 207).

Table 6-153 ACK packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)

Byte number	Value	Comment
1	0x55	Start of packet (SOP)
2	0x04	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x00	Command ID: ACK
5	0x*NN*	Status of command received (see Table 6-26 (page 125))
6	0x*NN*	ID of the command for which the response is being sent
7	0x*NN*	Checksum

Command 0x01: GetCurrentEQIndex

Direction: iPod → Device

This command is sent by the iPod to request the current Equalizer Setting from the device. In reponse, the device sends a RetCurrentEQIndex command with the current Equalizer Index.

The timeout for this command is 2500 ms (2.5 seconds). If the device does not respond within the allotted time, the iPod will retry up to three times. If the device does not respond within the specified time and retry count, the command will fail.

Table 6-154 GetCurrentEQIndex packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x01	Command ID: GetCurrentEQIndex
5	0xF5	Checksum

Command 0x02: RetCurrentEQIndex

Direction: Device → iPod

This command is sent by the device in response to the GetCurrentEQIndex command received from the iPod. The Equalizer Index returned may range from 0 (reserved for the off/none Equalizer Setting) to 254 (the maximum possible Equalizer Index). Devices are not required to support 255 settings; they may support as few as the two minimum required settings.

Table 6-155 `RetCurrentEQIndex` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x02	Command ID: `RetCurrentEQIndex`
5	0x*NN*	`eqIndex`: The current accessory Equalizer Index. See Table 6-156 (page 209).
6	0x*NN*	Checksum

Table 6-156 Device Equalizer Setting indices

Index	Description
0x00	Equalizer off/none
0x01	First Equalizer Setting
0x*NN*	Last Equalizer Setting (accessory-dependent maximum index)

Command 0x03: SetCurrentEQIndex

Direction: iPod → Device

This command is sent by the iPod to select an accessory Equalizer Index from the supported range. The valid range is from 0 to one less than the Equalizer Setting count returned by the `RetEQSettingCount` command. See Table 6-156 (page 209).

Note: Accessories may receive duplicate `SetCurrentEQIndex` commands at any time and must handle them correctly.

The timeout for this command is 3000 ms (3.0 seconds). If the device does not respond within the allotted time, the iPod will retry up to three times. If the device does not respond within the specified time and retry count, the command will fail.

Table 6-157 `SetCurrentEQIndex` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)

Byte number	Value	Comment
2	0x03	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x03	Command ID: `SetCurrentEQIndex`
5	0x*NN*	`eqIndex`: the selected accessory Equalizer Index
6	0x*NN*	Checksum

Command 0x04: GetEQSettingCount

Direction: iPod → Device

This command is sent by the iPod to determine how many Equalizer Settings the device supports. In response, the device sends a `RetEQSettingCount` command with the count of Equalizer Settings supported.

The timeout for this command is 3000 ms (3.0 seconds). If the device does not respond within the allotted time, the iPod will retry up to three times. If the device does not respond within the specified time and retry count, the command will fail.

Table 6-158 `GetEQSettingCount` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x04	Command ID: `GetEQSettingCount`
5	0xF2	Checksum

Command 0x05: RetEQSettingCount

Direction: Device → iPod

This command is sent by the device in response to a `GetEQSettingCount` command from the iPod. To support the Accessory Equalizer lingo, a device must report a minimum count of 0x02 (equalizer off/none plus one other setting) and may support a maximum count of 0xFF (equalizer off/none plus 254 other settings).

Table 6-159 `RetEQSettingCount` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x05	Command ID: `RetEQSettingCount`
5	0x*NN*	`eqCount`: the count of accessory equalizer indices. See Table 6-160 (page 211).
6	0x*NN*	Checksum

Table 6-160 `RetEQSettingCount` parameter values

Range	Comment
0x00–0x01	Invalid equalizer count (minimum count is 0x02)
0x02–0xFF	Valid equalizer count range

Command 0x06: GetEQIndexName

Direction: iPod → Device

This command is sent by the iPod to obtain the name string associated with a specified Equalizer Index. In response, the device sends a `RetEQIndexName` command with the same Equalizer Index and the associated Equalizer Setting name string.

The timeout for this command is 3000 ms (3.0 seconds). If the device does not respond within the allotted time, the iPod will retry up to three times. If the device does not respond within the specified time and retry count, the command will fail.

Table 6-161 `GetEQIndexName` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x06	Command ID: `GetEQIndexName`
5	0x*NN*	`eqIndex`: the selected accessory Equalizer Index

Byte number	Value	Comment
6	0x*NN*	Checksum

Command 0x07: RetEQIndexName

Direction: Device → iPod

This command is sent by the device in response to a `GetEQIndexName` command received from the iPod. The `eqIndex` byte is the same index that was sent by the `GetEQIndexName` command. The `eqName` string is a null-terminated UTF-8 character array. The array length must not exceed 32 characters plus a null terminator character. This length limit minimizes truncation of the Equalizer Setting name when it is displayed in the iPod accessory Equalizer Settings menu.

Table 6-162 `RetEQIndexName` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x04	Length of packet
3	0x08	Lingo ID: Accessory Equalizer lingo
4	0x07	Command ID: `RetEQIndexName`
5	0x*NN*	`eqIndex`: the selected accessory Equalizer Index
6-*NN*	0x*NN*...	The accessory equalizer name, as a null-terminated UTF-8 character array
(last byte)	0x*NN*	Checksum

Lingo 0x0A: Digital Audio Lingo

The Digital Audio lingo supports iAP commands, using either USB or UART transport, that let the iPod transfer digital audio to an accessory. The iPod uses these lingo commands to retrieve a list of supported sample rates from the accessory and to inform the accessory of the iPod's current sample rate, Sound Check value, and track volume adjustment value (which is set using iTunes). After these interchanges, the iPod performs audio sample rate conversion internally and transfers digital audio to the accessory device at one of the device's supported audio data sample rates.

Note: The iPod models that contain version 1.00 of the Digital Audio lingo do not correctly support digital audio. An accessory should check the attached iPod's version of the Digital Audio lingo and use digital audio only if the version number is greater than 1.00. See Table 3-1 (page 43) for a list of affected iPod models.

Accessory Authentication

Every accessory device that supports the Digital Audio lingo must authenticate itself with a connected iPod as soon as the iPod recognizes the device; deferred authentication is not permitted. This means that the accessory must be authenticated through the General lingo `IdentifyDeviceLingoes` command with the authentication control bits in its `Options` field set to 10_b (Authenticate immediately after identification). See Table 6-30 (page 127) for details.

Note: The General lingo `Identify` command cannot be used to identify an accessory device for any purpose if the device supports the Digital Audio lingo.

When the accessory identifies itself as supporting the Digital Audio lingo, authentication can happen in the background and the iPod can proceed to transfer digital audio as if authentication were successful.

The Digital Audio lingo ID is 0x0A. All its commands are transferred in small packet format and all require authentication. They are listed in Table 6-163 (page 213).

Table 6-163 Digital Audio lingo command summary

ID	Command	Data length	Protocol version	Authentication required
0x00	`AccAck`	0x04	1.00	Yes
0x01	`iPodAck`	0x04	1.00	Yes
0x02	`GetAccSampleRateCaps`	0x02	1.00	Yes
0x03	`RetAccSampleRateCaps`	*NN*	1.00	Yes
0x04	`NewiPodTrackInfo`	0x0E	1.00	Yes
0x05	`SetVideoDelay`	0x06	1.03	Yes
0x06–0xFF	`Reserved`	N/A	N/A	N/A

USB Audio Transport

Digital audio on the iPod supports USB Audio 1.0, with the addition that it will transfer digital audio on a High-Speed USB 2.0 bus as well as on a Full-Speed USB 1.1 bus. During transport, the accessory is the USB host, the iPod is the USB device, and PCM data is synchronized with the 1 ms USB start-of-frame (SOF) packets. More information about USB and its standards is available at `http://www.usb.org`.

When receiving digital audio over a USB audio streaming interface, the accessory may buffer data to achieve consistent audio playback. Since buffering introduces latency, it is suggested that this latency be kept below 200 ms to ensure timely response to user input. Apple may enforce this suggestion in the future.

Digital Audio and Extended Interface Mode

Digital Audio lingo version 1.01 requires the iPod to be in Extended Interface mode before USB audio can be transferred. With versions 1.02 and later, USB audio can be transferred in any mode.

Audio/Video Synchronization for Digital Audio

Digital Audio lingo versions 1.03 and above allow synchronization between iPod video and the sound for that video that is being transmitted as digital audio. The video may either appear on the iPod's display or be transmitted through the analog video output. For synchronization to occur, the accessory must use the `SetVideoDelay` command to send a fixed time (in milliseconds) that the iPod will add as a delay to its own video. The delay should be equal to the audio latency of the accessory; it corresponds to the difference between the time that digital audio data appears on the digital audio transport (USB) and the time the corresponding analog audio is sent to the speakers. The accessory must resend the `SetVideoDelay` command whenever its audio latency changes, for example at sample rate changes.

Line Level Output and Digital Audio Transport

Line level output and digital audio transport are mutually exclusive. All iPod audio is sent to the LINE-OUT pins of the 30-pin connector until digital audio transport is enabled, at which point the iPod's audio output is redirected to the USB audio streaming interface. When digital audio transport is disabled, iPod audio is again sent to the LINE-OUT pins (see Table 6-1 (page 107)).

When disabling digital audio transport, the iPod does not automatically enter the playback Pause state, but instead continues in its current Play or Pause state. The only difference is that the output mode changes to LINE-OUT. If appropriate, the accessory should set the iPod to Pause when disabling digital audio.

Enabling and Disabling USB Audio

Digital audio is enabled and LINE-OUT audio is disabled when:

- Version 1.01: The accessory selects a nonzero-bandwidth alternate USB audio streaming interface and enters Extended Interface mode.
- Version 1.02: The accessory replies to a `GetAccSampleRateCaps` command with a list of valid sample rates.

Digital audio is disabled and LINE-OUT audio is enabled when:

- Version 1.01: The accessory selects a zero-bandwidth alternate USB audio streaming interface, reidentifies itself without supporting the Digital Audio lingo, or disconnects from the USB Audio Configuration (that is, it disconnects the USB cable).
- Version 1.02: The accessory either reidentifies itself without supporting the Digital Audio lingo or disconnects from the USB Audio Configuration (that is, it disconnects the USB cable).

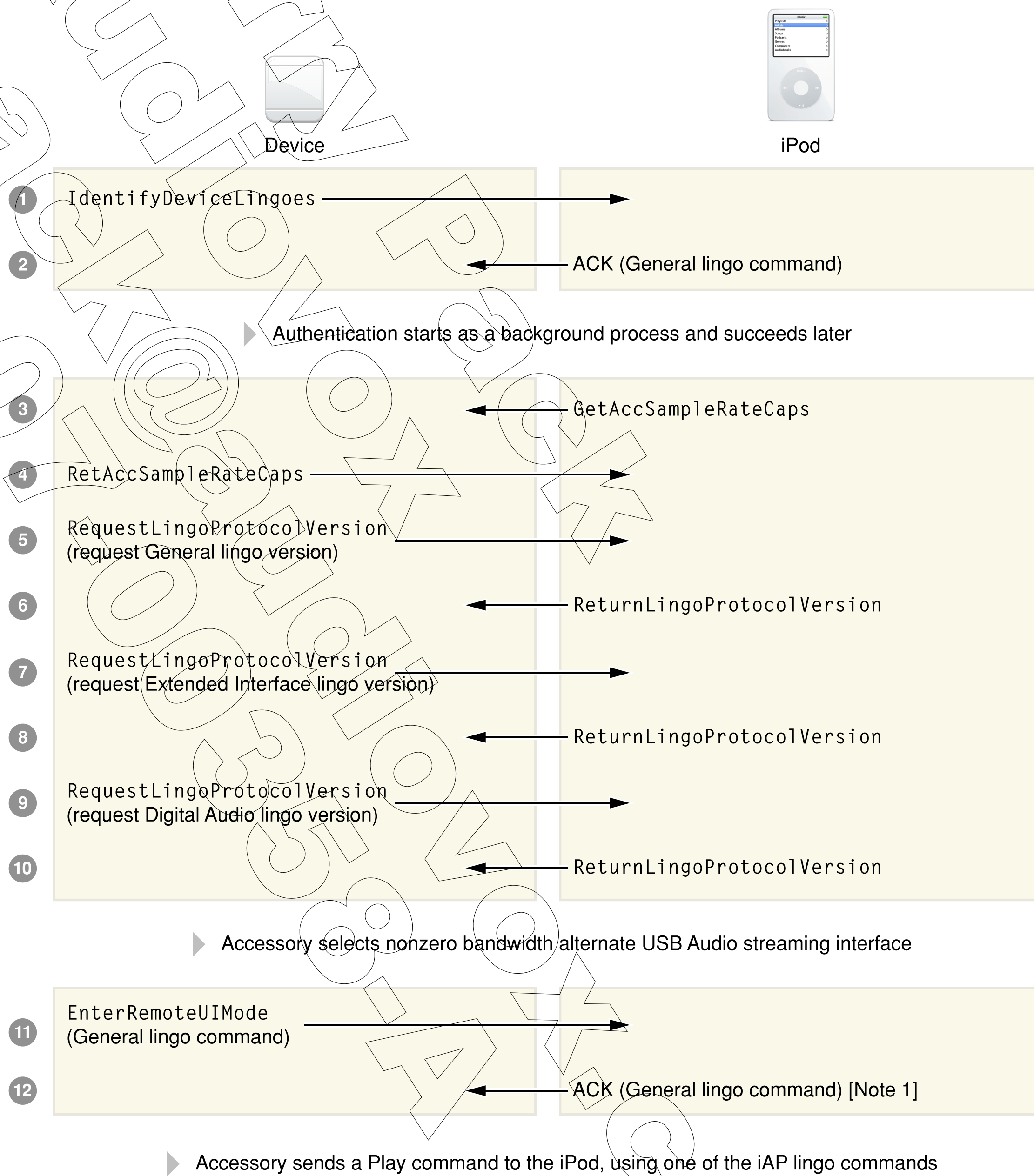
To enable USB audio, an iPod and its attached accessory must perform the following steps, in the order listed, after the user has connected the iPod to the accessory:

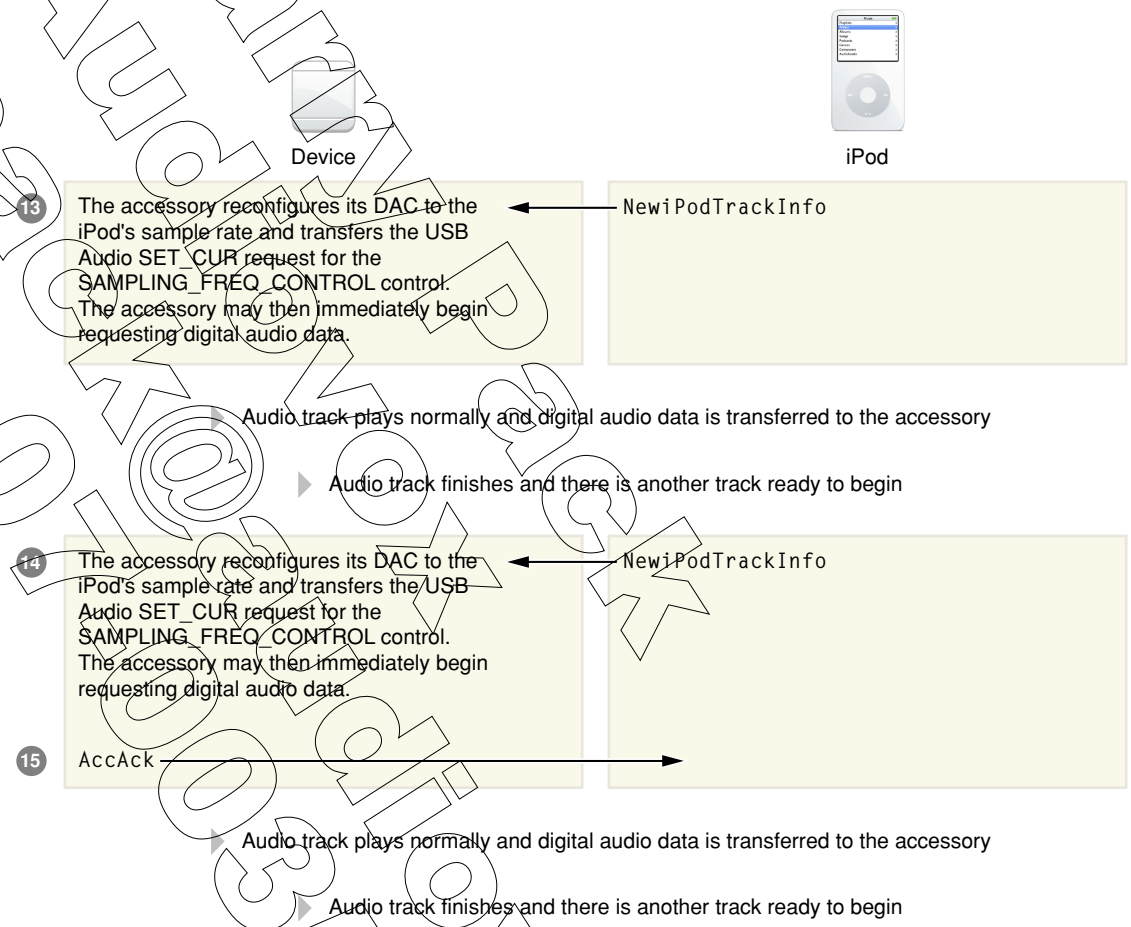
1. The iPod's USB enumeration lists two configurations: 1 = Mass Storage, 2 = USB Audio and HID device.
2. The device sends the USB standard request `Set_Configuration` to select configuration 2.
3. The device sends the `IdentifyDeviceLingoes` General lingo command with the Digital Audio, Extended Interface, and General lingo bits. (The Extended Interface bit is not required with Digital Audio lingo version 1.02). The iPod authenticates the device for the requested lingoes.
4. The device successfully responds to the iPod `GetAccSampleRateCaps` command.
5. The device selects a nonzero-bandwidth alternate USB audio streaming interface on the iPod.
6. The device tells the iPod to enter Extended Interface Mode using the General lingo `EnterRemoteUIMode` command (not required for Digital Audio lingo version 1.02). If an audio track is playing, playback will pause. If a video track is playing, playback will stop.
7. The device places the iPod in the Play state.
8. The iPod sends a `NewiPodTrackInfo` command to the device.
9. The device acknowledges the `NewiPodTrackInfo` command using the `AccAck` command.
10. The device configures its playback DAC to the iPod's sample rate.
11. The device sends a USB audio `SET_CUR` request for the `SAMPLING_FREQ_CONTROL` control, passing a sample rate equal to the value sent by the `NewiPodTrackInfo` command.
12. The device requests digital audio packets from the isochronous USB pipe.

Note: Step 2 is slightly different if a 191 kΩ resistor is used as the identify resistor (see Figure 2-11 (page 40)). In this case, the iPod will present the USB audio and HID device configuration as the first configuration, and the accessory will be required to select configuration 1 through the `Set_Configuration` USB standard request.

An example of this process is diagrammed in Figure 6-1. The transactions shown assume that USB enumeration and device authentication are successful.

Figure 6-1 Typical digital audio transactions between an iPod and an accessory





Note 1: The `ACK` reply to an `EnterRemoteUIMode` command may return a command pending acknowledgment along with a waiting time.

Note: With Digital Audio lingo version 1.01, switching from the zero bandwidth alternate USB audio streaming interface to the nonzero bandwidth interface without first exiting and entering Extended Interface Mode can prevent a `NewiPodTrackInfo` command from being sent from the iPod when a track change occurs. The accessory should never select the zero-bandwidth alternate interface on the iPod, even when changing sample rates, unless it is also exiting Extended Interface Mode.

With Digital Audio lingo version 1.01, reenabling digital audio after an accessory has disabled it requires the following actions:

1. Enable the nonzero-bandwidth alternate USB audio streaming interface on the iPod.
2. Enter Extended Interface Mode using the General lingo `EnterRemoteUIMode`. This will pause iPod playback.
3. Place the iPod in the Play state.

4. Receive a `NewiPodTrackInfo` command from the iPod.

5. Acknowledge the `NewiPodTrackInfo` command using the `AccAck` command.

6. Configure the accessory's playback DAC to the iPod's sample rate.

7. Send a USB audio `SET_CUR` request for the `SAMPLING_FREQ_CONTROL` control, passing a sample rate equal to the value sent by the `NewiPodTrackInfo` command.

8. Request digital audio packets from the isochronous USB pipe.

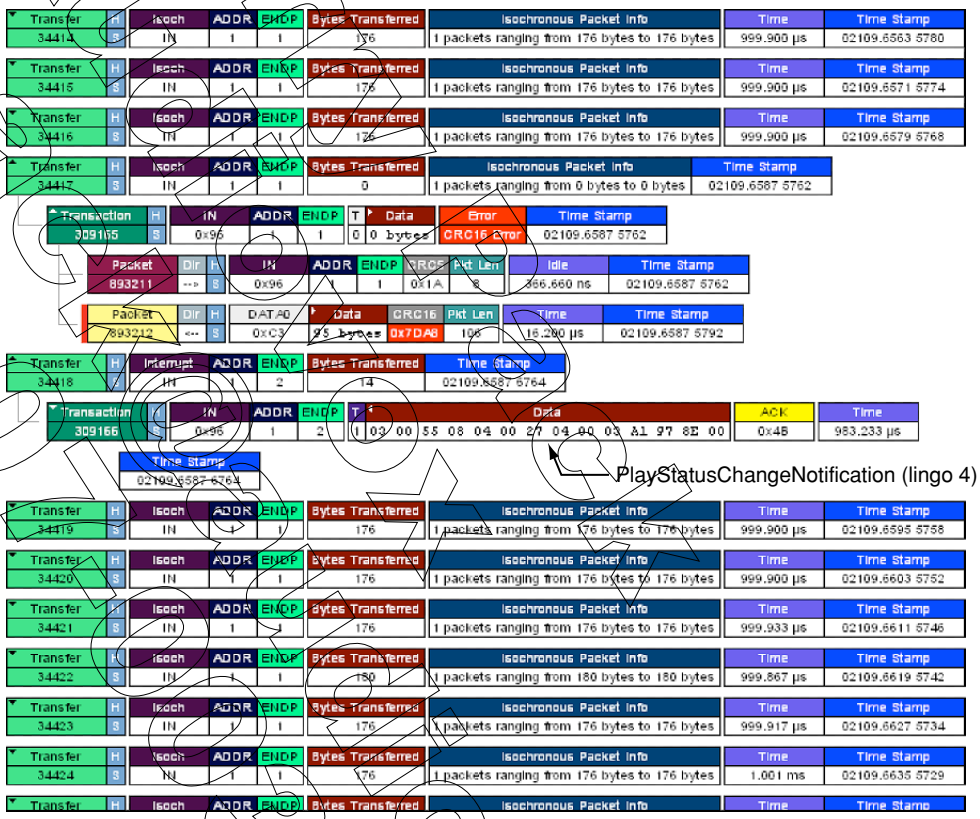
If the accessory requests digital audio data from the isochronous USB pipe before digital audio is enabled or before the correct digital sample rate has been negotiated, the iPod will return isochronous packets filled with zeros. The iPod will also return packets filled with zeros if authentication fails.

USB Audio Errors on Older iPods

On first-generation nano and 5G iPods, USB Audio data from the iPod will occasionally contain 16-bit cyclic redundancy check (CRC16) errors. If iAP commands are sent from the iPod to a USB host while USB audio is enabled, there is a possibility that the USB audio DATA0 packet immediately preceding the iAP command will contain a CRC16 error. This error occurs infrequently; however, enabling iPod notifications over USB (such as the Lingo 0x04 `PlayStatusChangeNotification` command, which is sent every 500 ms when enabled) greatly increases the possibility that a CRC16 error will occur.

To resolve this problem, the USB host must be able to recover immediately from a CRC16 error in the payload of the isochronous USB audio data. Figure 6-2 (page 219) illustrates a typical recovery sequence.

Figure 6-2 A USB host recovers from a CRC16 error



Digital Audio Lingo Commands

Command History of the Digital Audio Lingo

Table 6-164 (page 219) shows the history of command changes in the Digital Audio lingo:

Table 6-164 Digital Audio lingo command history

Lingo version	Command changes	Features
1.00	Add: 0x00–0x04	Digital audio sample rate and track information support
1.01	None	The `NewiPodTrackInfo` command is resent until it is acknowledged by the USB host with an `AccAck` command. Also corrected a bug where `NewiPodTrackInfo` was not being sent before every track; for this reason, accessories should not use Digital Audio with iPods that support only Digital Audio lingo version 1.00.
1.02	None	The Digital Audio lingo no longer requires the iPod to be in Extended Interface mode.

Lingo version	Command changes	Features
1.03	Add 0x05	Added `SetVideoDelay` to allow digital audio to synchronize with video playback.

Command 0x00: AccAck

Direction: Device → iPod

The accessory sends the `AccAck` command to acknowledge the receipt of a command from the iPod and returns the command status (see Table 6-166 (page 220)). An `AccAck` command should be sent only for commands whose documentation specifies that an `AccAck` command is needed.

Table 6-165 `AccAck` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x04	Length of packet payload
3	0x0A	Lingo ID: Digital audio lingo
4	0x00	Command ID: `AccAck`
5	0x*NN*	Command status. See Table 6-166 (page 220)
6	0x*NN*	The ID for the command being acknowledged
7	0x*NN*	Checksum

Table 6-166 `AccAck` status values

status	Meaning
0x00	Success (OK)
0x01	Reserved
0x02	ERROR: Command failed
0x03	ERROR: Out of resources
0x04	ERROR: Bad parameter
0x05	ERROR: Unknown ID
0x06	Reserved
0x07	ERROR: Accessory not authenticated

status	Meaning
0x08–0xFF	Reserved

Command 0x01: iPodAck

Direction: iPod → Device

The iPod sends the `iPodAck` command when it receives an invalid or unsupported command or a bad parameter. The command status returns are the same as those used for the `AccAck` command (see Table 6-166 (page 220)).

Table 6-167 `iPodAck` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x04	Length of packet payload
3	0x0A	Lingo ID: Digital audio lingo
4	0x01	Command ID: `iPodAck`
5	0x*NN*	Command status. See Table 6-166 (page 220)
6	0x*NN*	The ID for the command being acknowledged
7	0x*NN*	Checksum

Command 0x02: GetAccSampleRateCaps

Direction: iPod → Device

The iPod uses this command to request the list of supported sample rates from the accessory. The iPod sends it after the accessory announces that it supports the Digital Audio lingo using the General lingo `IdentifyDeviceLingoes` command. The accessory responds to this command using the `RetAccSampleRateCaps` command.

After the iPod sends a `GetAccSampleRateCaps` command, it waits up to 3 seconds for a `RetAccSampleRateCaps` command from the accessory before timing out. If a timeout occurs, the iPod resends the command. It will resend the command up to three times if necessary, and if all three attempts fail, the iPod will then respond to any digital audio request with zeros.

If the iPod receives a `RetAccSampleRateCaps` command with invalid parameters, it sends the appropriate command status using an `iPodAck` command (see Table 6-166 (page 220)). At this point the accessory should send a `RetAccSampleRateCaps` with correct parameters. The iPod allows up to three retries if necessary, and if all three attempts fail, the iPod then responds to any digital audio request with zeros.

Table 6-168 `GetAccSampleRateCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x02	Length of packet payload
3	0x0A	Lingo ID: Digital audio lingo
4	0x02	Command ID: `GetAccSampleRateCaps`
5	0xF2	Checksum

Command 0x03: RetAccSampleRateCaps

Direction: Device → iPod

An accessory sends the `RetAccSampleRateCaps` command in response to a `GetAccSampleRateCaps` command from an iPod.

The accessory returns the list of sample rates it supports with this command. The sample rates must be taken from the list of iPod supported sample rates shown in Table 6-170 (page 223). Any audio encoded at an unsupported sample rate is resampled internally by the iPod to conform to a supported sample rate. In general, audio quality will be better when no resampling is performed; therefore accessories are encouraged to support as many values listed in Table 6-170 (page 223) as possible.

Note: At a minimum, every accessory must support the sample rates 32 KHz, 44.1 KHz, and 48 KHz.

A `RetAccSampleRateCaps` command with sample rates not listed in Table 6-170 (page 223), or missing any of the required sample rates, is invalid. If the iPod receives such a command, it sends the accessory an `iPodAck` command with a negative acknowledgment as the command status.

Table 6-169 `RetAccSampleRateCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	2+4n	Length of packet payload
3	0x0A	Lingo ID: Digital audio lingo
4	0x03	Command ID: `RetAccSampleRateCaps`
5 ... (4+4n-1)	0x*NNNNNNNN*	A list of *n* sample rates (32-bit big-endian format) supported by the accessory. The sample rates must be taken from Table 6-170 (page 223).

Byte number	Value	Comment
5+4n	0x*NN*	Checksum

Table 6-170 Digital audio sample rates supported by iPods (in Hertz)

Decimal	Hexadecimal	Required/Optional
8000	0x00001F40	Optional
11025	0x00002B11	Optional
12000	0x00002EE0	Optional
16000	0x00003E80	Optional
22050	0x00005622	Optional
24000	0x00005DC0	Optional
32000	0x00007D00	Required
44100	0x0000AC44	Required
48000	0x0000BB80	Required

Command 0x04: NewiPodTrackInfo

Direction: iPod → Device

The iPod sends the `NewiPodTrackInfo` command before the first audio track begins playing. It sends the command again whenever it starts playing a track with different sample rate, sound check, or track volume parameters. In response to this command, accessories should prepare themselves to receive audio data with the new parameters.

The accessory must acknowledge this command, using the `AccAck` command, but the iPod does not wait for this acknowledgment before allowing digital audio to be transferred to the accessory.

The sample rate sent to the accessory is taken from the list of sample rates returned to the iPod by the `RetAccSampleRateCaps` command. If the accessory supports the sample rate of the current audio track, then it is sent as the current sample rate. If the accessory does not support the sample rate, the iPod resamples the audio data to a supported sample rate in real time and sends this new supported sample rate as the current sample rate.

The Sound Check value and track volume adjustment value are the corresponding values set by iTunes, rounded to the nearest integer. The values represent gain (in decibels) and may be positive or negative. Note that if the Sound Check option in the iPod is disabled, the `NewiPodTrackInfo` command always sends 0 as the new Sound Check value.

When the iPod sends a `NewiPodTrackInfo` command, it waits up to 500 ms for the `AccAck` command before timing out. If a timeout occurs or if the `AccAck` returns an error as the command status, the iPod sends the command again.

Important: The `NewiPodTrackInfo` command should not be used as a general mechanism to detect track changes. Use appropriate commands from the Display Remote or Extended Mode lingoes instead.

Table 6-171 `NewiPodTrackInfo` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x0E	Length of packet payload
3	0x0A	Lingo ID: Digital audio lingo
4	0x04	Command ID: `NewiPodTrackInfo`
5	0x*NN*	New sample rate 31:24 (32-bit big-endian format)
6	0x*NN*	New sample rate 23:16 (32-bit big-endian format)
7	0x*NN*	New sample rate 15:8 (32-bit big-endian format)
8	0x*NN*	New sample rate 7:0 (32-bit big-endian format)
9	*0xNN*	New Sound Check value 31:24 (32-bit big-endian format)
10	*0xNN*	New Sound Check value 23:16 (32-bit big-endian format)
11	*0xNN*	New Sound Check value 15:8 (32-bit big-endian format)
12	*0xNN*	New Sound Check value 7:0 (32-bit big-endian format)
13	*0xNN*	New track volume adjustment 31:24 (32-bit big-endian format)
14	*0xNN*	New track volume adjustment 23:16 (32-bit big-endian format)
15	*0xNN*	New track volume adjustment 15:8 (32-bit big-endian format)
16	*0xNN*	New track volume adjustment 7:0 (32-bit big-endian format)
17	0x*NN*	Checksum

Command 0x05: SetVideoDelay

Direction: Device → iPod

The device sends this command to inform the iPod of a new delay (in milliseconds) that must be applied to iPod video to synchronize it with the the audio for the currently playing video. See "Audio/Video Synchronization for Digital Audio" (page 214) . The iPod responds to `SetVideoDelay` with a an "iPodAck" (page 221) command.

Table 6-172 `SetVideoDelay` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART transport)
1	0x55	Start of packet
2	0x06	Length of packet payload
3	0x0A	Lingo ID: Digital audio lingo
4	0x05	Command ID: `SetVideoDelay`
5	0x*NN*	Video delay in milliseconds, 32-bit big-endian format (bits 31:24)
6	0x*NN*	Video delay in milliseconds, 32-bit big-endian format (bits 23:16)
7	0x*NN*	Video delay in milliseconds, 32-bit big-endian format (bits 15:8)
8	0x*NN*	Video delay in milliseconds, 32-bit big-endian format (bits 7:0)
9	0x*NN*	Checksum

Lingo 0x0C: Storage Lingo

The iAP Storage lingo, Lingo 0x0C, lets an accessory device store files on an attached iPod as if it were a hard drive. Use of the Storage lingo requires accessory authentication.

Command History of the Storage Lingo

Table 6-173 (page 225) shows the history of command changes in the Storage lingo:

Table 6-173 Storage lingo command history

Lingo version	Command changes	Features
1.01	Add: 0x00-0x02, 0x04, 0x07-08, 0x10-0x12	

Command Summary

Table 6-174 (page 226) lists the Storage lingo commands.

Table 6-174 Storage lingo commands

CmdID	Name	Direction	Payload:bytes
0x00	iPodACK	iPod to Dev	[ackStatus:1, cmdID:1, handle:1, (transactionID:2)]
0x01	GetiPodCaps	Dev to iPod	[None]
0x02	RetiPodCaps	iPod to Dev	[totalSpace:8, maxFileSize:4, maxWriteSize:2, Reserved:6, majorVersion:1, minorVersion:1]
0x03	Reserved		
0x04	RetiPodFileHandle	iPod to Dev	[handle:1]
0x05-0x06	Reserved		
0x07	WriteiPodFileData	Dev to iPod	[offset:4, handle:1, data:<var>]
0x08	CloseiPodFile	Dev to iPod	[handle:1]
0x09-0x0F	Reserved		
0x10	GetiPodFreeSpace	Dev to iPod	[None]
0x11	RetiPodFreeSpace	iPod to Dev	[freeSpace:8]
0x12	OpeniPodFeatureFile	Dev to iPod	[feature:8]
0x13-0xFF	Reserved		

The following is the typical sequence of commands used to store data on an iPod:

1. Send IdentifyDeviceLingoes (including the bit that identifies the Storage lingo) and pass authentication.
2. Send GetiPodCaps and receive RetiPodCaps, which returns the maximum write size.
3. Send OpeniPodFeatureFile and receive RetiPodFileHandle. The returned file handle points to a specific area in the iPod's file system. Currently the only accessible area is /iPod_Control/Device/Accessories/Tags/, used by the Radio Tagging System (see "iTunes Tagging" (page 239)).
4. Send one or more WriteiPodFileData commands with the data to be stored. Receive an iPodACK command for each one.
5. Send CloseiPodFile when finished writing all the data. Alternately, all files are automatically closed when the accessory detaches.

Command Details

This section describes the individual Storage lingo commands.

Command 0x00: iPodACK

Direction: iPod to Device

The iPod sends this command to acknowledge the receipt of a Storage lingo command from the accessory.

Table 6-175 `General iPodACK` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x05	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x00	Command ID: `iPodACK`
5	0x*NN*	Command result status. See Table 6-176 (page 227) .
6	0x*NN*	The ID for the command being acknowledged.
7	0x*NN*	The file handle for the command (0xFF if not applicable.)
8	0x*NN*	Checksum

Table 6-176 iPodACK responses

Value	Description
0x00	Success
0x01	N/A (reserved)
0x02	Command failed
0x03	Out of resources
0x04	Bad parameter
0x05	Unknown ID
0x06	N/A (reserved)
0x07	Not authenticated
0x08	Bad authentication version
0x09	N/A (reserved)
0x0A	N/A (reserved)

Value	Description
0x0B	N/A (reserved)
0x0C	N/A (reserved)
0x0D	Invalid file handle
0x0E-0xFF	Reserved

Command 0x01: GetiPodCaps

Direction: Device to iPod

The accessory asks the iPod to return its storage capabilities. The iPod replies with `RetiPodCaps`.

Table 6-177 `GetiPodCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x01	Command ID: `GetiPodCaps`
5	0xF1	Checksum

Command 0x02: RetiPodCaps

Direction: iPod to Device

The iPod tells the accessory about the iPod's storage capabilities:

- `totalSpace` is the amount of storage on the iPod in bytes, including space currently in use.
- `maxFileSize` is the largest possible size, in bytes, of any file on the iPod.
- `maxWriteSize` is the largest amount of data, in bytes, that can be written to the iPod in a single `WriteiPodFileData` command.
- `majorVersion` and `minorVersion` are the version number of the Storage lingo protocol implemented by the device. The current version is 1.1.

Table 6-178 `RetiPodCaps` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)

Byte number	Value	Comment
1	0x55	Start of packet (SOP)
2	0x18	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x02	Command ID: `RetiPodCaps`
5	0x*NN*	`totalSpace` (bits 63:56). The total amount of storage space on the iPod in bytes, including space currently in use.
6	0x*NN*	`totalSpace` (bits 55:48)
7	0x*NN*	`totalSpace` (bits 47:40)
8	0x*NN*	`totalSpace` (bits 39:32)
9	0x*NN*	`totalSpace` (bits 31:24)
10	0x*NN*	`totalSpace` (bits 23:16)
11	0x*NN*	`totalSpace` (bits 15:8)
12	0x*NN*	`totalSpace` (bits 7:0)
13	0x*NN*	`maxFileSize` (bits 31:24). The size in bytes of the largest possible file on the iPod.
14	0x*NN*	`maxFileSize` (bits 23:16)
15	0x*NN*	`maxFileSize` (bits 15:8)
16	0x*NN*	`maxFileSize` (bits 7:0)
17	0x*NN*	`maxWriteSize` (bits 15:8). The size in bytes of the largest possible amount of data than can be sent to the iPod with `WriteiPodFile`.
18	0x*NN*	`maxWriteSize` (bits 7:0)
19	0x*NN*	Reserved.
20	0x*NN*	Reserved.
21	0x*NN*	Reserved.
22	0x*NN*	Reserved.
23	0x*NN*	Reserved.
24	0x*NN*	Reserved.
25	0x*NN*	`majorVersion` (1 byte). The major version number.
26	0x*NN*	`minorVersion` (1 byte). The minor version number.

Byte number	Value	Comment
27	0x*NN*	Checksum

Command 0x04: RetiPodFileHandle

Direction: iPod to Device

The iPod returns a unique 8-bit handle to identify the file.

The value of `handle` is a session-specific identifier that represents a file. This is similar to a Unix file descriptor. The handle is valid until either the accessory is detached or the accessory sends a `CloseiPodFile` command with this handle.

Table 6-179 `RetiPodFileHandle` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x04	Command ID: `RetiPodFileHandle`
5	0x*NN*	`handle` (1 byte)
6	0x*NN*	Checksum

Command 0x07: WriteiPodFileData

Direction: Device to iPod

The accessory writes a block of data to a file starting at the offset passed with this command. The file must have been previously opened for writing; failure is reported as a bad parameter (see Table 6-176 (page 227)). If the file was not previously opened for writing, or the handle is invalid (invalid handle error), or the `writeSize` exceeds the iPod's capabilities (bad parameter error), then the operation will fail. If the caller attempts to write too much data, the state of the file will be undefined; some or none of the data may have been added to the file.

All data must be written sequentially, but write actions may occur across multiple `WriteiPodFileData` commands. The amount of data transferred must also be within the bounds set by the iPod's capabilities. If any of these criteria are not met, the iPod will return an `iPodACK` command with a failure message.

When a `WriteiPodFileData` command fails, the state of the file is left unknown. The accessory must close the file and then create a new file to write the data.

`WriteiPodFileData` passes the following parameters:

- offset is the offset (in bytes) into the file at which to begin writing.
- handle is a unique file identifier.
- data is the data to be written to the file.

Table 6-180 WriteiPodFileData packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x*NN*	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x07	Command ID: WriteiPodFileData
5	0x*NN*	offset (bits 31:24). The offset into the file at which to begin writing, in bytes.
6	0x*NN*	offset (bits 23:16)
7	0x*NN*	offset (bits 15:8)
8	0x*NN*	offset (bits 7:0)
9	0x*NN*	handle (1 byte).
10...N	0x*NN*	data (variable length). The file data.
(last byte)	0x*NN*	Checksum

Command 0x08: CloseiPodFile

Direction: Device to iPod

This command closes a file and releases its handle. The handle is invalid for further use after this call, and the iPod may assign the handle later to represent a different file. An iPodACK command is sent on success or failure. Parameter handle is a unique file identifier.

Table 6-181 CloseiPodFile packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x08	Command ID: CloseiPodFile

Byte number	Value	Comment
5	0x*NN*	`handle` (1 byte).
6	0x*NN*	Checksum

Command 0x10: GetiPodFreeSpace

Direction: Device to iPod

The accessory asks the iPod to return the amount of free space on its storage system. The iPod replies with `RetiPodFreeSpace`.

Table 6-182 `GetiPodFreeSpace` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x02	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x10	Command ID: `GetiPodFreeSpace`
5	0xE2	Checksum

Command 0x11: RetiPodFreeSpace

Direction: iPod to Device

The iPod tells the accessory the current amount of free space (in bytes) in its storage system.

Table 6-183 `RetiPodFreeSpace` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x0A	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x11	Command ID: `RetiPodFreeSpace`
5	0x*NN*	`freeSpace` (bits 63:56). The amount of the iPod's free space.
6	0x*NN*	`freeSpace` (bits 55:48)

Byte number	Value	Comment
7	0x*NN*	`freeSpace` (bits 47:40)
8	0x*NN*	`freeSpace` (bits 39:32)
9	0x*NN*	`freeSpace` (bits 31:24)
10	0x*NN*	`freeSpace` (bits 23:16)
11	0x*NN*	`freeSpace` (bits 15:8)
12	0x*NN*	`freeSpace` (bits 7:0)
13	0x*NN*	Checksum

Command 0x12: OpeniPodFeatureFile

Direction: Device to iPod

The device requests a file handle to write data for a specified feature. Currently the only valid value for the `feature` field is 1, representing the iTunes tagging feature (see "iTunes Tagging" (page 239)).

Table 6-184 `OpeniPodFeatureFile` packet

Byte number	Value	Comment
0	0xFF	Sync byte (required only for UART serial)
1	0x55	Start of packet (SOP)
2	0x03	Length of packet
3	0x0C	Lingo ID: Storage lingo
4	0x12	Command ID: `OpeniPodFeatureFile`
5	0x01	`feature` (1 byte)
6	0xDE	Checksum

iPod Power States and Accessory Power

To optimize music playback time and best utilize the internal battery and external power sources (such as a powered dock, power brick, computer, or other powered accessory), iPod models support several different power states. These power states affect accessories, particularly those that are powered by the iPod. The iPod can transition between power states due to inactivity of the iPod UI, accessory actions, or internal iPod battery-conservation actions.

iPod Power States

There are three iPod power states: Power On, Light Sleep, and Deep Sleep/Hibernate. The Power On state consumes the most power, because the display backlight may be on and tracks may be playing. Light Sleep consumes less power, although parts of the iPod are still powered in order to respond to iAP commands over the UART serial port link. Deep Sleep/Hibernate is the lowest power state and is used to preserve internal batter power for extended periods of inactivity, such as days or weeks. Table A-1 (page 235) describes these power states.

Table A-1 iPod power states

	Power On	Light Sleep	Deep Sleep/ Hibernate
Display State	ON	OFF	OFF
Attributes	The iPod UI may be active, allowing users to interface with the menus, listen to music, view images, and so forth. If the UI is inactive—that is, if no track is playing and there is no user input for 2 minutes—the iPod displays a large battery icon (either "Charged" or "Charging").	The iPod UI is inactive—no track is playing—and the UI state is preserved. The iPod can respond immediately to front panel buttons or to iAP simple remote buttons. Attaching some accessories may also wake the iPod.	The iPod is inactive (no track is playing). The iPod does not respond to iAP packets over any transports.

	Power On	Light Sleep	Deep Sleep/ Hibernate
Accessory Power	ON Typical accessory power consumption should be below 5 mA. Accessories using the intermittent high power option can consume up to a total of 100 mA while music is playing or during other high power modes, such as voice recording.	ON Maximum accessory power consumption must be below 5 mA. Simple remote accessories should consume less than 10 µA. High power mode is disabled.	OFF
iAP Transports	All transports are usable.	UART serial only. The iPod will not transition into this state if the USB port link is active.	None.
Transitions	To the Light Sleep state, when any of the following occurs: ■ The simple remote "Off" button is pressed or released. ■ The play/pause button is pressed or held for 2 seconds. ■ The iPod is idle (no track is playing) for 2 minutes. ■ The iPod is displaying the battery icon and the external power source is removed. The iPod will not transition out of the Power On state when an external power source is connected or a track is playing.	To the Power On state, when any of the following occurs: ■ An external power source is connected. ■ The user presses a button on the front panel of the iPod. ■ A Simple Remote lingo command other than "Off" is sent. To the Deep Sleep/Hibernate state when the appropriate amount of time expires. See Table 2-2 (page 28) for the duration of the Light Sleep state before the transition to the Deep Sleep/Hibernate state.	To the Power On state, when either of the following happens: ■ An external power source is connected. ■ The Menu or Select button on the front panel of the iPod are pressed.

Deep Sleep and Hibernate are grouped together into a single state because their behaviors are similar. However, they are mutually exclusive states. An iPod supports either a Deep Sleep state or a Hibernate state, but not both. The primary difference between the Deep Sleep and Hibernate states is the iPod UI context when it transitions back to the Power On state. An iPod in the Deep Sleep state loses all of its previous menu selections and playback context. When the iPod transitions to the Power On state, the UI displays the topmost menu.

In contrast, an iPod in the Hibernate state retains all of its menu selections and playback context. When the iPod transitions back to the Power On state, the previous menu and track selection are restored, so the user can resume their currently playing playlist and track position, exactly where they left off. See Table 3-2 (page 45) for more information about which iPod models support the Deep Sleep or Hibernate states.

WARNING: The trend in iPod design is for shorter sleep times before the iPod hibernates. In the 2G iPod nano, iPod classic, and 3G nano, the time is 30 minutes; the iPod touch does not have a light sleep stage. Accessories should not be designed to draw power from iPods in light sleep, because such power will be shut off when the iPod hibernates.

Device Power Usage

The power supply to serial devices is a single 2.85 V to 3.465 V supply. The iPod always provides accessory power when it is in the Power On or Light Sleep states. Accessory power is switched off for two seconds during the iPod bootstrap process and when the iPod can no longer supply it due to low battery conditions.

Simple remotes must use less than 5 mA at all times. Simple remotes are very likely to be left on during Light Sleep, in order to allow waking from the remote. During Light Sleep, power usage is very critical and thus power usage by an inactive remote should be less than 10 µA.

No single device or combination of devices may use more than 100 mA at any time.

iTunes Tagging

This appendix explains the iTunes tagging feature and specifies the requirements for an HD radio accessory to support it.

The iTunes Tagging Experience

To experience the capabilities of the iTunes tagging feature, an iPod user typically goes through the following steps:

1. While listening to broadcast music on an HD radio accessory, the user hears a song and decides to tag it for future review or purchase by pressing a special button or other user interface element.
2. The radio accessory, equipped with a tag button or equivalent software action, stores metadata for the currently playing song. The user tags the song without needing to know anything about it or its originating station. The song information is stored in the HD radio accessory until an iPod is connected to it.
3. When an iPod is connected to it, the radio transfers the tagged song information to the iPod. The iPod stores the data until the iPod is synched with iTunes.
4. When the iPod is connected to the user's computer, iTunes imports the tag data, analyzes it, and presents it to the user in the form of a tagged playlist. iTunes displays the song title, artist, album and other information for each tagged song.
5. Using iTunes, the user may save, review or delete any item in the tagged playlist and may easily purchase any of the listed songs through the iTunes store.

Tagging Feature Components

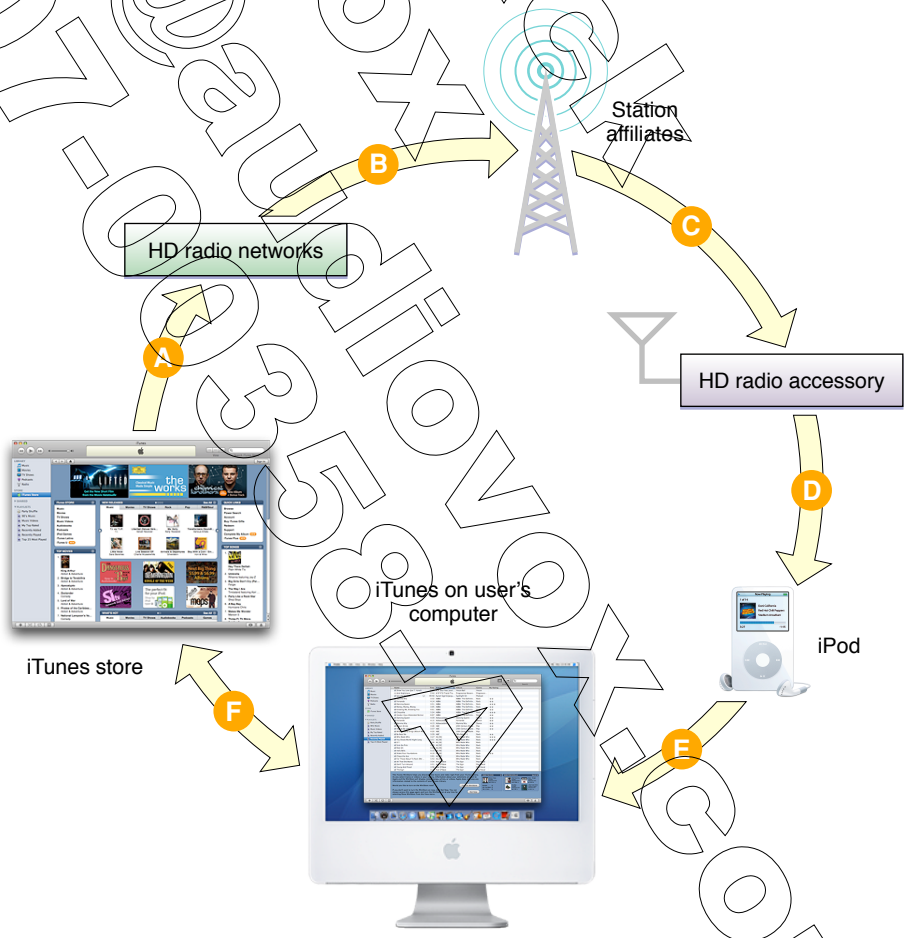
To implement the user experience described in the previous section, the iTunes tagging feature includes the following major components:

- The **iTunes store** creates and publishes unique identifiers for media available in its catalog, and allows users of accessories that support radio tagging for iTunes to purchase tagged songs.

- **HD radio networks and station affiliates** receive iTunes store data and broadcast it along with program music and other content.
- **HD radio accessories for the iPod** are designed to enable the tagging feature. These radio receivers may be portable, used in the home or in a car.
- An **iPod** is a media player used to transfer tags from iPod accessories to iTunes.
- The **iTunes application** is a desktop media player for managing media content, including newly-discovered music via the iTunes tagging feature.

The data flows among the components of the iTunes tagging feature listed above are diagrammed in Figure B-1 (page 240) .

Figure B-1 iTunes tagging feature data flows



Data flow	Description
A	An enterprise partner feed from Apple contains `iTunesSongIDs`, track names, artists, and album names for the contents of the iTunes store.
B	Network data is sent to the HD station affiliates, including Apple-supplied data.

Data flow	Description
C	Affiliates broadcast program service data (PSD) with each song, containing the song's `iTunesSongID`, track name, artist, album and other metadata.
D	The HD radio accessory writes XML tag data to a file on an attached iPod, using the iAP Storage lingo; the iPod signs the file.
E	iTunes verifies and uploads the iPod's tag files and presents a tagged music playlist to the user.
F	The user clicks a "Buy" button next to the tagged music and downloads the music from the iTunes Store.

Radio Accessory Requirements

HD radio accessories that support the iTunes tagging feature described in this specification must perform the following actions:

- Authenticate themselves to the attached iPod. This requires that the accessory contain an authentication coprocessor, Version 2.0A or 2.0B, supplied by Apple. For technical details, see Apple's *iPod Authentication Coprocessor 1.0 and 2.0A Specification* and *iPod Authentication Coprocessor 2.0B Specification*.
- Conform to the user interface requirements described in "Accessory User Interface" (page 247) .
- Generate tag data from the HD radio broadcast's program service data (PSD).
- Provide static storage capacity for 50 tags, and provide enough RAM to cache two tags in cases of tag ambiguity (see "Resolving Tag Ambiguity" (page 246)). Accessories should store tags in a data structure and only generate XML when writing files to the iPod. Tag data may be stored as an iBiquity purchase token, as defined by iBiquity Digital Corporation, or in any other format convenient to the accessory.
- Generate files from the tag data and write these files to the attached iPod. The files must be XML-formatted **plists,** as specified in "Tag Data Writing Process" (page 241) . The required plist fields are specified in "Data Transfer to the iPod" (page 243) , and the iAP commands used to write a file to the iPod are defined in "Lingo 0x0C: Storage Lingo" (page 225) .
- Design the accessory so that the plist files that it generates pass the verification test performed by Apple's plist verifier command-line utility.

Note: Apple provides Made For iPod developers with a plist verifier tool.

Tag Data Writing Process

Each tagged track file written to an iPod consists of an extensible XML dictionary of key-value pairs, formatted as a Mac OS X Core Foundation property list (**plist**). This format is widely used in Mac OS X and can be easily generated and parsed on other platforms. The format is documented at

developer.apple.com/documentation/CoreFoundation/Conceptual/CFPropertyLists/index.html. The plist document type definition (DTD) is available at www.apple.com/DTDs/PropertyList-1.0.dtd. Listings of typical of plist files can be found in "Sample Tag Files" (page 248) .

Note: Mac OS X plist files are UTF-8 encoded; the PSD data sent by HD radio broadcasters is ISO-8859-1 encoded. The radio accessory must convert the IS0-8859-1 data to UTF-8 before writing the plist file to the iPod. If the plist file contains ISO-8859-1 characters in the range 0x80 to 0xFF, iTunes cannot parse the file properly and the tag will be lost.

After authenticating itself to the attached iPod, an accessory typically uses the following command sequence to write a plist file to the iPod. For details of these commands, see "Lingo 0x0C: Storage Lingo" (page 225) .

Radio accessory	Attached iPod
`GetiPodCaps`	
	`RetiPodCaps`
`GetiPodFreeSpace`	
	`RetiPodFreeSpace`
`OpeniPodFeatureFile`	
	`RetiPodFileHandle`
`WriteiPodFileData` (as many times as needed)	
	`iPodACK` for each `WriteiPodFileData` command
`CloseiPodFile`	
	`iPodACK` of `CloseiPodFileData`

The accessory starts by sending a `GetiPodCaps` command to retrieve the iPod's storage lingo capabilities. The iPod responds with the `RetiPodCaps` command, which contains the following data:

- `totalSpace`: the amount of free storage on the iPod.
- `maxFileSize`: the largest possible size of a file on the iPod.
- `maxWriteSize`: the largest amount of data that can be written to the iPod in a single `WriteiPodFileData` command.
- `majorVersion` and `minorVersion`: the version number of the storage lingo protocol (currently 1.1).

Accessories are required to honor the iPod's `maxWriteSize` limitation; they should never send a `WriteiPodFileData` command with a data payload larger than the iPod's `maxWriteSize` value.

The accessory should also check the iPod's free space before creating a file, to ensure that there is enough free space to write the file. A tag takes approximately 1 KB of data space, so the accessory should verify that the iPod has 1 KB of free space for each tag is intends to write. The accessory should warn the user through its user interface if the iPod does not have enough free space; see "Accessory User Interface" (page 247) for details.

To create a file on the iPod, the accessory sends an `OpeniPodFeatureFile` command. This command returns a file handle that is used to write data and to close the file. The accessory must send a `WriteiPodFileData` command to write data to the open file. The size of the `WriteiPodFileData` payload is determined by the iPod's `maxWriteSize`; it may take several `WriteiPodFileData` commands to write an entire file. The iPod responds to each `WriteiPodFileData` command with an `iPodACK` command containing the status of the last write. Accessories must verify that each individual write succeeded before sending the next `WriteiPodFileData` command.

Only one tagging file may be opened at a time. Each time a tagging file is opened, a new file is created in `/iPod_Control/DeviceXAccessories/Tags/` on the iPod (this location may change in future releases). You can look at the files in this directory to confirm that the data you wrote using `WriteiPodFileData` was transferred correctly.

The accessory should close the file using the `CloseiPodFile` command as soon as all the plist data has been written to the iPod. The accessory should verify that the iPod closed the file properly by checking the status of the `iPodACK` command sent in response to the `CloseiPodFile` command. Accessories should never delete tag data stored locally until the `CloseiPodFile` acknowledgment status has been verified.

Data Transfer to the iPod

Table B-1 (page 243) shows the plist fields an accessory writes to an iPod. Some fields are required; iTunes will not create a tagged playlist without these fields. The minimum set of fields needed to create an iTunes playlist are `MajorVersion`, `MinorVersion`, `ManufacturerID`, `ManufacturerName`, `DeviceName`, `MarkedTracks`, `Name`, and `Artist`. However, the accessory should populate every plist field for which it receives data from the HD radio broadcast, including program service data (PSD) and station information services (SIS).

Table B-1 Plist fields written to the iPod

Name	Type	Description	Required	Source
`MajorVersion`	integer	The major revision level of the file format. A change to this number indicates changes or additions that break compatibility with all prior versions.	yes	Accessory
`MinorVersion`	integer	The minor revision level of the file format. A change to this number indicates changes or additions that preserve compatibility with prior versions that have the same major revision level.	yes	Accessory
`ManufacturerID`	integer	Manufacturer IDs are assigned by Apple's Made For iPod program (see note, below).	yes	Accessory

Name	Type	Description	Required	Source
`ManufacturerName`	string	The manufacturer-assigned user-visible name for the manufacturer.	yes	Accessory
`DeviceName`	string	The manufacturer-assigned user-visible name for the device.	yes	Accessory
`MarkedTracks`	array	An array of one or more dictionaries, specifying the data for each tagged track.	yes	Accessory
`AmbiguousTag`	Integer	A field used to mark ambiguous tracks.	no	Accessory
`ButtonPressed`	Integer	This field is used to designate which twin in an ambiguous pair was active when the tag button was pressed.	no	Accessory
`Name`	string	The name of the track.	yes	HD PSD
`Artist`	string	The name of the artist.	yes	HD PSD
`Album`	string	The name of the album.	no	HD PSD
`iTunesSongID`	integer	The iTunes song identifier.	no	HD PSD
`iTunesStorefrontID`	integer	The iTunes storefront identifier.	no	HD PSD
`StationFrequency`	string	The station's frequency.	no	HD HOST
`StationCallLetters`	string	The call letters for the station.	no	HD SIS
`StationURL`	string	The URL for the station.	no	HD PSD
`Genre`	string	The program type.	no	HD PSD
`TimeStamp`	date	The date and time at which the track was tagged.	no	HD SIS
`TimeLockStatus`	integer	The station's time lock status.	no	HD SIS
`StreamID`	integer	The station's stream identifier.	no	HD HOST
`iTunesAffiliateID`	integer	The iTunes affiliate ID.	no	HD PSD
`UnknownData`	data	A field used to pass unknown UFID data type IDs to iTunes (see "The UnknownData Field" (page 245)).	no	HD PSD

Note: To obtain a `manufacturerID`, a partner must submit a product plan to Apple's Made For iPod program. Once the product plan is approved, Apple will respond with a `manufacturerID` assigned to that plan.

The radio accessory normally writes a tag to the iPod each time the user presses the tag button. However, the tag should not be written if tag ambiguity exists and has not been resolved (see "Resolving Tag Ambiguity" (page 246)). The opening, writing, and closing of the file should happen in one continuous action to insure that no tag information is lost. Accessories should not keep files open any longer than absolutely necessary. Do not append tags to open files; write a new file to the iPod each time the user presses the tag button.

An accessory should never write a tag more than once. If the user presses the tag button multiple times during the same song, the accessory should write the tag to the iPod after the first button press and then filter out redundant tags by setting a byte to indicate that the broadcast tag data has been written to the iPod.

The accessory should store tags internally when no iPod is connected. If the accessory runs out of storage space, it must prompt the user to connect an iPod. When tags are stored locally, the accessory should write all tag data to a single file the next time an iPod is docked. The `MajorVersion`, `MinorVersion`, `ManufacturerName`, `ManufacturerID`, and `MarkedTracks` fields need to be written only once per file when writing multiple tags to a single file. The rest of the tag data is written in a tag array with each tag designated by the `<dict>` key. "Sample Tag Files" (page 248) shows an example of multiple tags in a single file.

Before it deletes any tag stored locally, the accessory must verify that the iPod has sent an `iPodACK` command in response to each `WriteiPodFileData` command and the `CloseiPodFile` command. The accessory must notify the user that the tags were successfully written to the iPod.

Sending a Podcast Feed URL

Broadcasting a UFID data ID type of 0x06 (see Table B-3 (page 246)) indicates that the contents of the `stationURL` field is the URL of a podcast feed (a `PodcastFeedURL`), not a station URL.

The UnknownData Field

Accessories can support features announced after their product release by supporting the `UnknownData` plist field. This field is used to pass unrecognized data types in the broadcast PSD's Unique File Identifier (UFID) Data field to iTunes. iTunes parses the data in this field and handles it appropriately. HD radio accessories should use this field when encountering an unrecognized UFID data ID type while parsing the HD broadcast PSD's UFID data field. The data written to the `UnknownData` plist field should be formatted in `base64` format. Multiple data blocks may be written, each formatted as shown in Table B-2 (page 246) .

Table B-2 UnknownData block format

Bytes	Contents
0-1	ID type (see Table B-3 (page 246))
2-3	Data length (*NN* + 1)
4-*NN*	Data payload

The currently known UFID data ID types are listed in Table B-3 (page 246) .

Table B-3 UFID data ID types

ID type	Description
0x0001	`iTunesSongID`
0x0002-0x0003	Reserved
0x0004	`iTunesAffiliateID`
0x0005	`iTunesStorefrontID`
0x0006	The contents of the `StationURL` field is a `PodcastFeedURL`.

Resolving Tag Ambiguity

A tag is deemed ambiguous if the tag button is pressed within 10 seconds before or after a program service data (PSD) change. This ambiguity period exists in HD radio because PSD changes do not occur exactly on song boundaries, but within 10 seconds before or after that boundary.

The accessory must flag ambiguous tags using the `ambiguousTag` plist field. The iTunes application will present a user interface dialog to resolve the ambiguity once the tags have been imported. Accessories should not try to resolve tag ambiguity through their own user interface. Accessories must identify each twin of the ambiguous pair and save this information as a part of the tag data in RAM. Accessories must also identify which twin was active at the time of the tag button press by setting the `ButtonPressed` plist field.

To support tag ambiguity, accessories must store both the current and previous sets of tag data in RAM and update both as PSD changes occur. A timestamp or timer can be used to mark when either the PSD changes or the tag button is pressed. The previous and current tags should be stored (when no iPod is connected) or written to the iPod with the `ambiguousTag` plist field set to 1 if these two events occur within 10 seconds of each other.

If the tag button is pressed within 10 seconds **before** a PSD change:

- The accessory timestamps the button press, or starts a 10 second timer when the tag button is pressed, and sets the `ButtonPressed` byte in the current tag data.
- If a PSD change occurs within the 10 seconds, the tag data in the current slot is moved to the previous slot and the broadcast tag data fills the current slot. The accessory sets the `ambiguousTag` fields for both tags to 1 and writes both tags to the iPod in a single file.

If the tag button is pressed within 10 seconds **after** a PSD change:

- The accessory timestamps the PSD change event, or starts a 10 second timer when the PSD change occurs, and updates the tag data in the current and previous slots.
- If the tag button is pressed within 10 seconds of the PSD change, the accessory sets the `ButtonPressed` byte in the current tag data, sets the `ambiguousTag` fields for both tags to 1, and writes both tags to the iPod in a single file.

Note: Accessories must resolve tag ambiguity (if it exists) before saving tags or writing them to the iPod. This adds a 10 second delay to the file writing process.

Accessory User Interface

Table B-4 (page 247) lists the user interface elements that must be used for the tagging feature in a radio accessory. The table shows three sections: Required UI, Optional UI, and UI not allowed. Table B-5 (page 248) lists user interface messages that may appear on the accessory's display.

Table B-4 tagging feature user interface implementation

Action		Implementation	
		Speaker	**Head unit**
Required UI	Indicate tag available	Button (LED) illuminates	Button or logo/type appears
		Logo/type appears (LCD)	Button color or logo/type changes
	Indicate tag captured	Button LED blinks	Button blinks
		Button LED changes color	Button changes color
		Logo/type blinks (LCD)	Graphic
		Message on LCD	Message on screen
		Audible feedback	Audible feedback
	Indicate accessory memory full	Message (LCD)	Message
		Audible feedback	Audible feedback
			Graphic
	Indicate iPod memory full	Message (LCD)	Message
		Audible feedback	Audible feedback

Action		Implementation	
		Speaker	Head unit
Optional UI	Indicate write to iPod	Button LED blinks twice	Button blinks twice
		Message on LCD	Message on screen
		Audible feedback	Audible feedback
	Indicate tag not captured	LED color changes	Button color changes
		Message on LCD	Message on screen
		Audible feedback	Audible feedback
	Tags remaining	Message on LCD	Message on screen
UI not allowed	Resolving tag ambiguity		
	Choosing whether or not to write tags to the iPod		

Table B-5 tagging feature UI text messages

Condition	Message
Tag captured	“Tag stored"
	“Tag stored. XX remaining.”
Accessory memory full	“Memory full. Connect iPod.”
	“Connect iPod to transfer tags.”
iPod memory full	“iPod full. Tags cannot be stored.”
Write to iPod	“Tags transferred to iPod.”
	“Tags saved to iPod. XX remaining.”

Note: These message texts are preliminary and are provided for guidance only. The specific texts to be displayed are expected to change as the tagging feature user interface is further defined.

Sample Tag Files

The following is a sample plist-formatted tag file showing a header and one stored tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
```

```
<dict>
    <key>MajorVersion</key>
    <integer>1</integer>
    <key>MinorVersion</key>
    <integer>0</integer>
    <key>ManufacturerID</key>
    <integer>17</integer>
    <key>ManufacturerName</key>
    <string>Acme</string>
    <key>DeviceName</key>
    <string>AC-HDR100</string>
    <key>MarkedTracks</key>
    <array>
        <dict>
            <key>Name</key>
            <string>Times Like These</string>
            <key>Artist</key>
            <string>Foo Fighters</string>
            <key>Album</key>
            <string>One by One</string>
            <key>iTunesSongID</key>
            <integer>6906304</integer>
            <key>iTunesStorefrontID</key>
            <integer>143441</integer>
            <key>StationFrequency</key>
            <string>104.9</string>
            <key>StationCallLetters</key>
            <string>KCNL</string>
            <key>StationURL</key>
            <string>http://www.channel1049.com</string>
            <key>Genre</key>
            <string>Alternative</string>
            <key>TimeStamp</key>
            <date>2007-04-16T00:35:42Z</date>
            <key>TimeLockStatus</key>
            <integer>1</integer>
            <key>StreamID</key>
            <integer>1</integer>
            <key>StationGroup</key>
            <string>ClearChannel</string>
            <key>iTunesAffiliateID</key>
            <integer>0</integer>
            <key>UnknownData</key>
            <data>AAoAAf8ACwACvu8ADAAEESIzRAANAAiqu8zd7v8AEQAOAAA=</data>
        </dict>
    </array>
</dict>
</plist>
```

The following is a sample plist-formatted tag file showing how ambiguous tags are saved to an iPod:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>MajorVersion</key>
    <integer>1</integer>
```

```
    <key>MinorVersion</key>
    <integer>0</integer>
    <key>ManufacturerID</key>
    <integer>17</integer>
    <key>ManufacturerName</key>
    <string>Acme</string>
    <key>DeviceName</key>
    <string>AC-HDR100</string>
    <key>MarkedTracks</key>
    <array>
        <dict>
            <key>AmbiguousTag</key>
            <integer>1</integer>
            <key>ButtonPressed</key>
            <integer>1</integer>
            <key>Name</key>
            <string>Times Like These</string>
            <key>Artist</key>
            <string>Foo Fighters</string>
            <key>Album</key>
            <string>One by One</string>
            <key>iTunesSongID</key>
            <integer>6906304</integer>
            <key>iTunesStorefrontID</key>
            <integer>143441</integer>
            <key>StationFrequency</key>
            <string>104.9</string>
            <key>StationCallLetters</key>
            <string>KCNL</string>
            <key>StationURL</key>
            <string>http://www.channel1049.com</string>
            <key>Genre</key>
            <string>Alternative</string>
            <key>TimeStamp</key>
            <date>2007-04-16T00:35:42Z</date>
            <key>TimeLockStatus</key>
            <integer>1</integer>
            <key>StreamID</key>
            <integer>1</integer>
            <key>StationGroup</key>
            <string>ClearChannel</string>
            <key>iTunesAffiliateID</key>
            <integer>0</integer>
        </dict>
        <dict>
            <key>AmbiguousTag</key>
            <integer>1</integer>
            <key>ButtonPressed</key>
            <integer>0</integer>
            <key>Name</key>
            <string>Vertigo</string>
            <key>Artist</key>
            <string>U2</string>
            <key>Album</key>
            <string>How to Dismantle an Atomic Bomb</string>
            <key>iTunesSongID</key>
            <integer>29600235</integer>
            <key>iTunesStorefrontID</key>
```

```
            <integer>143441</integer>
            <key>StationFrequency</key>
            <string>104.9</string>
            <key>StationCallLetters</key>
            <string>KCNL</string>
            <key>StationURL</key>
            <string>http://www.channel1049.com</string>
            <key>Genre</key>
            <string>Rock</string>
            <key>TimeStamp</key>
            <date>2007-04-16T00:35:45Z</date>
            <key>TimeLockStatus</key>
            <integer>1</integer>
            <key>StreamID</key>
            <integer>1</integer>
            <key>StationGroup</key>
            <string>ClearChannel</string>
            <key>iTunesAffiliateID</key>
            <integer>0</integer>
        </dict>
    </array>
</dict>
</plist>
```

The following is a sample plist-formatted tag file showing how an accessory that has multiple tags stored locally writes those tags to an iPod:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>MajorVersion</key>
    <integer>1</integer>
    <key>MinorVersion</key>
    <integer>0</integer>
    <key>ManufacturerID</key>
    <integer>17</integer>
    <key>ManufacturerName</key>
    <string>Acme</string>
    <key>DeviceName</key>
    <string>AC-HDR100</string>
    <key>MarkedTracks</key>
    <array>
        <dict>
            <key>Name</key>
            <string>Times Like These</string>
            <key>Artist</key>
            <string>Foo Fighters</string>
            <key>Album</key>
            <string>One by One</string>
            <key>iTunesSongID</key>
            <integer>6906304</integer>
            <key>iTunesStorefrontID</key>
            <integer>143441</integer>
            <key>StationFrequency</key>
            <string>104.9</string>
            <key>StationCallLetters</key>
```

```
        <string>KCNL</string>
        <key>StationURL</key>
        <string>http://www.channel1049.com</string>
        <key>Genre</key>
        <string>Alternative</string>
        <key>TimeStamp</key>
        <date>2007-04-16T00:35:42Z</date>
        <key>TimeLockStatus</key>
        <integer>1</integer>
        <key>StreamID</key>
        <integer>1</integer>
        <key>iTunesAffiliateID</key>
        <integer>0</integer>
    </dict>
    <dict>
        <key>Name</key>
        <string>Vertigo</string>
        <key>Artist</key>
        <string>U2</string>
        <key>Album</key>
        <string>How to Dismantle an Atomic Bomb</string>
        <key>iTunesSongID</key>
        <integer>29600235</integer>
        <key>iTunesStorefrontID</key>
        <integer>143441</integer>
        <key>StationFrequency</key>
        <string>104.9</string>
        <key>StationCallLetters</key>
        <string>KCNL</string>
        <key>StationURL</key>
        <string>http://www.channel1049.com</string>
        <key>Genre</key>
        <string>Rock</string>
        <key>TimeStamp</key>
        <date>2007-04-16T00:43:45Z</date>
        <key>TimeLockStatus</key>
        <integer>1</integer>
        <key>StreamID</key>
        <integer>1</integer>
        <key>iTunesAffiliateID</key>
        <integer>0</integer>
    </dict>
    <dict>
        <key>Name</key>
        <string>Ball and Chain</string>
        <key>Artist</key>
        <string>Social Distortion</string>
        <key>Album</key>
        <string>Social Distortion</string>
        <key>iTunesSongID</key>
        <integer>197986000</integer>
        <key>iTunesStorefrontID</key>
        <integer>143441</integer>
        <key>StationFrequency</key>
        <string>104.9</string>
        <key>StationCallLetters</key>
        <string>KCNL</string>
        <key>StationURL</key>
```

```
            <string>http://www.channel1049.com</string>
            <key>Genre</key>
            <string>Rock</string>
            <key>TimeStamp</key>
            <date>2007-04-16T00:48:45Z</date>
            <key>TimeLockStatus</key>
            <integer>1</integer>
            <key>StreamID</key>
            <integer>1</integer>
            <key>iTunesAffiliateID</key>
            <integer>0</integer>
        </dict>
    </array>
</dict>
</plist>
```

Interfacing With the 3G iPod

The 3G iPod is the oldest model iPod with a 30-pin connector. Because later models also have 30-pin connectors, users may assume that an accessory designed for later iPods will work in the same way when a 3G iPod is plugged into it. However, there are functional differences. This appendix summarizes some of the model-specific design issues that must be addressed if an accessory device for current iPod models is to provide 3G iPod support.

Accessory Detection

Serial accessories designed to work with the 3G iPod must use a 549 kΩ R_{ID} resistor for accessory detection (see "Accessory Detect and Identify" (page 32)). With any other resistor value, the 3G iPod will not correctly detect when the accessory has been detached. If the accessory uses Extended Interface Mode, this can cause the 3G iPod to remain locked in Extended mode until a new serial accessory is attached to it.

Connector Usage

The 3G iPod serial port is shared between the 30-pin connector and the 9-pin Audio/Remote connector. Attaching an accessory to the 30-pin connector disables any accessory that may be plugged into the 9-pin connector.

Connecting both FireWire power and USB power simultaneously will put the 3G iPod into an invalid state.

The 3G iPod's serial communication rate is limited to 19200 baud, and it does not support autobaud on framing errors (see "UART Serial Port Link" (page 47)).

The 3G iPod does not automatically awake from light sleep when it receives a packet over the UART serial port link (see "iPod Power States" (page 235)).

Protocol Compatibility

3G iPod firmware version 2.0.0 supports only the small packet format with a maximum payload of 127 bytes; it does not support commands that require a larger payload. See Table 5-3 (page 63).

3G iPod firmware versions 2.1.0 and later include limited support for the RF Transmitter lingo (Lingo 0x05), but only through the 9-pin audio/remote connector. They do not support this lingo through the 30-pin connector.

Communication and Commands

The 3G iPod does not support the `IdentifyDeviceLingoes` command; it supports only the older `RequestIdentify` and `Identify` commands. In addition, the 3G iPod does not support authentication.

The 3G iPod does not return a response to an `Identify` command; an attached accessory must use another mechanism to determine whether the iPod has successfully processed the command.

The 3G iPod does not support the Display Remote lingo (Lingo 0x03).

The 3G iPod has limited functionality when executing certain database navigation commands in Extended Interface mode:

- `SelectDBRecord` does not support the track category.
- `RetrieveCategorizedDatabaseRecords` does not support a record count of –1.
- `ResetDBSelection` does not clear the database sort order.

For descriptions of these commands, see the *iPod Extended Interface Specification*.

User Interface Restrictions

The 3G iPod does not handle Fast Forward and Rewind notifications properly (see "Command 0x09: RemoteEventNotification" (page 140)).

The 3G iPod does not support podcasts.

The 3G iPod does not support audiobooks.

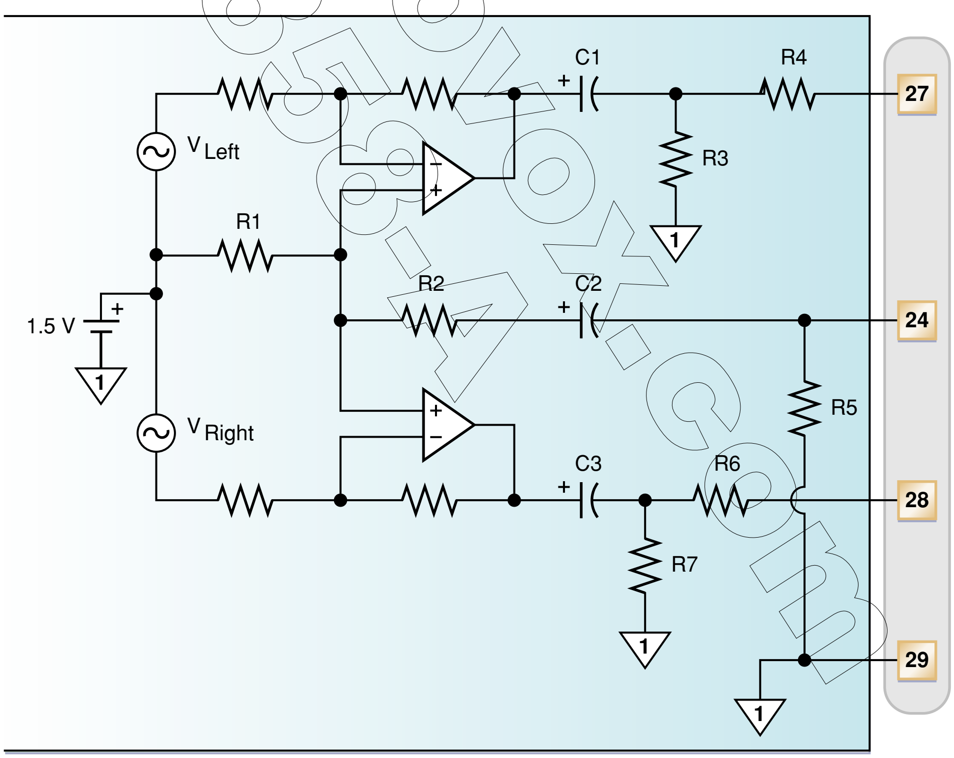
Sample Accessory Circuits

To assist developers with their accessory designs, this appendix includes sample schematics for handling audio and video in three kinds of accessories, as well as partial schematics of the iPod's internal audio and video circuitry. The schematics should help developers avoid crosstalk and extraneous noise that may affect the quality of audio and video playing from iPods.

Internal iPod Audio Circuits

Figure D-1 (page 257) shows schematically some of the audio circuitry inside a typical iPod.

Figure D-1 Typical iPod audio circuitry



Component	Value
C1, C3	10 µF
C2	4.7 µF
R1, R2	10 kΩ
R3, R7	100 kΩ
R4, R6	100 Ω
R5	47 Ω
1	iPod internal ground

Note: Unless otherwise specified, all component tolerances in this appendix are ±5%.

30-pin connector pin	Signal name	I/O	Description
24	Remote Sense	I	See "Minimizing Crosstalk and Noise" (page 38).
27	LINE-OUT L	O	Line level output to the iPod for the left channel.
28	LINE-OUT R	O	Line level output to the iPod for the right channel.
29	Audio Return	—	Audio return. This is a signal and should never be grounded inside the accessory.

The iPod's analog audio circuits, as well as the video subsystems in iPods with video output capability, share a common star point as their internal ground reference. This point is shown as 1 in Figure D-1 (page 257). It is very important to maintain the integrity of the dock connector analog signals with respect to this ground; otherwise, the analog signal handling of the entire iPod/accessory system may exhibit unwanted behavior. This behavior causes no harm to the iPod, but it can easily ruin the quality of its audio and video output.

The analog output signal paths are best thought of as loops. The video signals S Video Y (pin 21), S Video C (pin 22), and Composite Video (pin 23), leave the dock connector and must be terminated in a 75.0 Ω ±1% load if they are in use. The load return current flows to Audio Return (pin 29). These input/output connections are listed in "30-pin Connector" (page 23).

Similarly, analog audio flows from LINE-OUT L (pin 27) and LINE-OUT R (pin 28) to an external load. The load return current flows to Audio Return (pin 29). The external load may be any value in the range 1 kΩ to 100 kΩ.

To make the audio signal as free of video crosstalk as possible requires careful routing of the low side differential sense signal, which is connected to Remote Sense (pin 24). The Remote Sense line should be routed directly from the iPod 30-pin connector to the point in the accessory where audio and video share a ground connection (for example, at the ground sleeve of an A/V connector). This trace should be routed to minimize coupling with other signals, because any outside signals coupled into the

Remote Sense circuit will appear as extraneous noise in the system's audio outputs. Accessories that do not use the iPod's video output should terminate Remote Sense to Audio Return at the iPod's 30-pin connector.

Verifying an Accessory's Audio/Video Output Design

An accessory's audio/video connections to the iPod dock are most easily verified by making measurements with an audio spectrum analyzer, as well as by careful listening and viewing. In an iPod/accessory system, video-to-audio crosstalk appears in the audio band spectrum as a cluster of noise lines between 25 Hz and 1 kHz and another cluster of lines around 15.6 kHz.

The most bothersome audible artifact is a buzzing sound that may occur when the vertical video raster component appears in the audio. This component will duplicate the field rate (50 or 59.94 Hz) with an amplitude around –106 dBV. The frequency component at the horizontal rate (15625 or 15734 Hz) will be around –90 dBV. When measuring broadband noise density using 20 kHz bandwidth and 512 bins, the broadband density noise will be –122 dBV/bin ±3 dBV.

Careful listening is suggested as an adjunct to careful measurement to verify an iPod/accessory audio/video design. Careful viewing is helpful to determine whether audio-to-video crosstalk is a problem. A full-scale 1 kHz sine wave audio signal makes a good test; the audio-to-video crosstalk appears as horizontal bars in the video image when the audio is playing.

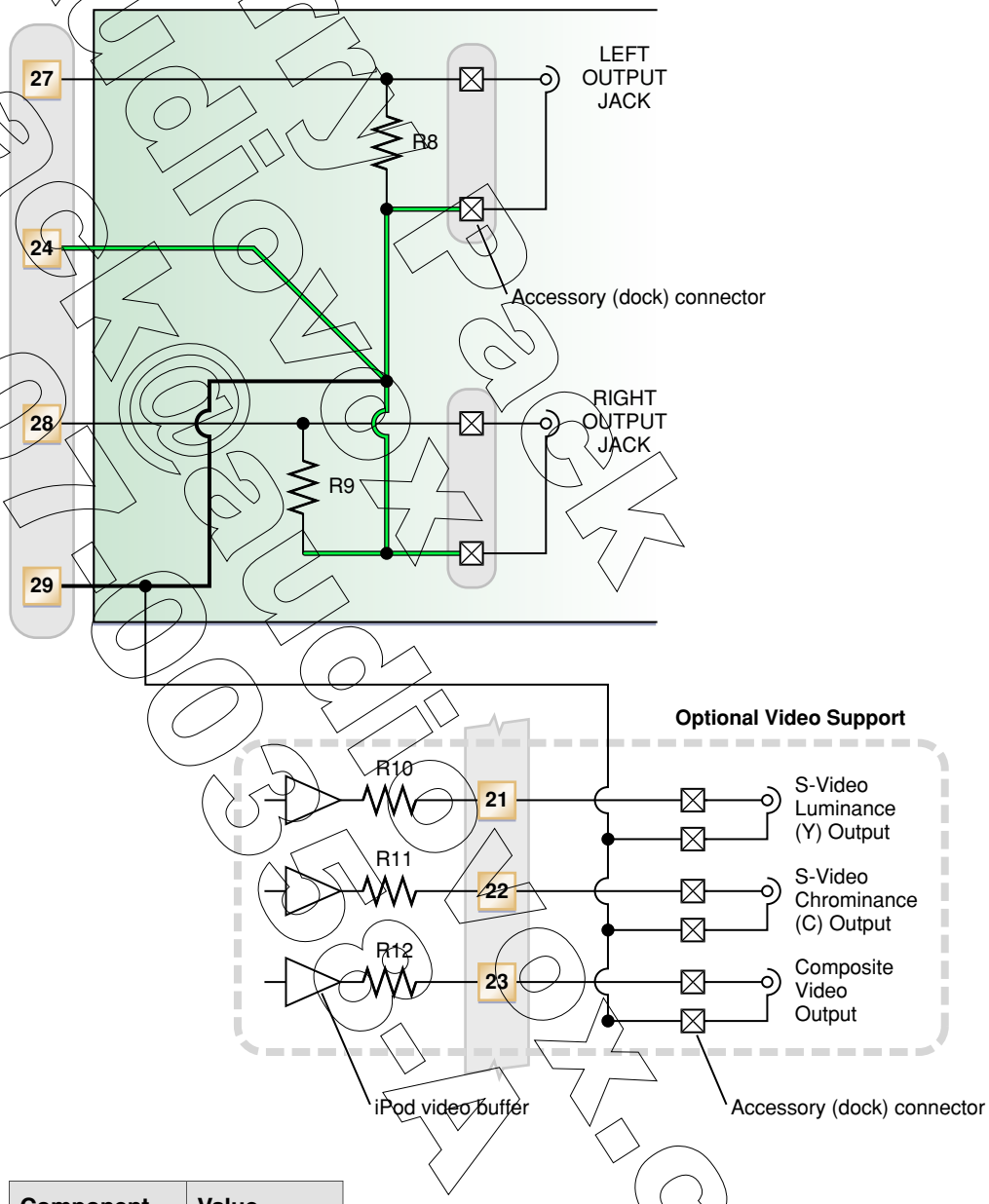
Correct accessory system designs should exhibit insignificant video-to-audio crosstalk and audio-to-video crosstalk when playing iPod outputs.

WARNING: The circuits illustrated in the rest of this appendix are samples, *not* reference designs. They are shown here only for general guidance; using them as-is will not guarantee the successful operation of any specific device.

Sample 1: A Passive Dock Accessory

Figure D-2 (page 260) shows sample circuitry of an accessory that serves as a passive dock for iPods.

Figure D-2 A passive iPod dock



Component	Value
R8, R9	47 k Ω
R10, R11, R12	75.0 Ω ±1%

30-pin connector pin	Signal name	I/O	Description
21	S Video Y	O	The luminance component of S Video.
22	S Video C	O	The chrominance component of S Video.

30-pin connector pin	Signal name	I/O	Description
23	Composite Video	O	Composite Video output.
24	Remote Sense	I	See "Minimizing Crosstalk and Noise" (page 38).
27	LINE-OUT L	O	Line level output to the iPod for the left channel.
28	LINE-OUT R	O	Line level output to the iPod for the right channel.
29	Audio Return	—	Audio return. This is a signal and should never be grounded inside the accessory.

The following notes apply to the sample design in Figure D-2 (page 260) if the accessory supports audio:

- Connect the left and right connector shields together, using a short trace, to make sure they are at the same potential.
- Connect the midpoint of the foregoing connection to pin 29. This completes the audio return circuit back to the iPod.
- Connect pin 24 to the mid-point of the first connection.
- Any external noise coupled to pin 24 appears at the output; therefore, make sure pin 24 is not routed near noisy traces.
- Resistors R8 and R9 prevent the left and right audio signals from floating when the iPod is unplugged.

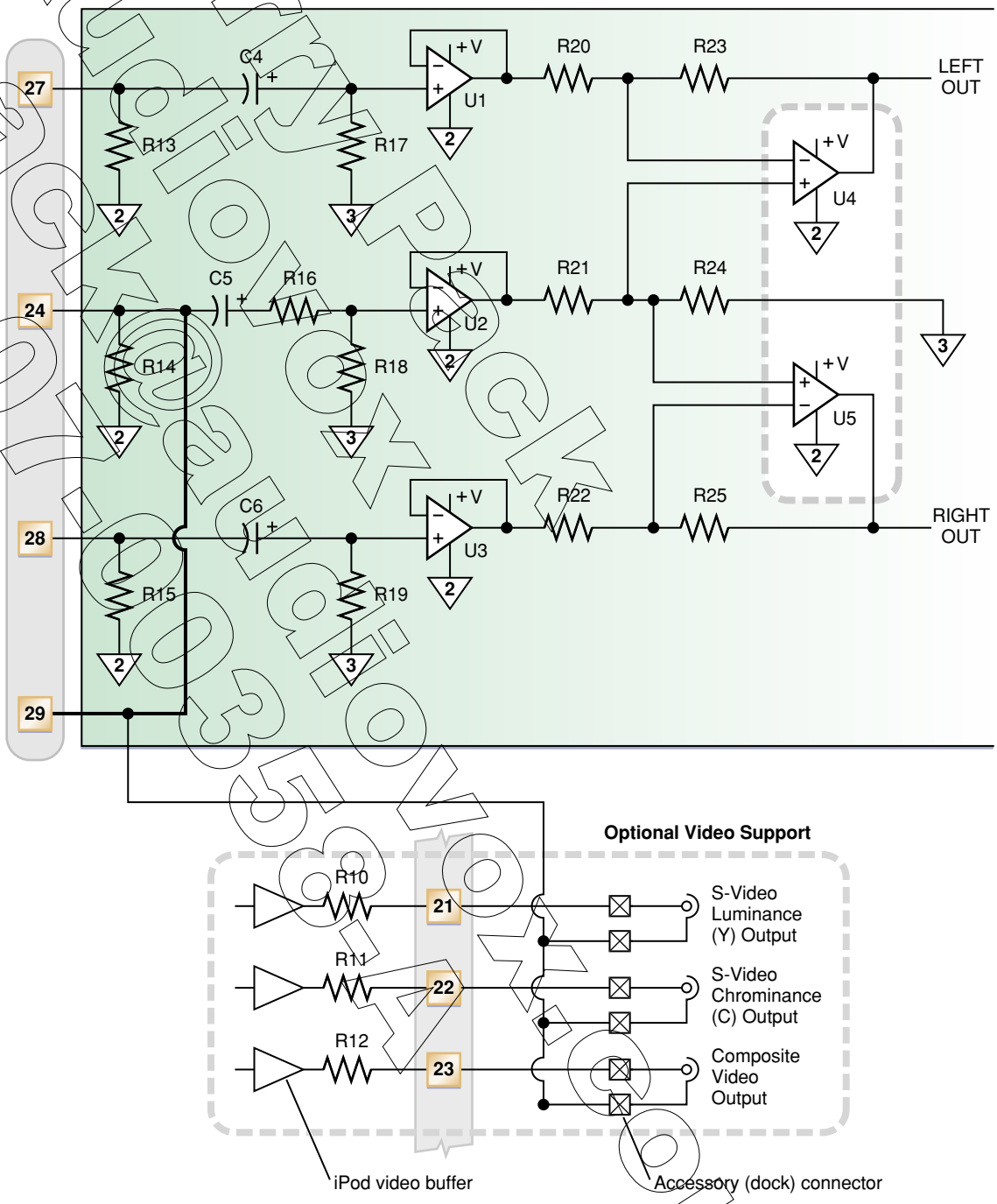
If the accessory also supports video, these additional notes apply:

- Connect the S video Y, S video C, and composite output connector shields together, using a short trace.
- Connect the audio return and the video return together at pin 29.

Sample 2: An iPod-Powered Accessory

Figure D-3 (page 262) shows an example of an accessory that draws its power from an attached iPod. Circuitry of the type shown is useful when connecting the iPod's ground to a different ground in a high common-mode noise environment. This example shows an arrangement that uses only one power supply, that in the iPod.

Figure D-3 An iPod-powered accessory



Component	Value
C4, C5, C6	10 µF
R10, R11, R12	75.0 Ω ±1%
R13, R15, R17, R19	100 k Ω

Component	Value
R14	1 kΩ
R16	100 Ω
R18, R20, R21, R22, R23, R24, R25	49.9 kΩ ±1%
U1, U2, U3, U4, U5	Differential amplifiers
2	iPod digital ground (pins 1, 2, 15, 16, and 30)
3	Accessory device reference voltage

30-pin connector pin	Signal name	I/O	Description
21	S Video Y	O	The luminance component of S Video.
22	S Video C	O	The chrominance component of S Video.
23	Composite Video	O	Composite Video output.
24	Remote Sense	I	See "Minimizing Crosstalk and Noise" (page 38).
27	LINE-OUT L	O	Line level output to the iPod for the left channel.
28	LINE-OUT R	O	Line level output to the iPod for the right channel.
29	Audio Return	—	Audio return. This is a signal and should never be grounded inside the accessory.

The following notes apply to the sample design in Figure D-3 (page 262) if the accessory supports audio:

- Resistors R20 through R25 set common mode rejection.
- R16 is needed for maximum common mode rejection.
- LEFT OUT and RIGHT OUT lead to the accessory's audio circuits. To minimize popping and clicking sounds that occur at power-up and power-down, add circuitry to the accessory.

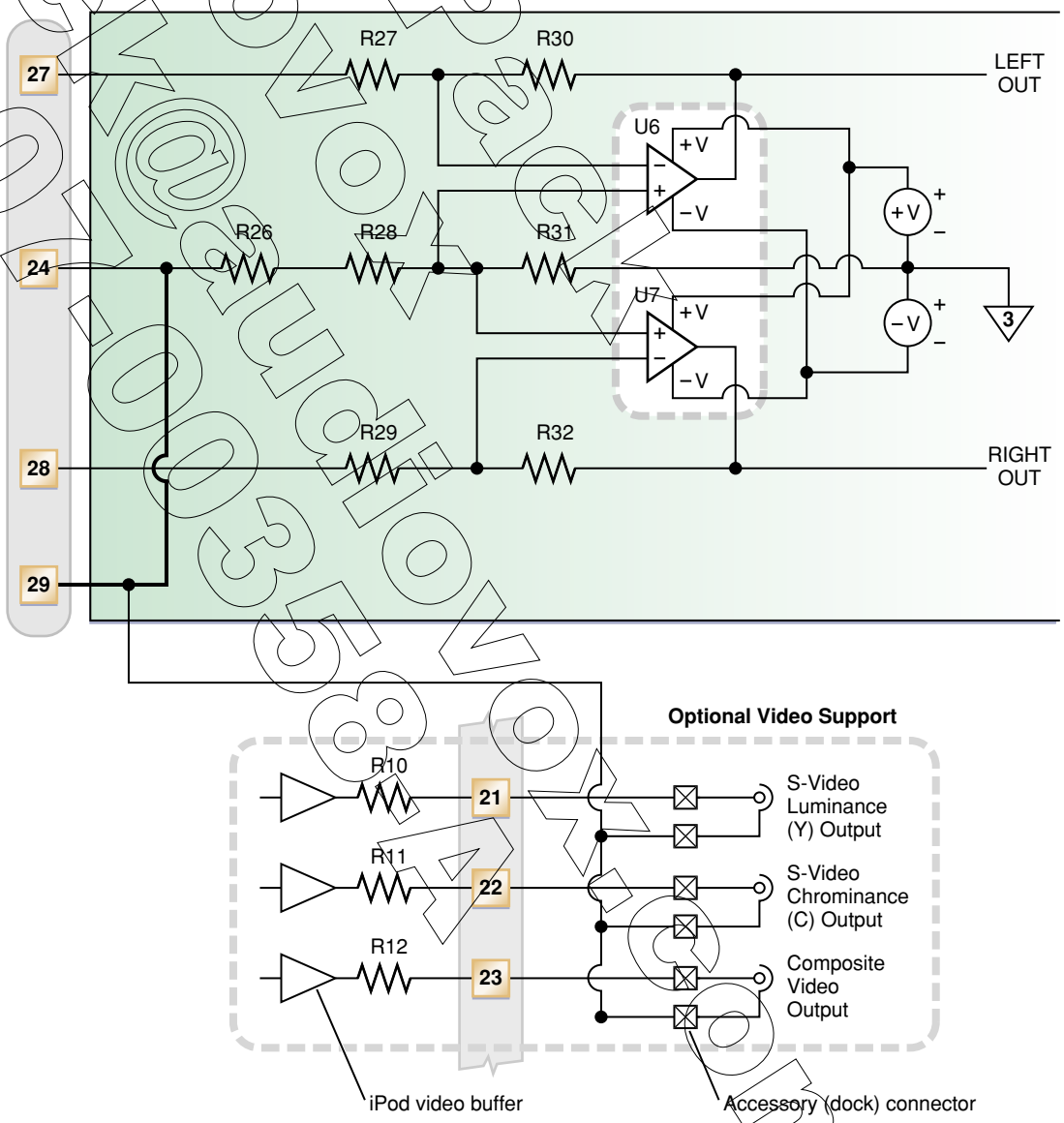
If the accessory also supports video, these additional notes apply:

- Connect the S Video Y, S Video C, and Composite output connector shields together, using a short trace.
- Connect the audio return and the video return together at pin 29.

Sample 3: A Self-Powered Accessory

Figure D-4 (page 264) shows an example of an accessory that provides its own power. Circuitry of the type shown is useful when connecting the iPod's ground to a different ground in a high common-mode noise environment.

Figure D-4 A self-powered accessory



Component	Value
R10, R11, R12	75.0 Ω ±1%
R26	100 Ω

Component	Value
R27, R28, R29, R30, R31, R32	22.1 kΩ ±1%
U6, U7	Differential amplifiers
▽3	Accessory device reference voltage

30-pin connector pin	Signal name	I/O	Description
21	S Video Y	O	The luminance component of S Video.
22	S Video C	O	The chrominance component of S Video.
23	Composite Video	O	Composite Video output.
24	Remote Sense	I	See "Minimizing Crosstalk and Noise" (page 38).
27	LINE-OUT L	O	Line level output to the iPod for the left channel.
28	LINE-OUT R	O	Line level output to the iPod for the right channel.
29	Audio Return	—	Audio return. This is a signal and should never be grounded inside the accessory.

The following notes apply to the sample design in Figure D-4 (page 264) if the accessory supports audio:

- Common mode rejection is set by resistors R30 through R35, which must have tolerances of ±1% or better.
- R29 is needed for maximum common mode rejection.
- Using a split power supply for U6 and U7, as shown in the diagram, is the easiest way to avoid poor common mode rejection at low frequencies.
- The accessory reference circuit must have a low source impedance.
- The connection to pin 24 must be direct and must not run near external noise sources.
- LEFT OUT and RIGHT OUT lead to the accessory's audio circuits. To minimize popping and clicking sounds that occur at power-up and power-down, add circuitry to the accessory.

If the accessory also supports video, these additional notes apply:

- Connect the S Video Y, S Video C, and Composite output connector shields together, using a short trace.
- Connect the audio return and the video return together at pin 29.

AC Adapter Guidelines

Accessory developers may design AC adapters for the iPod touch, but such designs **must** follow the guidelines in this section to avoid interfering with the touch-sensing operation of the iPod touch keyboard. This appendix summarizes design guidelines for third-party developers of AC adapter accessories for the iPod touch.

Noise Reduction Using a YCAP AC Capacitor

AC adapter control switching frequencies are much higher than power line frequencies. They or their harmonics can easily interfere with the iPod touch's touch-sensor modulation frequencies. It is strongly suggested that any AC adapter design for the iPod touch include a YCAP AC capacitor (up to 1000 pF), to reduce common-mode noise at the adapter's switching frequencies. The use of such a YCAP is illustrated in Figure E-1 (page 268) .

Minimum AC Adapter Switching Frequencies

To be compatible with the iPod touch's frequency-hopping touch sensors, an AC adapter design should obey the following guidelines for its switching frequencies:

- Use a fixed switching fundamental frequency not less than 60 kHz.
- Do not use ICs with switch control loops that skip switches to maintain regulation at light loads. This is sometimes called "burp" or "skip" mode.
- Fixed-frequency switching AC adapter designs are preferred, but variable frequency designs are acceptable if the time scale of the frequency variation is slow compared to the time scale of the iPod touch's touch-sensor frequency hopping and noise detection algorithms.

  In variable-frequency AC adapters switching at a frequency below 500 kHz, control loops should be designed such that there is no operating mode in which a repetitive or periodic event occurs such that the switch frequency changes more than 60 kHz during any one second. Variable-frequency switching supplies should also keep their fundamental switching frequency above 60 kHz at all times.

Impedance Stability of the Diode Bridge

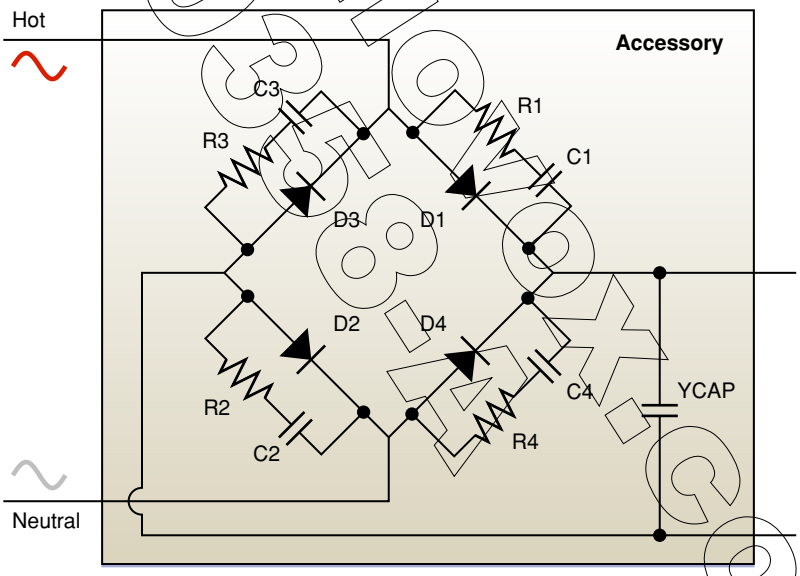
The diodes used in its full-wave bridge rectifier can be a major source of abrupt changes in an AC adapter's series impedance. To reduce unwanted touch sensor output oscillations, the AC adapter circuit should be designed such that its series impedance does not change abruptly.

If the AC adapter bridge diodes have large inherent reverse capacitance (greater than 100 pF, as many large power diodes do), then the net impedance change due to diode switching may be acceptably small; it will not adversely affect the touch sensor output. In more compact IC designs, however, the chip area of each diode may be reduced in size and its reverse capacitance may become correspondingly smaller.

To stabilize the impedance of bridge diodes with unacceptably low reverse capacitance, follow the example shown in Figure E-1 (page 268) . In this example, capacitors C1, C2, C3, and C4 have been placed in parallel with diodes D1, D2, D3, and D4 to stabilize the bridge impedance. Their values are larger than the inherent reverse capacitances of the diodes.

Resistors R1, R2, R3, and R4 are optional; if included, they can block noise at very high frequencies, which can help with EMI compatibility. The suggested values of R1, R2, R3, R4 shown were chosen to have trivial levels of impedance relative to the impedances of C1, C2, C3, and C4 at power line frequencies.

Figure E-1 Typical diode bridge circuit for an AC adapter



Component	Value
C1, C2, C3, C4	47 pF typical
R1, R2, R3, R4	2 kΩ typical
YCAP	up to 1000 pF, as needed

Physical Dimensions and Connector Layout Specifications

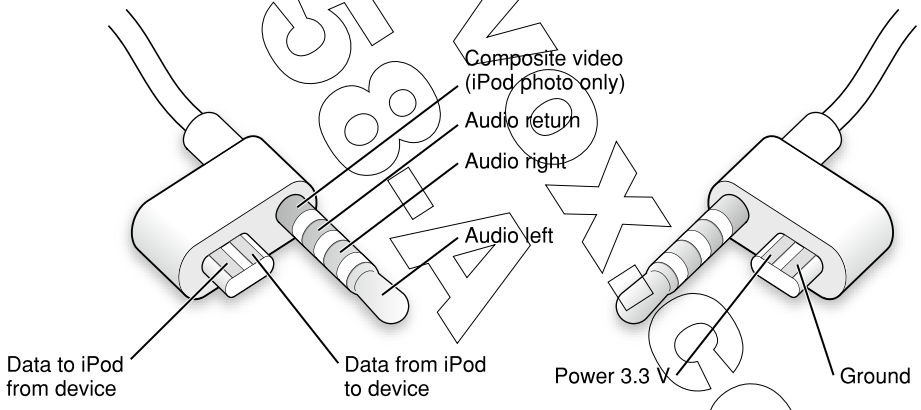
30-pin Connector

Contact Apple Developer Relations for detailed dimensional drawings.

9-Pin Audio/Remote Connector Layout

Figure F-1 (page 269) shows a diagram of the 9-pin Audio/Remote connector. Contact Apple Developer Relations for detailed dimensional drawings.

Figure F-1 Audio/Remote connector



Glossary

authentication A mechanism used by an iPod to verify whether an attached device is an authorized accessory and by an accessory to authenticate the iPod, if desired.

checksum The byte sum of packet bytes from the payload length through the last packet byte. This is used to validate the contents of a command packet. For a valid packet, the sum of the bytes, including the checksum byte, must be `0x00`. The packet checksum byte—the last byte in a packet—should be the 2's complement (the negative) of the sum of the payload length byte up to, but not including, the packet checksum byte.

deprecated Used to describe a technology or feature that is supported but whose use is discouraged and not recommended. Such a technology or feature has typically been replaced by a newer one and is likely to become unsupported in the future.

device An external electronic component connected to the iPod using the 30-pin connector or the 9-pin Audio/Remote connector.

HID (Human Interface Device) HID is a standard USB class. A USB host such as a PC or Macintosh will recognize any attached USB device that supports a HID interface and makes it available to the application layers of the operating system via a set of programming interfaces. A common application of a HID interface is a USB mouse or joystick.

HID report A single unit of data that is used to send information to the HID interface of the iPod or from the iPod to the host. iAP packets are broken into HID reports before being sent across the USB port link and are reassembled on the receiving side.

iUI (iPod USB Interface) A configuration of the iPod when attached as a device over USB. This configuration allows the iPod to be controlled using iAP, using a USB HID class interface as a transport mechanism.

LCB (Link Control Byte) A byte used by the iUI to indicate report sets and manage data flow.

lingo The command category used by a device. There is a General lingo that must be supported by all devices. Other lingoes are designed for use by specific devices, such as simple remote controls and microphones.

link The logical connection between an external device and the iPod via serial port or other physical connection.

packet The logical set of bytes that compose a valid command sequence. This set includes the packet start byte, packet payload length, payload, and payload checksum. Note that a sync byte is appended to the beginning of the packet when using the UART serial port as the data transport link. There are two different packet types: small format and large format.

payload The sequence of bytes consisting of the lingo, command, and data that are contained within a packet.

podcasting A way to publish multimedia files on the Internet that lets users receive new files automatically by subscription. Podcast files are typically downloaded to iPods through Apple's iTunes application.

RDS/RBDS (Radio [Broadcast] Display System) A technology for broadcasting and displaying artist, album, track titles, and similar information on FM radio receivers.

resistor-based accessory An accessory that uses an Accessory Identify resistor to access only limited functions in an iPod. Compare Serial accessory.

RSSI (Receive Signal Strength Indicator) A measure of the strength of an RF signal coming into a radio frequency tuner.

serial accessory An accessory that uses the iPod Accessory Protocol Interface to access a range of iPod functions. Compare Resistor-based accessory.

UART (Universal Asynchronous Receiver/Transmitter) A piece of computer hardware that translates between parallel and serial bits of data. A UART is usually an integrated circuit used for serial communications over a computer or peripheral device serial port.

USB (Universal Serial Bus) An interface standard for communication between a computer and external peripherals over a cable using biserial transmission.

USB descriptor A standard USB data structure that is passed from a USB device to the host upon request. Descriptors are used by the USB device to communicate its characteristics and resource requirements to the host.

USB endpoint A logical connection point that is used to set up a data transfer pipe between a USB host and interface on a device. For instance, the HID interface on the iPod uses an interrupt-type endpoint to enable a pipe for transferring data to the USB Host.

USB host A single computer connected to one or more USB devices or functions. The host is responsible for recognizing that a USB device has been attached to it and for driving the communications with the device. For the purposes of this document, the iPod is a USB device that provides a function, and the accessory is the USB host.

X.509 certificate A standard defined by the International Telecommunication Union (ITU) that governs the format of certificates used for authentication and sender identity verification in public-key cryptography. In the iAP, X.509 certificates contain the public keys used in the authentication process.

Document Revision History

This table describes the changes to *iPod Accessory Protocol Interface Specification*.

Date	Notes
2007-10-02	*Revision R30:*
	Added "Lingo 0x0C: Storage Lingo" (page 225) to Chapter 6.
	Added new Appendix B, "iTunes Tagging" (page 239) .
2007-09-05	*Revision R29:*
	Added documentation for the iPod classic, iPod 3G nano, and iPod touch.
	Added documentation for component video outputs.
	Added new command, `SetVideoDelay`, to the Digital Audio lingo.
	Deprecated the FireWire interface on the 30-pin connector.
	Added new preference IDs to `GetiPodPreferences`.
	Added design guidelines for third-party developers of AC adapter accessories for the iPod touch ("AC Adapter Guidelines" (page 267)).
	Added design guidelines for third-party developers of carrying cases for iPod touch ("iPod touch Carrying Case Design" (page 40)).
2007-06-29	*Revision R28:*
	Revised pin connections in 30-pin to FireWire cable (Figure 2-1 (page 29))
	Updated model listings to include iPhone and new iPod models (Table 5-27 (page 81))
	Updated requirements for artwork count data (Table 6-70 (page 160))
	Added caution about sending signals to the iPod UART when its serial receive block is off.
	Documented X.509 certificate classes (Table 5-2 (page 59))

Date	Notes
	Added line-out usage controls (Table 5-61 (page 104))
	Added section "USB Audio Errors on Older iPods" (page 114)
	Added authentication requirement to General lingo commands 0x1A-0x1F
	Deprecated Level V1 authentication for new designs (see "iPod Authentication of Device" (page 59))
	Deprecated "Command 0x01: Identify" (page 69)
2007-02-06	*Revision R27:*
	Added section "Minimizing Crosstalk and Noise" (page 38).
	Added new information to Table 3-1 (page 43).
	Removed autobaud on parity errors from Table 3-2 (page 45).
	Clarified UART communication rates in "UART Serial Port Link" (page 47).
	Removed Manufacturer String and Product String from "Choosing an iPod USB Configuration" (page 48).
	Added example of using iAP over USB in Figure 4-1 (page 49).
	Distinguished behavior of audio and video playback when iPod enters Extended Interface mode in "Command 0x05: EnterRemoteUIMode" (page 74).
	Added example of using Display Remote Protocol in "Lingo 0x03: Display Remote Lingo" (page 129).
	Clarified usage of `NewiPodTrackInfo` command in "Command 0x04: NewiPodTrackInfo" (page 223).
	Added new appendix, "Interfacing With the 3G iPod" (page 255).
	Added new appendix, "Sample Accessory Circuits" (page 257).
	Added temperature range to "UART Serial Port Link" (page 47).
2006-11-03	Added new information to Tables 3-1 and 5-26.
2006-10-17	*Revision R26:*
	Added new information to the note after Table 2-2.
	Updated Table 2-3.
	Added note to section "Line Level Output."
	Added new section "Headphone Jack on Video-Capable iPods."

Date	Notes
	Added new iPod models and software versions.
2006-09-12	*Revision R25:*
	Added iPod options and preferences commands (0x24-0x25 and 0x29-0x2B) to General lingo.
	Added USB Host Control lingo (0x06).
	Added RF Tuner lingo (0x07).
	Updated list of model ID strings (Table 5-26 (page 80) and Table 5-27 (page 81)).
	Corrected `RetTrackArtworkData` packet listing (Table 6-77 (page 165)).
2006-06-19	*Revision R24:*
	Added Accessory Equalizer lingo, number 0x08.
	Added USB Digital Audio lingo, number 0x0A.
	Added Authentication level V2 (X.509 certification) to General Lingo (0x00).
	Added Album Art commands (0x16-0x19 and 0x1F-0x20) to Display Remote Lingo (0x03).
	Added GetAccessoryInfo (0x27) and RetAccessoryInfo (0x28) commands to the General lingo (0x00).
	Added Dedicated Media commands (0x00-0x04) to the Simple Remote lingo (0x02).
	Added timeout and retry information to various command descriptions.
	Moved documentation of Extended Interface commands (0x03-0x06, General lingo) from the *iPod Extended Interface Specification*.
	Revised model and feature tables in Chapter 3.
	Added and updated lingo history tables in Chapter 6.
2006-02-10	*Revision R23:*
	Pages 17 and 27: Changed audio output power specification to 25 mW.
	Page 87: A simple remote device must send a data payload when all buttons are released; 200 ms timeout removed.
	Corrected Table 3-1 (page 43), page 44.
	Note, page 17: Specified Technical Note TN001i by name.
2006-01-05	*Revision R22:*

Date	Notes
	General update and reorganization of content.
	Added information on iPod power states.
	Added Figure 2-2 (page 30) and Figure 2-7 (page 36).
2005-10-12	Updated functional descriptions.
	Added new microphone lingo commands.
	Additional information about hibernate mode.
	Bug fixes for volume control and others.
2005-09-07	Added support for USB/iUI, authentication and additions to the new Display Remote lingo (0x03)
2005-03-21	Added Equalizer Control lingo support and commands.
2004-11-12	Added general lingo commands, bug fixes and pinouts for the iPod photo release
2004-08-03	Reserved the 28k pulldown resistor
2004-07-21	Added lingo command packet examples.
	Corrected doc properties.
	Added minor clarifications.
	Added new accessory detect image.
	Updated content based on internal review.
2004-05-17	Incorporated review feedback
2004-04-20	Update simple remote, doc reformat
2004-01-14	Minor clarifications
2003-08-12	5mA access power note
2003-08-04	New serial, add bottom serial
2003-07-22	Picture to show Remote Data lines
2003-07-07	License agreement
2003-04-15	Remote protocols added
2003-04-02	Car Charger Detect added
2003-02-04	Accessory Detect Resistor Change

Document Revision History

Date	Notes
2003-01-09	Rx, Tx Clarification
2002-12-04	Initial Release.

Document Revision History