What’s New in HomeKit

Session 710

Dennis Mathews HomeKit Engineering
Home Database
Home Database
Home Database
Home Sharing
Home Sharing

Sharing
Home Sharing

Sharing

- Owner invites users
Home Sharing

Sharing

- Owner invites users
- User accepts invitation
Common Protocol
Common Protocol
Common Protocol
Accessory Communication
Accessory Communication

Wi-Fi

Bluetooth Low Energy
Remote Access
Apple TV
Remote Access
Apple TV

Remote access gateway
Security
Security

End-to-end secure
Security

End-to-end secure
Perfect Forward Secrecy
Security

End-to-end secure
Perfect Forward Secrecy
Data is private
HomeKit Accessories
What’s New in HomeKit
What’s New in HomeKit
What’s New in HomeKit

Platform
What’s New in HomeKit

Platform
New Accessories
What’s New in HomeKit

Platform
New Accessories
Framework Updates
What’s New in HomeKit

Platform

New Accessories

Framework Updates
Platform
Platform
Home App
Home App

iOS devices
Home App

iOS devices
Home App
Apple Watch
Platform
Remote Access and Automation
Remote Access and Automation

Apple TV
Remote Access and Automation

Apple TV
Remote Access and Automation

Apple TV

Remote access
Remote Access and Automation
Apple TV

Remote access
Automation
Remote Access and Automation

Apple TV

Remote access

Automation

• Event & timer triggers
Remote Access and Automation

Apple TV

Remote access

Automation

• Event & timer triggers

Access control for shared users
Remote Access and Automation

Apple TV

Remote access

Automation

• Event & timer triggers

Access control for shared users

• Administrator access
Remote Access and Automation

Apple TV

Remote access

Automation

• Event & timer triggers

Access control for shared users

• Administrator access

• Control remote access
Remote Access and Automation

Apple TV and iPad

Remote access

Automation
- Event & timer triggers

Access control for shared users
- Administrator access
- Control remote access

Also supported on iPad
Multiple Remote Access Devices
Multiple Remote Access Devices
Platform
tvOS 10
HomeKit Framework
tvOS

View home configuration
HomeKit Framework
tvOS

View home configuration
Control accessories
HomeKit Framework
tvOS

View home configuration
Control accessories
Execute scenes
tvOS HomeKit Framework

Siri
tvOS HomeKit Framework

Top Movies

- DEADPOOL
- MIDNIGHT SPECIAL
- ZOOTOPIA
- 10 CLOVERFIELD LANE
- HAIL CAESAR

"Set the thermostat to 72"
Platform Recap
Platform Recap

Home App
Platform Recap

Home App
Remote Access and Automation
Platform Recap

Home App
Remote Access and Automation
HomeKit Framework for tvOS
What’s New in HomeKit

Platform
New Accessories
Framework Updates
What’s New in HomeKit

Platform
New Accessories
Framework Updates
New Accessories
Air Treatment Accessories
Air Treatment Accessories

Air Conditioners / Heaters
Air Treatment Accessories

Air Conditioners / Heaters
Air Purifiers
Air Treatment Accessories

Air Conditioners / Heaters
Air Purifiers
Humidifiers
Camera Accessories
Camera Accessories

Display live streams
Camera Accessories

Display live streams
Display still images
Camera Accessories

Display live streams
Display still images
Control the camera settings
Camera Accessories

- Display live streams
- Display still images
- Control the camera settings
- Control the speaker and microphone
Doorbell

NEW
Doorbell
Doorbell

Generates an event
Doorbell

Generates an event
Volume control
Doorbell

Generates an event
Volume control
Visual indicator control
Doorbell Camera

Generates an event
Volume control
Visual indicator control
Service Association
Service Association

HMRoom
Service Association

HMRoom
Service Association
Service Association

Automatically associates services
Service Association

Automatically associates services
Quick action controls
Rich Notifications

HMRoom
Rich Notifications
Rich Notifications
Rich Notifications

HMRoom
Rich Notifications
Rich Notifications

HMRoom

NEW

There's someone at the Front Door.
What’s New in HomeKit

Platform
New Accessories
Framework Updates
What’s New in HomeKit

Platform
New Accessories
Framework Updates
Framework Updates
Primary Service

- Fan Service
- Light Service
Primary Service

- Fan Service
- Light Service
Primary Service

- Fan Service
- Light Service
Primary Service

- Fan Service
- Light Service
public class HMService : NSObject {

    ...

    public var isPrimaryService: Bool { get }

Linked Services

Outlet1  Outlet2  Outlet3  Switch
Linked Services

Outlet1

Outlet2

Outlet3

Switch
public class HMService : NSObject {

    ...
    public var linkedServices: [HMService]? { get }

Valid Values
Valid Values

<table>
<thead>
<tr>
<th>Target Heating Cooling State</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Min Value</td>
<td>0</td>
</tr>
<tr>
<td>Max Value</td>
<td>3</td>
</tr>
<tr>
<td>State</td>
<td></td>
</tr>
<tr>
<td>0 = Off</td>
<td></td>
</tr>
<tr>
<td>1 = Heat</td>
<td></td>
</tr>
<tr>
<td>2 = Cool</td>
<td></td>
</tr>
<tr>
<td>3 = Auto</td>
<td></td>
</tr>
</tbody>
</table>
### Target Heating Cooling State

<table>
<thead>
<tr>
<th>Min Value</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Value</td>
<td>3</td>
</tr>
</tbody>
</table>

**State**

- 0 = Off
- 1 = Heat
- 2 = Cool
- 3 = Auto

NEW
Valid Values

Valid Values  [ 0, 2 ]

<table>
<thead>
<tr>
<th>Target Heating Cooling State</th>
<th>Min Value</th>
<th>Max Value</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>3</td>
<td>0 = Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = Heat</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 = Cool</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 = Auto</td>
</tr>
</tbody>
</table>

public class HMCharacteristicMetadata : NSObject {
    ...  
    public var validValues: [NSNumber]? { get }
Setting Up HomeKit Accessories

Find nearby accessories and add to home
Setting Up HomeKit Accessories

Find nearby accessories and add to home

```swift
public class HMAccessoryBrowser : NSObject {

    ...

    public func startSearchingForNewAccessories()

```
Setting Up HomeKit Accessories

Find nearby accessories and add to home

```swift
public class HMAccessoryBrowser : NSObject {
...
    public func startSearchingForNewAccessories()
```
Setting Up HomeKit Accessories

Find nearby accessories and add to home

```swift
public class HMAccessoryBrowser : NSObject {

    ...

    public func startSearchingForNewAccessories()


// HMAccessoryBrowserDelegate

    ...

    @objc optional func accessoryBrowser(_ browser: HMAccessoryBrowser,
                                        didFindNewAccessory accessory: HMAccessory)
```
Setting Up HomeKit Accessories

Find nearby accessories and add to home

```swift
public class HMAccessoryBrowser : NSObject {
    ...
    public func startSearchingForNewAccessories()
}
```

```swift
// HMAccessoryBrowserDelegate
    ...
    @objc optional func accessoryBrowser(_ browser: HMAccessoryBrowser,
                                           didFindNewAccessory accessory: HMAccessory)
```
Setting Up HomeKit Accessories

Find nearby accessories and add to home

```swift
public class HMAccessoryBrowser : NSObject {
    ...
    public func startSearchingForNewAccessories()
}

// HMAccessoryBrowserDelegate

    ...
    @objc optional func accessoryBrowser(_ browser: HMAccessoryBrowser, 
    didFinishNewAccessory accessory: HMAccessory)

extension HMHome {
    ...
    public func addAccessory(_ accessory: HMAccessory, completionHandler completion: 
    (NSError?) -> Void)
```
Setting Up HomeKit Accessories

Find nearby accessories and add to home

```swift
public class HMAccessoryBrowser : NSObject {
    ...
    public func startSearchingForNewAccessories()
}
```

```swift
// HMAccessoryBrowserDelegate
...
@objc optional func accessoryBrowser(_ browser: HMAccessoryBrowser,
                                    didFinishNewAccessory accessory: HMAccessory)
```

```swift
extension HMHome {
    ...
    public func addAccessory(_ accessory: HMAccessory, completionHandler completion: (NSError?) -> Void)
```
Setting Up HomeKit Accessories

Find nearby accessories and add to home
Setting Up HomeKit Accessories

Find nearby accessories and add to home

```
extension HMHome {
    ...
    public func addAndSetupAccessories(completionHandler completion: (NSError?) -> Void)
```

Setting Up HomeKit Accessories

Find nearby accessories and add to home

extension HMHome {

...public func addAndSetupAccessories(completionHandler completion:
    (NSError?) -> Void)
Setting Up HomeKit Accessories
Setting Up HomeKit Accessories

Searching for Accessories...
Make sure your accessory is powered on and nearby. Then tap the accessory you would like to add to your home.

Outlet

My accessory isn't shown here.
Setting Up HomeKit Accessories

Searching for Accessories...
Make sure your accessory is powered on and nearby. Then tap the accessory you would like to add to your home.

Outlet

Add “Outlet”
Position the 8-digit setup code in the frame. It can be found in the manufacturer’s packaging or on the accessory itself.
Setting Up HomeKit Accessories

Searching for Accessories...
Make sure your accessory is powered on and nearby. Then tap the accessory you would like to add to your home.

Add “Outlet”
Position the 8-digit setup code in the frame. It can be found in the manufacturer’s packaging or on the accessory itself.

Enter Code Manually

My accessory isn’t shown here.
Setting Up HomeKit Accessories

Searching for Accessories...
Make sure your accessory is powered on and nearby. Then tap the accessory you would like to add to your home.

Add "Outlet"
Position the 8-digit setup code in the frame. It can be found in the manufacturer’s packaging or on the accessory itself.

My accessory isn't shown here.
Camera Accessories

New Framework APIs
HMCameraProfile

Getting the camera profile

```swift
extension HMAccessory {
    public var cameraProfiles: [HMCameraProfile]? { get }
}
```
extension HMAccessory {
    public var cameraProfiles: [HMCameraProfile]? { get }
}
HMCCameraProfile
HMCameraProfile

Interfaces to interact with a camera
HMCameraProfile

Interfaces to interact with a camera

• Display and control the camera stream
HMCameraProfile

Interfaces to interact with a camera

• Display and control the camera stream
• Display still images
HMCameraProfile

Interfaces to interact with a camera

• Display and control the camera stream
• Display still images
• Control camera settings
HMCameraProfile

Interfaces to interact with a camera
• Display and control the camera stream
• Display still images
• Control camera settings
• Control speaker and microphone on the camera
Interfaces to interact with a camera

```swift
public class HMCameraProfile : HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? { get }

    ...
}
```
HMCameraProfile

Interfaces to interact with a camera

```swift
public class HMCameraProfile: HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? {
        get
    }
    ...
}
```
public class HMCameraStreamControl : HMCameraControl {
  ... 
  public func startStream()
  public func stopStream()
  public var cameraStream: HMCameraStream? { get }
}
HMCameraStreamControl

Controls the camera stream

```swift
public class HMCameraStreamControl : HMCameraControl {
    ...
    public func startStream()
    public func stopStream()
    public var cameraStream: HMCameraStream? { get }
}
```
public class HMCameraStreamControl : HMCameraControl {

... public func startStream()
  public func stopStream()
  public var cameraStream: HMCameraStream? { get }
}

@objc public protocol HMCameraStreamControlDelegate : NSObjectProtocol {
  @objc optional func cameraStreamControlDidStartStream( ... )
  @objc optional func cameraStreamControl( ... didStopStreamWithError error: ... )
}

Controls the camera stream
HMCameraStreamControl

Controls the camera stream

```swift
public class HMCameraStreamControl : HMCameraControl {
    public func startStream()
    public func stopStream()
    public var cameraStream: HMCameraStream? { get }
}

@objc public protocol HMCameraStreamControlDelegate : NSObjectProtocol {
    @objc optional func cameraStreamControlDidStartStream( ... )
    @objc optional func cameraStreamControl( ... didStopStreamWithError error: ... )
}
```
HMCameraStreamControl

Controls the camera stream

```swift
public class HMCameraStreamControl : HMCameraControl {
...
    public func startStream()
    public func stopStream()
    public var cameraStream: HMCameraStream? { get }
}

@objc public protocol HMCameraStreamControlDelegate : NSObjectProtocol {
    @objc optional func cameraStreamControlDidStartStream( ... )
    @objc optional func cameraStreamControl( ... didStopStreamWithError error: ... )
}
```
public class HMCameraStreamControl : HMCameraControl {

...  

    public func startStream()

    public func stopStream()

    public var cameraStream: HMCameraStream? { get }

}

@objc public protocol HMCameraStreamControlDelegate : NSObjectProtocol {

    @objc optional func cameraStreamControlDidStartStream( ... )

    @objc optional func cameraStreamControl( ... didStopStreamWithError error: ... )

}
HMCameraStreamControl

Controls the camera stream

```swift
public class HMCameraStreamControl : HMCameraControl {

    ...  

    public func startStream()
    public func stopStream()

    public var cameraStream: HMCameraStream? { get }
}

@objc public protocol HMCameraStreamControlDelegate : NSObjectProtocol {

    @objc optional func cameraStreamControlDidStartStream( ... )
    @objc optional func cameraStreamControl( ... didStopStreamWithError error: ... )
}
```
public class HMCameraStream : HMCameraSource {
    public var audioStreamSetting: HMCameraAudioStreamSetting
}

public class HMCameraStream : HMCameraSource {
    public var audioStreamSetting: HMCameraAudioStreamSetting
}

HMCameraStream
public class HMCameraStream : HMCameraSource {
    public var audioStreamSetting: HMCameraAudioStreamSetting
}

public enum HMCameraAudioStreamSetting : UInt {
    case muted
    case incomingAudioAllowed
    case bidirectionalAudioAllowed
}
public class HMCameraStream : HMCameraSource {
    public var audioStreamSetting: HMCameraAudioStreamSetting
}

public enum HMCameraAudioStreamSetting : UInt {
    case muted
    case incomingAudioAllowed
    case bidirectionalAudioAllowed
    case bidirectionalAudioAllowed
}
public class HMCameraStream : HMCameraSource {
    public var audioStreamSetting: HMCameraAudioStreamSetting
}

public enum HMCameraAudioStreamSetting : UInt {
    case muted
    case incomingAudioAllowed
    case bidirectionalAudioAllowed
}
public class HMCameraStream : HMCameraSource {
    public var audioStreamSetting: HMCameraAudioStreamSetting
}

public enum HMCameraAudioStreamSetting : UInt {
    case muted
    case incomingAudioAllowed
    case bidirectionalAudioAllowed
}
public class HMCameraStream : HMCameraSource {
    public var audioStreamSetting: HMCameraAudioStreamSetting
}

public enum HMCameraAudioStreamSetting : UInt {
    case muted
    case incomingAudioAllowed
    case bidirectionalAudioAllowed
}
HMCameraView

iOS 10, tvOS 10

View that can render a camera source

```swift
public class HMCameraView : UIView {
    public var cameraSource: HMCameraSource?
}
```
HMCameraView

iOS 10, tvOS 10

View that can render a camera source

```swift
public class HMCameraView: UIView {
    public var cameraSource: HMCameraSource?
}
```
WKInterfaceHMCamera

watchOS 3

View that can render a camera source

```swift
public class WKInterfaceHMCamera : WKInterfaceObject {
    public func setCameraSource(_ cameraSource: HMCameraSource?)
}
```
WKInterfaceHMCamera

watchOS 3

View that can render a camera source

```swift
public class WKInterfaceHMCamera : WKInterfaceObject {

    public func setCameraSource(_ cameraSource: HMCameraSource?)
}
```
// Example – Start and display a live stream from a camera accessory

import HomeKit
...

let cameraAccessory: HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.streamControl?.startStream()
}

let liveStreamView = HMCameraView()

// MARK: HMCameraStreamControlDelegate

func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
    liveStreamView.cameraSource = cameraStreamControl.cameraStream
}
// Example - Start and display a live stream from a camera accessory

import HomeKit
...

let cameraAccessory : HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.streamControl?.startStream()
}

let liveStreamView = HMCameraView()

// MARK: HMCameraStreamControlDelegate

func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
    liveStreamView.cameraSource = cameraStreamControl.cameraStream
}
// Example - Start and display a live stream from a camera accessory

import HomeKit
...

let cameraAccessory: HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.streamControl?.startStream()
}

let liveStreamView = HMCameraView()

// MARK: HMCameraStreamControlDelegate

func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
    liveStreamView.cameraSource = cameraStreamControl.cameraStream
}
// Example - Start and display a live stream from a camera accessory

import HomeKit
...

let cameraAccessory : HMAccessory = ...

if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.streamControl?.startStream()
}

let liveStreamView = HMCameraView()  // MARK: HMCameraStreamControlDelegate

func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
    liveStreamView.cameraSource = cameraStreamControl.cameraStream
}
// Example - Start and display a live stream from a camera accessory

import HomeKit
...

let cameraAccessory: HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.streamControl?.startStream()
}

let liveStreamView = HMCameraView()

// MARK: HMCameraStreamControlDelegate

func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
    liveStreamView.cameraSource = cameraStreamControl.cameraStream
}

// Example - Start and display a live stream from a camera accessory

import HomeKit
...

let cameraAccessory: HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.streamControl?.startStream()
}

let liveStreamView = HMCameraView()

// MARK: HMCameraStreamControlDelegate

func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
    liveStreamView.cameraSource = cameraStreamControl.cameraStream
}

// Example - Start and display a live stream from a camera accessory

```swift
import HomeKit
...

let cameraAccessory: HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.streamControl?.startStream()
}

let liveStreamView = HMCameraView()

// MARK: HMCameraStreamControlDelegate

func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
    liveStreamView.cameraSource = cameraStreamControl.cameraStream
}
```
HMCameraProfile

Interfaces to interact with a Camera

```swift
public class HMCameraProfile : HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? { get }
    public var snapshotControl: HMCameraSnapshotControl? { get }
    ..
}
```
Interfaces to interact with a Camera

public class HMCameraProfile : HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? { get }
    public var snapshotControl: HMCameraSnapshotControl? { get }
    ..
}
HMCameraSnapshotControl

Capture a still image

```swift
public class HMCameraSnapshotControl : HMCameraControl {
...
    public func takeSnapshot()
    public var mostRecentSnapshot: HMCameraSnapshot? { get }
}
```
HMCameraSnapshotControl

Capture a still image

```swift
public class HMCameraSnapshotControl : HMCameraControl {
  ...

  public func takeSnapshot()

  public var mostRecentSnapshot: HMCameraSnapshot? { get }
}
```
HMCameraSnapshotControl

Capture a still image

```swift
public class HMCameraSnapshotControl : HMCameraControl {
...
    public func takeSnapshot()
    public var mostRecentSnapshot: HMCameraSnapshot? { get }
}
```

```swift
@objc public protocol HMCameraSnapshotControlDelegate : NSObjectProtocol {
    @objc optional func cameraSnapshotControl( ... didTake snapshot: ...)
}
```
Capture a still image

```swift
public class HMCameraSnapshotControl : HMCameraControl {
...

    public func takeSnapshot()
    public var mostRecentSnapshot: HMCameraSnapshot? { get }
}
```

```swift
@objc public protocol HMCameraSnapshotControlDelegate : NSObjectProtocol {
    @objc optional func cameraSnapshotControl( ... didTake snapshot: ...)
}
```
Capture a still image

```swift
public class HMCameraSnapshotControl : HMCameraControl {
...
    public func takeSnapshot()
    public var mostRecentSnapshot: HMCameraSnapshot? { get }
}

@objc public protocol HMCameraSnapshotControlDelegate : NSObjectProtocol {
    @objc optional func cameraSnapshotControl( ... didTake snapshot: ...)
}
```
public class HMCameraSnapshot : HMCameraSource {
    public var captureDate: Date { get }
}
public class HMCameraSnapshot : HMCameraSource {
    public var captureDate: Date { get }
}

HMCameraSnapshot
public class HMCameraSnapshot : HMCameraSource {
    public var captureDate: Date { get }
}
import HomeKit
...

let cameraAccessory : HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.snapshotControl?.takeSnapshot()
}
let snapshotView = HMCameraView()

// MARK: HMCameraSnapshotControlDelegate
func cameraSnapshotControl(_ cameraSnapshotControl: HMCameraSnapshotControl, 
didTake snapshot: HMCameraSnapshot?, error: NSError?) {
    if error == nil {
        snapshotView.cameraSource = snapshot
    } else {
        // Error handling
    }
}
// Example – Take and display snapshot from the camera accessory

import HomeKit
...

let cameraAccessory : HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.snapshotControl?.takeSnapshot()
}

let snapshotView = HMCameraView()

// MARK: HMCameraSnapshotControlDelegate

func cameraSnapshotControl(_ cameraSnapshotControl: HMCameraSnapshotControl, 
didTake snapshot: HMCameraSnapshot?, error: NSError?) {
    if error == nil {
        snapshotView.cameraSource = snapshot
    } else {
        // Error handling
    }
}
// Example – Take and display snapshot from the camera accessory
import HomeKit
...

let cameraAccessory : HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.snapshotControl?.takeSnapshot()
}
let snapshotView = HMCameraView()

// MARK: HMCameraSnapshotControlDelegate

func cameraSnapshotControl(_ cameraSnapshotControl: HMCameraSnapshotControl,
                          didTake snapshot: HMCameraSnapshot?, error: NSError?) {
    if error == nil {
        snapshotView.cameraSource = snapshot
    } else {
        // Error handling
    }
}
// Example - Take and display snapshot from the camera accessory

import HomeKit
...

let cameraAccessory : HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.snapshotControl?.takeSnapshot()
}

let snapshotView = HMCameraView()

// MARK: HMCameraSnapshotControlDelegate

func cameraSnapshotControl(_ cameraSnapshotControl: HMCameraSnapshotControl,
    didTake snapshot: HMCameraSnapshot?, error: NSError?) {
    if error == nil {
        snapshotView.cameraSource = snapshot
    } else {
        // Error handling
    }
}
// Example - Take and display snapshot from the camera accessory

import HomeKit
...

let cameraAccessory: HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.snapshotControl?.takeSnapshot()
}

let snapshotView = HMCameraView()

// MARK: HMCameraSnapshotControlDelegate

func cameraSnapshotControl(_ cameraSnapshotControl: HMCameraSnapshotControl, 
didTake snapshot: HMCameraSnapshot?, error: NSError?) {
    if error == nil {
        snapshotView.cameraSource = snapshot
    } else {
        // Error handling
    }
}
// Example - Take and display snapshot from the camera accessory

import HomeKit
...

let cameraAccessory : HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.snapshotControl?.takeSnapshot()
}

let snapshotView = HMCameraView()

// MARK: HMCameraSnapshotControlDelegate

func cameraSnapshotControl(_ cameraSnapshotControl: HMCameraSnapshotControl,
    didTake snapshot: HMCameraSnapshot?, error: NSError?) {
    if error == nil {
        snapshotView.cameraSource = snapshot
    } else {
        // Error handling
    }
}
// Example - Take and display snapshot from the camera accessory

import HomeKit
...

let cameraAccessory : HMAccessory = ...
if let cameraProfile = cameraAccessory.cameraProfiles?.first {
    cameraProfile.snapshotControl?.takeSnapshot()
}
let snapshotView = HMCameraView()

// MARK: HMCameraSnapshotControlDelegate

func cameraSnapshotControl(_ cameraSnapshotControl: HMCameraSnapshotControl,
                          didFinish: HMCameraSnapshot?,
                          error: NSError?) {
    if error == nil {
        snapshotView.cameraSource = snapshot
    } else {
        // Error handling
    }
    // Error handling
HMCameraProfile

Interfaces to interact with a camera

```swift
public class HMCameraProfile : HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? { get }
    public var snapshotControl: HMCameraSnapshotControl? { get }
    public var settingsControl: HMCameraSettingsControl? { get }
    ...
}
```
Interfaces to interact with a camera

```swift
public class HMCameraProfile : HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? { get }
    public var snapshotControl: HMCameraSnapshotControl? { get }
    public var settingsControl: HMCameraSettingsControl? { get }
    ...
}
```
HMCameraSettingsControl
Control the settings on the camera
HMCameraSettingsControl

Control the settings on the camera

• Night Vision
HMCameraSettingsControl

Control the settings on the camera

• Night Vision
• Tilt
HMCameraSettingsControl

Control the settings on the camera

• Night Vision
• Tilt
• Zoom
HMCameraSettingsControl

Control the settings on the camera

• Night Vision
• Tilt
• Zoom
• Rotation
Control the settings on the camera

- Night Vision
- Tilt
- Zoom
- Rotation
- Mirroring
Interfaces to interact with a camera

```cpp
public class HMCameraProfile : HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? { get }
    public var snapshotControl: HMCameraSnapshotControl? { get }
    public var settingsControl: HMCameraSettingsControl? { get }
    public var speakerControl: HMCameraAudioControl? { get }
    public var microphoneControl: HMCameraAudioControl? { get }
}
```
HMCameraProfile

Interfaces to interact with a camera

```java
public class HMCameraProfile : HMAccessoryProfile {
    public var streamControl: HMCameraStreamControl? { get }
    public var snapshotControl: HMCameraSnapshotControl? { get }
    public var settingsControl: HMCameraSettingsControl? { get }
    public var speakerControl: HMCameraAudioControl? { get }
    public var microphoneControl: HMCameraAudioControl? { get }
}
```
HMCameraAudioControl
HMCameraAudioControl

Controls the audio settings on the camera
HMCameraAudioControl

Controls the audio settings on the camera
• Change mute settings on the microphone
HMCameraAudioControl

Controls the audio settings on the camera

- Change mute settings on the microphone
- Change mute settings on the speaker
HMCameraAudioControl

Controls the audio settings on the camera

- Change mute settings on the microphone
- Change mute settings on the speaker
- Control the microphone gain
HMCameraAudioControl

Controls the audio settings on the camera

• Change mute settings on the microphone
• Change mute settings on the speaker
• Control the microphone gain
• Control the speaker volume
Summary
Summary

Platform
Summary

Platform

• Home App
Summary

Platform

• Home App
• Remote Access and Automation
Summary

Platform

- Home App
- Remote Access and Automation
- tvOS HomeKit Framework
Summary

Platform
• Home App
• Remote Access and Automation
• tvOS HomeKit Framework

New Accessories
Summary

Platform
- Home App
- Remote Access and Automation
- tvOS HomeKit Framework

New Accessories
- Air Treatment
Summary

Platform
• Home App
• Remote Access and Automation
• tvOS HomeKit Framework

New Accessories
• Air Treatment
• Cameras
Summary

Platform
• Home App
• Remote Access and Automation
• tvOS HomeKit Framework

New Accessories
• Air Treatment
• Cameras
• Doorbells
Summary

Platform

- Home App
- Remote Access and Automation
- tvOS HomeKit Framework

New Accessories

- Air Treatment
- Cameras
- Doorbells

Framework Updates
Summary

Platform
- Home App
- Remote Access and Automation
- tvOS HomeKit Framework

New Accessories
- Air Treatment
- Cameras
- Doorbells

Framework Updates
- Primary Service, Linked Services, Valid Values
Summary

Platform
• Home App
• Remote Access and Automation
• tvOS HomeKit Framework

New Accessories
• Air Treatment
• Cameras
• Doorbells

Framework Updates
• Primary Service, Linked Services, Valid Values
• Accessory Setup
Summary

Platform
• Home App
• Remote Access and Automation
• tvOS HomeKit Framework

New Accessories
• Air Treatment
• Cameras
• Doorbells

Framework Updates
• Primary Service, Linked Services, Valid Values
• Accessory Setup
• Camera Accessory Classes
Summary

Platform
- Home App
- Remote Access and Automation
- tvOS HomeKit Framework

New Accessories
- Air Treatment
- Cameras
- Doorbells

Framework Updates
- Primary Service, Linked Services, Valid Values
- Accessory Setup
- Camera Accessory Classes
Accessibility
MFi Program

https://developer.apple.com/mfi/
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disability and Innovation: The Universal Benefits of Accessible Design</td>
<td>Presidio</td>
<td>Tuesday 12:20PM</td>
</tr>
<tr>
<td>Designing for tvOS</td>
<td>Presidio</td>
<td>Tuesday 4:00PM</td>
</tr>
<tr>
<td>Auditing Your Apps for Accessibility</td>
<td>Nob Hill</td>
<td>Wednesday, 10:00AM</td>
</tr>
<tr>
<td>Labs</td>
<td>Frameworks Lab</td>
<td>Time</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>tvOS Lab</td>
<td>D</td>
<td>Thursday 09:00AM</td>
</tr>
<tr>
<td>HomeKit Lab</td>
<td>C</td>
<td>Thursday 09:00AM</td>
</tr>
<tr>
<td>Accessories Lab</td>
<td>C</td>
<td>Friday 09:00AM</td>
</tr>
<tr>
<td>HomeKit Lab</td>
<td>A</td>
<td>Friday 11:30AM</td>
</tr>
</tbody>
</table>