HDHomeRun Development Guide
(20100701)

HDHomeRun Software release 20100213 contains enhancements and improvements used in this guide.

The latest HDHomeRun drivers, code, and firmware can be found on the Silicondust website: http://www.silicondust.com/downloads

I. Scripting - HDHomeRun Config

The HDHomeRun can be scripted using the hdhomerun_config command line utility.

This utility is cross-platform:

- Windows, Linux, Mac, *BSD, Solaris.
- 32-bit or 64-bit operating systems.
- Big-endian and little-endian CPUs.
- PC or embedded platforms.

Compiling hdhomerun_config

Windows (pre-compiled):

Download and install the HDHomeRun software for Windows. The hdhomerun_config.exe executable can be found in the program directory - default C:\Program Files\Silicondust\HDHomeRun.

Windows (MSVC++):

Download and extract the libhdhomerun archive. Create a new empty project and include all the .c and .h files. From the project properties page under C/C++ advanced, change the Compile As type to C, click apply and then change it back to C++ -- this is a workaround for a bug in MSVC++. Under the Linker input, add Ws2_32.lib and iphlpapi.lib as Additional Dependencies.

Compile.

Cygwin/Linux/Mac/*BSD:

Download and extract the libhdhomerun archive.

Run "make"

Using hdhomerun_config

The list of supported commands can be obtained by running hdhomerun_config without any parameters:

```
hdhomerun_config discover
hdhomerun_config <id> get help
hdhomerun_config <id> get <item>
hdhomerun_config <id> set <item> <value>
hdhomerun_config <id> scan <tuner> [<filename>]
hdhomerun_config <id> save <tuner> <filename>
hdhomerun_config <id> upgrade <filename>
```

Discover the HDHomeRun devices on the network:

The discover command will find HDHomeRun devices that are on the same subnet as the host:

```
hdhomerun_config discover
```

Sending commands to a specific HDHomeRun:

The "<id>" shown above represents a unique identifier for a HDHomeRun device, this can be either Device ID, or IP address:

```
hdhomerun_config <device id> get help
hdhomerun_config <ip address> get help
```

To address by Device ID the HDHomeRun must be on the same subnet as the host.

A Device ID of FFFFFFFF can be used as a wild card for the first HDHomeRun device found on the network. Do not use this syntax if there are multiple HDHomeRun devices on the network, as the device used will be random based on discovery order, which may change between commands.

Query the list of options supported by a HDHomeRun device:

The get/set options supported by a specific HDHomeRun device can be queried using the get help command:

```
hdhomerun_config <id> get help
```

Example output:

```
Supported configuration options:
  /tuner<n>/channel <modulation>:<freq|ch>     Get/set modulation and frequency
  /tuner<n>/channelmap <channel map>           Get/set channel to frequency map
  /tuner<n>/filter 0x<nnnn>-0x<nnnn> [...]     Get/set PID filter
  /tuner<n>/program <program number>           Get/set MPEG program filter
  /tuner<n>/target <ip>:<port>                 Get/set target IP for tuner
  /tuner<n>/status                             Display status of tuner
  /tuner<n>/streaminfo                         Display stream info
  /tuner<n>/debug                              Display debug info for tuner
  /tuner<n>/lockkey                            Set/clear tuner lock
  /ir/target <ip>:<port>                       Get/set target IP for IR
  /lineup/location <countrycode>:<postcode>    Get/Set location for lineup
  /lineup/location disabled                    Disable lineup server connection
  /sys/model                                   Display model name
  /sys/features                                Display supported features
  /sys/version                                 Display firmware version
  /sys/copyright                               Display firmware copyright
  /sys/debug                                   Display debug info
```

Channelmap:

The channelmap is used to configure the auto-modulation detection and channel scan. This should be configured correctly for each tuner.

The channelmap configuration is stored in non-volatile memory so only needs to be set once.

```
format: hdhomerun_config <device id> get /tuner<n>/channelmap
format: hdhomerun_config <device id> set /tuner<n>/channelmap <channelmap>
eg:     hdhomerun_config FFFFFFFF get /tuner0/channelmap
eg:     hdhomerun_config FFFFFFFF set /tuner0/channelmap us-bcast
```

Channel maps supported by HDHR-US hardware:

Name	Description	Location
us-bcast	Digital Antenna (ATSC).	US, Canada
us-cable	Digital Cable - Normal frequency layout.	US, Canada
us-hrc	Digital Cable - HRC frequency layout.	US, Canada
us-irc	Digital Cable - IRC frequency layout.	US, Canada

Channel maps supported by HDHR-EU hardware:

Name	Description	Location
au-bcast	Digital Antenna (Australia).	Australia
au-cable	Digital Cable (Australia).	Australia
eu-bcast	Digital Antenna (Europe).	Europe, New Zealand
eu-cable	Digital Cable (Europe).	Europe, New Zealand
tw-bcast	Digital Antenna (Taiwan).	Taiwan
tw-cable	Digital Cable (Taiwan).	Taiwan

Channel scan:

To run a channel scan:

```
format: hdhomerun_config <device id> scan /tuner<n> [<log filename>]
eg:     hdhomerun_config FFFFFFFF scan /tuner0 scan0.log
```

This command will scan all channels on the selected channelmap plus any additional channelmaps associated with the selected channelmap. All standard modulation types for the selected channelmap are tested.

When a digital channel is found it will identify the programs on the channel.

The log filename is optional; if included it will log to the given filename.

Tuning a physical channel:

To set a channel use the set channel command:

```
format: hdhomerun_config <id> set /tuner<n>/channel <modulation>:<frequency>
format: hdhomerun_config <id> set /tuner<n>/channel <modulation>:<channel>
eg:     hdhomerun_config FFFFFFFF set /tuner0/channel auto:651000000
eg:     hdhomerun_config FFFFFFFF set /tuner0/channel auto:60
```

Supported modulation types can be queried with the get sys-features command:

```
hdhomerun_config <id> get /sys/features
```

To stop the tuner set the channel to none:

```
format: hdhomerun_config <id> set /tuner<n>/channel none
eg:     hdhomerun_config FFFFFFFF set /tuner0/channel none
```

Checking the signal strength:

The basic signal information can be obtained by using the get status command:

```
format: hdhomerun_config <device id> get /tuner<n>/status
eg:     hdhomerun_config FFFFFFFF get /tuner0/status
```

Example output:

```
ch=qam:33 lock=qam256 ss=83 snq=90 seq=100 bps=38807712 pps=0
```

- ch = channel requested
- lock = actual modulation detected
- ss = signal strength. 80% is approximately -12dBmV.
- snq = signal to noise quality (based on analog signal to noise ratio).
- seq = symbol error quality (number of uncorrectable digital errors detected).
- bps = raw channel bits per second.
- pps = packets per second sent through the network.

More advanced information can be obtained by using the get debug command:

```
format: hdhomerun_config <device id> get /tuner<n>/debug
eg:     hdhomerun_config FFFFFFFF get /tuner0/debug
```

Example output:

```
tun: ch=qam:33 lock=qam256 ss=84 snq=88 seq=100 dbg=22081-6930
dev: resync=0 overflow=0
ts:  bps=38809216 ut=94 te=0 miss=0 crc=0
flt: bps=38809216
net: pps=0 err=0 stop=0
```

Each line contains a prefix to indicate the type of data, followed by the values.

- tun = tuner status
  - see above section
- dev = device status
- ts = transport stream
  - bps = bits per second
  - ut = utilization percentage (100% is filled to capacity)
  - te = transport error counter (uncorrectable reception error)
  - miss = missed packet counter (jump in sequence numbers)
  - crc = crc error counter
- flt = results after pid filtering
  - bps = bits per second

- net = network status
  - pps = packets per second
  - err = packets or TS frames dropped before transmission.
  - stop = reason for stopping the stream

The counters are reset to zero upon a channel change, but may indicate a small number of errors caused before the tuner locks on the channel. As a result, diagnostics should be based on the change in values over time, and not the initial values.

Detecting the programs on a physical channel:

The HDHomeRun will detect the programs (sub-channels). Use the get streaminfo command to query the detected programs:

```
format: hdhomerun_config <id> get /tuner<n>/streaminfo
eg:     hdhomerun_config FFFFFFFF get /tuner0/channel streaminfo
```

The output format is:

```
<program number>: <virtual major>.<virtual minor> [<name>] [(<flags>)]
```

Example output:

```
3: 20.1 KBWB-HD
4: 20.4 AZTECA
```

Digital cable does not always provide the channel name or virtual channel number:

```
1: 0
2: 0 (encrypted)
3: 0 (control)
```

It may take several seconds after setting the channel for the stream information to be fully populated (depending on how long the channel takes to lock and how often the stream information is sent by the broadcaster/cable provider).

Filtering by program (sub-channel):

The HDHomeRun supports automatic PID filtering by program number:

```
format: hdhomerun_config <id> set /tuner<n>/program <program number>
eg:     hdhomerun_config FFFFFFFF set /tuner0/program 3
```

When filtering by program the PAT and PMT tables are generated by the HDHomeRun. The result is a valid single-program transport stream.

The program filter is cleared when a set channel or a set filter command is received.

Advanced: By default the PAT and PMT are generated. To also generate a ATSC-style TVCT use:

```
hdhomerun_config <id> set /tuner<n>/program "<program number>
tvct_from_pmt=<virtual major>.<virtual minor>(<name>)"

eg:
hdhomerun_config <id> set /tuner0/program "3 tvct_from_pmt=11.2(TEST)"
```

Filtering by PID:

The HDHomeRun supports arbitrary hardware PID filtering:

```
format: hdhomerun_config <id> set /tuner<n>/filter <filter>
eg:     hdhomerun_config FFFFFFFF set /tuner0/filter "0x0000-0x1FFF"
        hdhomerun_config FFFFFFFF set /tuner0/filter "0x0000 0x0030-0x0033 0x1FFB"
```

When filtering by PID the stream is filtered but otherwise unmodified.

The filter is cleared to pass-all (0x0000-0x1FFF) when a set channel command is received.

Saving a stream:

The hdhomerun_config command can be used to automate the process of saving to the local filesystem:

```
format: hdhomerun_config <id> save /tuner<n> <filename>
eg:     hdhomerun_config FFFFFFFF save /tuner0 capture.ts
```

While saving the stream, a single period "." will be displayed every second. Additionally, as of the 20080609 release, the hdhomerun_config will detect reception and network errors, replacing the "." with an alternative character to indicate the problem.

Example output:

```
............n..............................n.........ts..........
-- Video statistics --
23323 packets recieved, 2 network errors, 1 transport errors, 1 sequence errors
```

Advanced: A filename of "null" indicates no file should be created, allowing the use of the save command as a diagnostic tool.

Advanced: "-" may be used as a filename to indicate standard output, allowing the save command to be used as a pipe on supported platforms.

```
eg:     hdhomerun_config FFFFFFFF save /tuner0 - | vlc -
```

Streaming to a target machine:

Set the target IP address and port number using the set target command:

```
format: hdhomerun_config <id> set /tuner<n>/target udp://<ip>:<port>
format: hdhomerun_config <id> set /tuner<n>/target rtp://<ip>:<port>
eg:     hdhomerun_config FFFFFFFF set /tuner0/target udp://192.168.1.100:5000
eg:     hdhomerun_config FFFFFFFF set /tuner0/target rtp://192.168.1.100:5000
```

The target machine must be listening on the given UDP port. The HDHomeRun will automatically clear the target if a ICMP port unreachable message is received.

A global broadcast (255.255.255.255) or subnet broadcast can be specified however care must be taken to ensure that the broadcast traffic will not cause problems with other devices on the network. If the local network is bridged to a wireless network then the AP will typically transmit at a low broadcast speed saturating the wireless network.

Example: Streaming to VLC:

Run VLC: File, Open Network Stream. Select UDP/RTP. Specify port 5000.

Discover the HDHomeRun:

```
hdhomerun_config discover
```

Run a channel scan:

```
hdhomerun_config FFFFFFFF scan /tuner0 scan0.log
```

Set the physical channel:

```
hdhomerun_config FFFFFFFF set /tuner0/channel auto:651000000
```

Check sub-programs:

```
hdhomerun_config FFFFFFFF get /tuner0/streaminfo
```

Select a sub-program:

```
hdhomerun_config FFFFFFFF set /tuner0/program 3
```

Set the target:

```
hdhomerun_config FFFFFFFF set /tuner0/target <ip address of pc>:5000
```

II. Windows BDA Drivers

The HDHomeRun Windows software include BDA drivers (32-bit and 64-bit) for use with third party software.

Direct Show filter:

The BDA driver consists of a single Direct Show filter. The Direct Show filter is registered both as a "BDA Source Filter" (KSCATEGORY_BDA_NETWORK_TUNER) and as a "BDA Receiver Component" (KSCATEGORY_BDA_RECEIVER_COMPONENT) for compatibility. Only one instance should be used (typically KSCATEGORY_BDA_NETWORK_TUNER).

BDA Demodulator type:

Windows XP includes the older "Microsoft XXXX Network Provider" components. These "Microsoft XXXX Network Provider" components will not work with a driver that publishes multiple demodulator nodes.

Windows MCE 2005 and Vista support both the "Microsoft XXXX Network Provider" components and the newer universal "Microsoft Network Provider" component. This newer universal component supports drivers that publish multiple demodulator nodes.

Windows Media Center TV Pack (WMCTVP) and later require that a US-QAM driver publish both an 8VSB demodulator node and a QAM demodulator node. This requirement means a driver that supports US-QAM for WMCTVP will not work with the older "Microsoft XXXX Network Provider" components.

Many third party BDA applications continue to use the older "Microsoft XXXX Network Provider" components for compatibility with XP. This has the advantage of supporting XP, but the disadvantage of not being able to programatically detect if an ATSC tuner supports US-QAM.

Unfortunately the conflicting combinations mean that it is not possible for a driver to publish one set of features. To solve this problem the HDHomeRun BDA driver will auto-detect which network provider(s) are part of the graph and use this detection to select which demodulator nodes to publish.

It is recommended that an application attempt to use the universal "Microsoft Network Provider" component, and only use the "Microsoft XXX Network Provider" components if the universal component fails to instantiate.

The following table shows the demodulator nodes published by the HDHomeRun driver based on the network provider detected.

Hardware	Network Provider	Demodulator Node Type
ATSC	Universal	KSNODE_BDA_8VSB_DEMODULATOR+KSNODE_BDA_QAM_DEMODULATOR
ATSC	ATSC	KSNODE_BDA_8VSB_DEMODULATOR
DVBT/C	Universal	KSNODE_BDA_COFDM_DEMODULATOR+KSNODE_BDA_QAM_DEMODULATOR
DVBT/C	DVBT	KSNODE_BDA_COFDM_DEMODULATOR if user configures "Digital Antenna".
DVBT/C	DVBC	KSNODE_BDA_QAM_DEMODULATOR if user configures "Digital Cable".

Frequency Offset:

With **analog** TV the channel frequency is typically expressed as the video carrier frequency.

With **digital** TV the channel frequency is typically expressed as the digital center frequency.

The HDHomeRun hardware uses the digital center frequency to tune a channel.

For ATSC/US-QAM operation the Windows BDA system uses the NTSC video carrier frequency. The HDHomeRun BDA driver will add 1.75MHz to the requested frequency to convert the requested frequency into the digital center frequency needed by the hardware.

For DVBT/DVBC operation the Windows BDA system uses the digital center frequency. The BDA driver will pass the requested frequency directly to the hardware.

Program/PID filtering:

The HDHomeRun supports hardware PID filtering. When used this reduces the network bandwidth to that of the sub-channel being watched/recorded. This is useful for US-QAM as the HDHomeRun will stream 80Mbps of network traffic if both tuners are streaming unfiltered.

There are three options for using the hardware PID filter support - filtering by program number, custom PID handling, or built in Windows PID handling.

Filter by program number:

The HDHomeRun driver supports a custom IHDHomeRun_ProgramFilter interface to allow the BDA application to select a program (sub-channel) of interest. The HDHomeRun will automatically detect the required PIDs and set the PID filter appropriately.

The API consists of two functions - put_ProgramNumber and get_ProgramNumber.

1) Locate the KSNODE_BDA_PID_FILTER node of the HDHomeRun tuner.
2) Get the IHDHomeRun_ProgramFilter interface.
3) Set the program number (1-65535) by calling put_ProgramNumber. To pass all programs unfiltered pass 0 as the program number. To disable program filtering and return to PID filtering pass -1 as the program number.

Custom PID handling:

The HDHomeRun driver supports the Microsoft IMPEG2PIDMap interface for configuring the hardware PID filter.

4) Locate the KSNODE_BDA_PID_FILTER node of the HDHomeRun tuner.
5) Get the IMPEG2PIDMap interface.
6) Call MapPID and UnmapPID APIs are needed.

For ATSC, typically the application will set he PID filter to PAT(0x0000) + PSIP(0x1FFB) on a channel change.

Once the PAT has been detected and processed the application will enable the PMT PIDs identified in the PAT.

Then, once the desired PMT has been detected and processed the application will enable the ES PIDs (audio and video) identified in the PMT.

Built in Windows PID handling:

XP: The Windows BDA system supports the PID filtering APIs but does not set the PID filter.

XP + MCE 2005: The Windows BDA system supports the PID filtering APIs and will set the PID filter automatically when using the standard Windows BDA components.

Vista: The Windows BDA system supports the PID filtering APIs and will attempt to set the PID

filter automatically. There is a bug in one of the standard Windows BDA components that prevents this from working.

Windows 7: The Windows BDA system supports the PID filtering APIs and will set the PID filter automatically when using the standard Windows BDA components.

HDHomeRun Setup will auto-detect if the OS has working built in PID filter support and can configure the driver appropriately.

MCE 2005 or Windows 7 is required to test the built in Windows hardware PID filter handling. It is important to test both changing to a new frequency and changing to a different sub-channel on the same frequency.

Built in Windows PID handling cannot be used with normal XP (non MCE 2005) or normal Vista (non WMCTVP).

US digital cable support:

The HDHomeRun BDA driver supports native US-QAM (ClearQAM) operation, as well as providing a channel remap mechanism to allow US digital cable to be used with legacy ATSC-only applications.

The modulation is auto-detected by the HDHomeRun. The only application change typically needed to tune a native digital cable channel is to pass a cable frequency rather than an OTA frequency.

Modulation:

US digital cable providers mostly use QAM256 modulation, however can also use QAM64 and/or 8VSB modulation.

The HDHomeRun uses the modulation type set by the application as a hint, but always auto-detects the modulation type.

The general-purpose method for detecting if a driver supports auto-modulation detection is to query the "IBDA_AutoDemodulate" interface. If this interface exists, call put_AutoDemodulate. If this function returns OK then the tuner supports automatic modulation detection.

Channel map:

There are three common channel number to frequency tables used by US cable providers - Cable, IRC, and HRC.

The Cable and IRC channel maps are almost identical. With the exception of channels 5 and 6, all IRC channels are within 25kHz of the matching Cable channels (less than the tuning resolution of a typical tuner).

To test all US cable channels an application must test all Cable channels, all HRC channels, and IRC channels 5 and 6.

The HDHomeRun driver can operate in one of four modes:

Native: the application is responsible for setting the frequency. The HDHomeRun will tune the requested frequency (plus 1.75MHz to convert the requested frequency from NTSC video carrier frequency to digital center frequency).

AutoUSCableMap: used for legacy applications only support the Cable channel map. The HDHomeRun driver will look up the requested frequency and compare with the channel configuration recorded by HDHomeRun Setup. The requested frequency is converted to a HRC or IRC frequency as needed.

AutoUSCableMapFiltered: used for legacy applications that only support the Cable channel map. The HDHomeRun driver will look up the requested frequency and compare with the channel configuration recorded by HDHomeRun Setup. The requested frequency is converted to a HRC or IRC frequency as needed.

If the requested frequency is known not to contain unencrypted digital video then the channel will be set to "none". This will result in a zero signal strength and speed up the channel scan process.

RemapFrequency: used for legacy applications that support DVB-T but not DVB-C. The HDHomeRun driver will look up the requested OTA frequency in the lineup file (RemapTrigger field). If a match is found it will tune the actual frequency specified in the lineup file.

RemapProgram: used for legacy applications that support ATSC but not ClearQAM. The HDHomeRun driver will look up the requested OTA frequency in the lineup file (RemapTrigger field). If a match is found it will tune the frequency+program specified in the lineup file. When operating in this mode the HDHomeRun will generate the PAT, PMT, and VCT information needed to work with ATSC only applications. When using this mode only 68 channels can be used as there are 68 OTA frequencies (channels 2 to 69).

Channel identification:

US Digital Cable providers typically do not sent a VCT with the stream. In cases where a VCT is included in the stream this information often incorrect.

Silicondust maintains a lineup server that tracks unencrypted digital cable channels to match with guide data. Silicondust provides access to this information via a web API – see following Lineup Server chapter.

Driver configuration:

The user specifies the application they intend to use with the HDHomeRun when the run HDHomeRun Setup. This selection sets a number of registry options to adjust the driver behavior to suit the specified application.

For development the Application can be set to "Other: Registry". This tells HDHomeRun Setup not to overwrite the registry options, thus allowing the options to be set manually.

For production please contact Silicondust to have your application added to the list of user-selectable applications in HDHomeRun Setup.

Registry:

The BDA driver configuration is per tuner, stored in:

HKEY_LOCAL_MACHINE\SOFTWARE\Silicondust\HDHomeRun\Tuners\<device id>-<tuner index>

Values are case sensitive.

Name	Valid Values
Model	Firmware model string from the HDHomeRun hardware. Used by the BDA driver to configure device specific features.
SourceType	"Digital Antenna", "Digital Cable", "Disabled". Used by HDHomeRun Setup to enable source specific features/options. Used by the BDA driver to select auto-modulation group if not specified by the application.
Source	Name of the source. Should be set to "Digital Antenna", "Digital Cable", or "Disabled" unless there are multiple directional antennas or multiple cable systems. Used by HDHomeRun Setup to manage channel lineups. Used by Dynamic Tuner Allocation feature to identify interchangeable tuners.

	Used by BDA driver to select the lineup XML file to use when a lineup-aware option such as AutoUSCableMap or Remap mode is enabled.
Application	Name of application known to HDHomeRun Setup, or “Other: Registry”. Used by HDHomeRun Setup GUI to store user selection.
BDADemodulator	Deprecated - not used by driver.
BDAPIDFilter	“Enabled” - enable hardware PID filter support. “Disabled” - disable hardware PID filter support.
ChannelMapping	“Native”, “AutoUSCableMap”, “AutoUSCableMapFiltered”, “RemapProgram”, “RemapFrequency” - see channel map section under US digital cable support.

III.Lineup Server

Silicondust maintains a lineup server to track and identify antenna and cable channels. The main focus is on identifying unencrypted digital cable channels in the US.

For end-customers the lineup information is available from the Silicondust website: http://www.silicondust.com/hdhomerun/channels

For application use the lineup information is available via a XML/HTTPS based web API.

Sub-channel tracking:

The lineup server tracks and groups sub-channels by comparing the video content using a real-time video hash. Matching is automatic and is not dependent on the frequency or the stream layout.

Video matching allows the lineup server to automatically match and track names from ATSC channels, DTA-identified channels, PSIP-identified channels, and user-identified channels.

The server is currently tracking 90,000 unencrypted cable sub-channels and typically identifies 100,000 tuner-to-tuner video matches per minute.

Encrypted vs Unencrypted:

The lineup server uses I-Frame detection from HDHomeRun devices to detect unencrypted sub-channels. This improves the reliability of reporting which sub-channels are unencrypted.

Lineup tracking:

The lineup server automatically identifies and tracks cable lineups. Each lineup is a collection of cable head-ends with the identical unencrypted channels, excluding on-demand channels. This information is used to locate the correct lineup for a user request for channel identification.

ATSC:

ATSC (over the air) channels are validated against a dictionary of latitude/longitude, callsign, transport stream ID, and virtual channel numbers. The output of the dictionary is the TMS-style callsign for matching guide data.

ATSC rebroadcast on cable:

The lineup server automatically matches over-the-air sub-channels with cable sub-channels by matching the video content. This allows the ATSC channel name to be applied to the cable channel without relying on user feedback. Likewise if a cable provider changes the channel layout the ATSC-rebroadcast channels are automatically detected and named.

Comcast DTA information:

The lineup server processes Comcast DTA information to aid with channel identification.

Sub-channels identified via Comcast DTA information are automatically matched with sub-channels on non-DTA based headends based on the video content.

User feedback:

The lineup server accepts and processes user feedback for tracking channels that cannot be automatically identified using ATSC, DTA, or PSIP matching.

Sub-channels are automatically matched between head ends based on video content. The user feedback is analyzed both at the local level for the specific head-end and at a wider level for all head-ends that broadcast the specific video feed.

Service:

For application use the lineup information is available via a XML/HTTPS based web API.

The web API is available for use with HDHomeRun tuners at no cost.

For non-HDHomeRun tuners the service is available for a nominal cost based on the number of users. This helps cover bandwidth and operating costs.

The web API documentation follows.

Web API: Channel scan/name resolution:

In most cases a DVR application will request name resolution directly after a channel scan. This may be due to a user initiated channel scan or may be a background task that runs a channel scan periodically.

Request URL:

https://www.silicondust.com/hdhomerun/lineupui?Cmd=IdentifyPrograms2

Request format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineupUIRequest>
      <Vendor>{Vendor}</Vendor>
      <Application>{Application}</Application>
      <Command>IdentifyPrograms2</Command>
      <UserID>{UserID}</UserID>
      [<DeviceID>{DeviceID}</DeviceID>]
      <Location>{Country code}:{Postcode}</Location>
      <Program>
            <Modulation>{Modulation}</Modulation>
            <Frequency>{Center frequency Hz}</Frequency>
            <TransportStreamID>{TSID}</TransportStreamID>
            <ProgramNumber>{Program number}</ProgramNumber>
            <SeenTimestamp>{YYYY-MM-DD HH:MM:SS}</SeenTimestamp>
      </Program>
      <Program>
            <Modulation>{Modulation}</Modulation>
            <Frequency>{Center frequency Hz}</Frequency>
            <TransportStreamID>{TSID}</TransportStreamID>
            <ProgramNumber>{Program number}</ProgramNumber>
            <SeenTimestamp>{YYYY-MM-DD HH:MM:SS}</SeenTimestamp>
      </Program>
</LineupUIRequest>
```

- Vendor: Test string to identify the application vendor. For example “Silicondust USA”.
- Application: Text string to identify the application, version, and platform. For example “HDHomeRun Setup 20090415 (Windows)”
- Command: “IdentifyPrograms2”.
- UserID: Globally unique string to identify the user/household. Typically this will be the user's email address or a 3 or 4 letter abbreviation of the product name followed by the product serial number (for example “FOO:12345678”). If there are multiple computers in the same household they should all return the same UserID.
- DeviceID: Optional field to specify the Device ID of a HDHomeRun if a HDHomeRun is present. For example “10100000”. This field can be present multiple times for multiple HDHomeRun devices.
- Location: Two character ISO country code followed by the postcode (colon separated). For example “US:94086”. The ISO country code should be capitalized.
- Modulation: For US digital cable the modulation should be one of “qam256”, “qam64”, or

"8vsb" (case sensitive).

- Frequency: Digital center frequency in Hz. For example, US cable channel 100 is "651000000". If the NTSC video carrier frequency is used internally by the DVR application (649250000 for channel 100) then add 1.75MHz to get the digital center frequency.
- TransportStreamID: 16-bit transport stream ID as detected from the PAT. This may be specified as an integer ("173") or in hex ("0x00AD").
- ProgramNumber: 16-bit program number from the PAT (or VCT). The values of 0 and 65535 are reserved.
- SeenTimestamp: UTC time of when the channel/program was detected in "YYYY-MM-DD HH:MM:SS" format. For a complete channel scan the timestamp for all programs can be set to the time the channel scan was initiated.
- All detected unencrypted programs should be sent in the request. The complete set of program information is used to detect the right head-end match.

Response format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineupUIResponse>
      <Command>IdentifyPrograms2</Command>
      <Program>
            <Modulation>{Modulation}</Modulation>
            <Frequency>{Center frequency Hz}</Frequency>
            <TransportStreamID>{TSID}</TransportStreamID>
            <ProgramNumber>{Program number}</ProgramNumber>
            [<GuideName>{Guide name}</GuideName>]
            [<GuideNumber>{Guide number}</GuideNumber>]
      </Program>
      <Program>
            <Modulation>{Modulation}</Modulation>
            <Frequency>{Center frequency Hz}</Frequency>
            <TransportStreamID>{TSID}</TransportStreamID>
            <ProgramNumber>{Program number}</ProgramNumber>
            [<GuideName>{Guide name}</GuideName>]
            [<GuideNumber>{Guide number}</GuideNumber>]
      </Program>
</LineupUIResponse>
```

- GuideName: UTF-8 encoded name of the program (sub-channel) matching the guide. For US digital cable Tribune/Zap2it guide names are provided. This field will be omitted if not known. Special purpose GuideName results:
  - "ONDEMAND" indicates a dynamic program that should default to disabled.
- GuideNumber: "n.n" or "n" format virtual channel number. For US digital cable the virtual channel numbers are unreliable for matching – the virtual channel number should not be used for channel matching except as a fall-back if the returned GuideName does not exist in the guide. This field will be omitted if not known.

Web API: Correction/feedback:

If the application provides a mechanism to manually enter or correct the name of a channel then this information should be sent to the lineup server using the feedback API. This feedback is used to improve the results and to avoid overwriting user corrections if the user runs another channel scan.

Request URL:

https://www.silicondust.com/hdhomerun/lineupui?Cmd=IdentifyFeedback2

Request format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineupUIRequest>
      <Vendor>{Vendor}</Vendor>
      <Application>{Application}</Application>
      <Command>IdentifyFeedback2</Command>
      <UserID>{UserID}</UserID>
      [<DeviceID>{DeviceID}</DeviceID>]
      <Location>{Country code}:{Postcode}</Location>
      <Program>
            <Modulation>{Modulation}</Modulation>
            <Frequency>{Center frequency Hz}</Frequency>
            <TransportStreamID>{TSID}</TransportStreamID>
            <ProgramNumber>{Program number}</ProgramNumber>
            <UserGuideName>{Guide name}</UserGuideName>
            [<UserGuideNumber>{Guide number}</UserGuideNumber>]
            <UserModified>{YYYY-MM-DD HH:MM:SS}</UserModified>
      </Program>
      <Program>
            <Modulation>{Modulation}</Modulation>
            <Frequency>{Center frequency Hz}</Frequency>
            <TransportStreamID>{TSID}</TransportStreamID>
            <ProgramNumber>{Program number}</ProgramNumber>
            <UserGuideName>{Guide name}</UserGuideName>
            [<UserGuideNumber>{Guide number}</UserGuideNumber>]
            <UserModified>{YYYY-MM-DD HH:MM:SS}</UserModified>
      </Program>
</LineupUIRequest>
```

- Command: “IdentifyFeedback2”.
- UserGuideName: UTF-8 encoded name of the program (sub-channel) matching the guide.
- UserGuideNumber: “n.n” or “n” format guide number for the program (sub-channel). This field can be omitted if not known.
- UserModified: UTC time of when the user modified the guide name or the guide number of program (sub-channel) in “YYYY-MM-DD HH:MM:SS” format.
- Only names that have been user-modified should be sent using this API.

Response format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineupUIResponse>
      <Command>IdentifyFeedback2</Command>
</LineupUIResponse>
```

Key technical terms of use:

- HTTPS to be used for requests.
- XML encoding to be UTF-8.
- The script or DVR application to identify itself in the request including the version.
- Requests to the lineup server to be made from the user/client application, not from a central server.
- Support for the correction/feedback API is required if the application provides a way for the user to edit channel names.

- Business and Technical contact information to be registered with Silicondust.

IV.Library API

The libhdhomerun library can be used for direct programmatic control of the HDHomeRun.

See "Scripting - HDHomeRun Config" for platform compatibility and information on compiling.

Discovery API

The discovery API is used to find HDHomeRun units on the network - documented in hdhomerun_discover.h.

For repetitive background polling use hdhomerun_discover_create() and hdhomerun_discover_find_devices().

For less frequent use the hdhomerun_discover_find_devices_custom() API provides discovery using a single call.

Note: the Discovery API will return the list of HDHomeRun units on the network but does not indicate the number of tuners associated with each HDHomeRun unit. This will be addressed in the future. If this affects your application please contact support@silicondust.com.

Device API

The Device API is the primary API for controlling a HDHomeRun tuner - documented in hdhomerun_device.h.

Create:

Call hdhomerun_device_create() or hdhomerun_device_create_from_str() to create a device object. See hdhomerun_device.h for parameters and usage.

Typically a device object will be created for each tuner to be controlled.

Get/Set:

The get/set APIs can be used in a similar way to the get/set commands of hdhomerun_config. See hdhomerun_device.h for the complete list of get/set functions.

For example, to set a channel call:

hdhomerun_device_create()

hdhomerun_device_set_tuner_channel()

hdhomerun_device_set_tuner_program()

Note: for Get functions that return a string the string is valid until another libhdhomerun API call.

Video stream:

Option 1: libhdhomerun video socket support:

The device-video API handles the video UDP socket, target configuration, buffering, sequence checking, etc.

After setting the channel call hdhomerun_device_stream_start() to create the video UDP socket and start the video service thread.

Call hdhomerun_device_stream_recv() periodically to receive the video stream.

The hdhomerun_device_stream_start/stop() APIs call hdhomerun_device_set_tuner_target() internally as needed.

Option 2: external video socket support:

For applications with existing UDP or RTP video support use hdhomerun_device_set_tuner_target() to set the target to the listening UDP port.

When streaming is no longer required use the same function to set the target to "none".

Resource lock:

The resource lock API is used to prevent conflicts when there are multiple hosts on the network controlling the same set of tuners.

Low level: use hdhomerun_device_tuner_lockkey_request() to request exclusive control of the tuner. This function will return success (1) if the request was successful, reject (0) if the tuner is in use by anther host, or error (-1) if a communication error occurs.

When the tuner lock is no longer required use hdhomerun_device_tuner_lockkey_release() to release the resource lock.

The hdhomerun_device_tuner_lockkey_force() API will cause a resource lock to be released regardless of which host owns the lock. This function should only be called if the user has been presented with a dialog and the user has explicitly requested that the lock be overridden.

The HDHomeRun will self expire the resource lock if not streaming and no commands are received in a 30s period. If the host application crashes without releasing the resource lock this timeout will allow the tuner to be used again. The exception - if the tuner is streaming video and the host has a firewall that prevents ICMP port unreachable messages the tuner will not be informed that the video is no longer being accepted by the host. Streaming will continue and the tuner will maintain the lock.

To avoid this situation use the Device Selector API to request the resource lock. This will auto-detect and release dead locks - see Device Selector API section below.

Note: the detection and automatic release of dead resource locks is planned to be moved to hdhomerun_device_tuner_lockkey_request(). If this affects your application please contact support@silicondust.com

Device Selector API

The Device Selector API is used to automatically select and lock an available tuner from a pool of tuners - documented in device_selector.h.

A pool is typically all tuners on the local network that have the same signal source.

Setup:

Use hdhomerun_device_selector_create() to create the selector object.

Use hdhomerun_device_selector_add_device() to add a device to the selection pool.

Tuner selection:

Start: use hdhomerun_device_selector_choose_and_lock() to find and lock an available tuner.

Stop: use hdhomerun_device_tuner_lockkey_release() to release the lock when the tuner is no longer required.

Channel change: there are two options for handling the resource lock when changing channel:

1) Release the resource lock by calling hdhomerun_device_tuner_lockkey_release(), then call hdhomerun_device_selector_choose_and_lock() to find and lock an available tuner.

2) Refresh the existing lock by calling hdhomerun_device_tuner_lockkey_request(). If the application still holds the resource lock the request function will return success (1). If the lock has been lost at some point but the tuner is available the request function will reacquire the lock and return success (1). If the lock as been lost at some point and the tuner is now locked by another host the request function will return busy (0). In this situation the application should call hdhomerun_device_tuner_lockkey_release() to release the local lock state and call hdhomerun_device_selector_choose_and_lock() to find another tuner.