iDevices

COYIoTE

iDevices IoT Cloud Infrastructure

Introduction To
iDevices Coyote

Coyote efficiently solves communication needs for connected devices without increasing development costs, retaining end-to-end model (no additional server programming required) with minimal memory and CPU footprints, and with **orders of magnitude lower costs for device lifetime connectivity when compared to competing solutions.** It handles firewalls and NATs smoothly, and its datagram nature greatly reduces the load on the network infrastructure while ensuring ping-level end-to-end latencies.

BUSINESS DIFFERENTIATORS

- Orders of magnitude lower costs to eliminate consumer- pricing impact
  Sample lifetime cost:
  - One-way environmental sensors: $0.50
  - Smart plugs: $1.00
  - Video streaming 1hr/day, 2hrs stored: $20.00
- Device lifetime access granted at manufacture time. Zero after-market onboarding.
- Eliminates the need for, and cost of, server programming. All interaction is between devices and controllers (handsets, etc.) If it works locally, it works remotely.
- The same technology covers a wide range of use cases, from simple sensors and switches to video streaming.
- Users, not the IoT/IP Infrastructure, own data.

TECHNOLOGY DIFFERENTIATORS

- Short latency real-time communications
- Has storage
- Wide range of applications, from remote control to live video streaming
- Support for battery operated devices with ultra-low duty cycles
- Lifetime provisioning model at time of manufacturing
- Zero on-boarding, handled by ‘coins’
- Native handling of NATs and firewalls
- Secure and audited

SERVICES

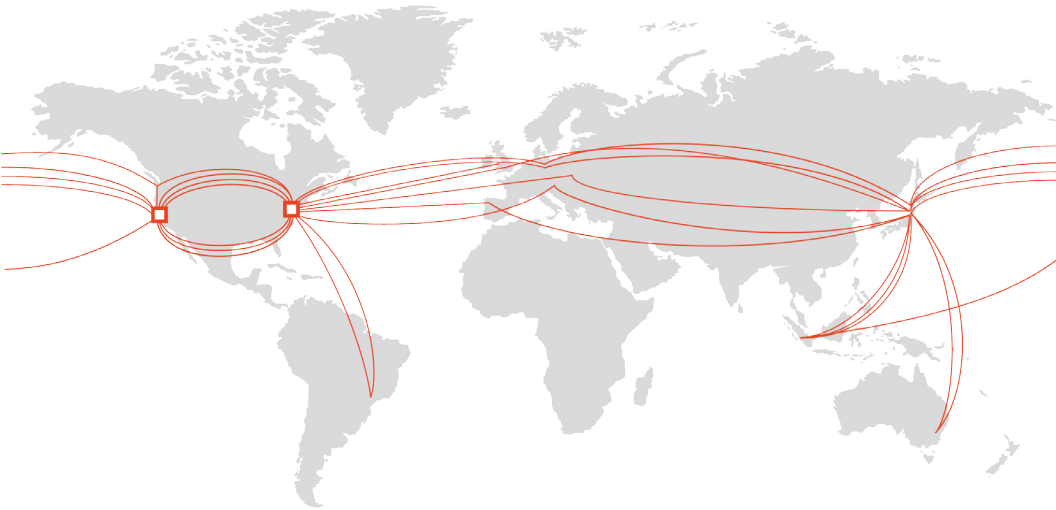
- ***Conduit:*** A fast, secure and transparent real-time communication bridge between devices and controllers. On one side this bridge is terminated by IoT/IP enabled accessories (devices), which may also provide remote connectivity to other (legacy) devices without IoT/IP capability. On the other side, it is terminated by controllers (handsets) or server-based applications. No new trust relationships are formed and the existence of Coyote is hidden from the users.

- ***EStorage:*** Key-value storage service for short or long time-shifted data exchange between accessories, controllers and server-based applications, including video streaming.

For both services, 'pairing' an accessory with one or more controllers enables the basic naming and security model. This pairing provides a unique identifier in a sparse name space, which is required to access real-time communication channels or stored records. It can be viewed as a key-value communication channel and a key-value store. 'Keys' are sparse and impossible to guess (similar to, but far more secure than telephone confernce call PINs). While all communication between Coyote and clients is encrypted, application developers are advised to use independent end-to-end encryption provided by the SDK. Coyote is without knowledge of what data goes through it or is stored in it.

Infrastructure

Coyote is a passive conduit and storage channel of data that it cannot understand. It does not need to identify clients, instead, the IoT/IP access authorization itself is based on anonymous bearer certificates ('coins'), and it does not understand the semantics of communications, or of stored records. Only devices (accessories) need coins, everyone else gets access for free, within the constraints of specific services. It involves no onboarding, meaning there are no accounts, no logins, no dashboards and no new relationships.



DDoS

General DDoS attacks are mitigated by large, uniform attack surfaces across all servers, and targeted DoS attacks are prevented by the huge private address space.

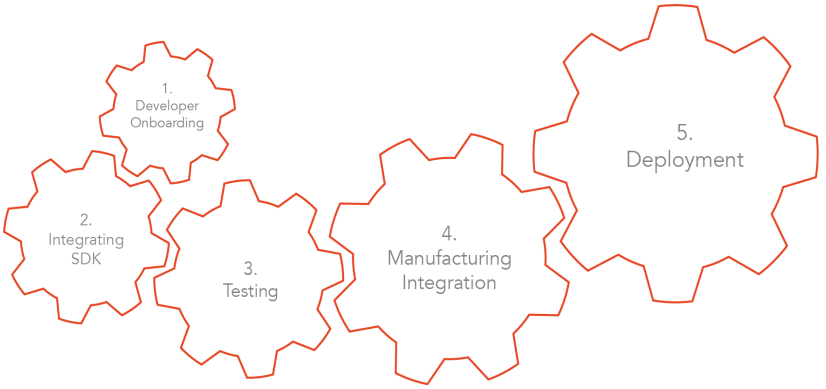
HIGH AVAILABILITY

Coyote consists of growing sets of bare metal servers placed in premium colocations at major Internet exchange points, connected to the backbone, in all cases, by two independent tier-1 bandwidth providers. The system operates in a highly redundant mode without performance or availability bottlenecks.

CAPACITY

As of July 2015, the system capacity is 400,000 transactions per second, which supports around 50 million switch-like devices, and it is expected to double by the end of 2015.

Integration Roadmap



1. DEVELOPER ONBOARDING

Authorized IoT/IP developers get access to the iDevices IoT SDK, support, test coins and the development infrastructure. In some cases, iDevices' staff performs the initial integration.

2. INTEGRATING SDK

The flow of this step depends on the nature of the project. Some possible scenarios are:

- Completely new device, built ground-up for IoT/IP
- Adding IoT/IP connectivity to the exisiting device
- Providing IoT/IP connectivity only at the handset level
- Using an existing IoT/IP-enabled device as a bridge

3. TESTING

Developers get access to both a dedicated testing infrastructure (X-Ray servers), which provides detailed feedback essential in the early phases of development, and to the fast deployment infrastructure, where they can experience IoT/IP at full speed.

4. MANUFACTURING INTEGRATION

The OEM is issued coins for their particular class of devices and each device is permanently provided with one coin. Coins are typically valid for the lifetime of the device.

5. DEPLOYMENT

Devices are sold to consumers and no further action is required. If necessary, iDevices deploys additional servers closer to the target market(s). Co-operation of Coyote with OEM is an option.

Coyote Essentials

The main objective of Coyote is to provide an efficient solution for two major aspects of device connectivity: communication setup and communication channel.

DISCOVERY + PROVISIONING

The discovery involves establishing the initial after-market association between devices (out of the box) and their controllers (usually the exisiting handsets and/or servers.) During this process the two sides learn each other's identities and perform an exchange of credentials and other information required for operation of the device. This will ensure a reliable establishment of secure connections in the future.

Coyote and iDevices IoT SDK provide several features to facilitate the provisioning:

- ***CrossCast:*** Fast and reliable Wi-Fi® communication method where the device does not need to know Wi-Fi® access point credentials or associate with one, and does not require either side to enter Access Point mode. CrossCast typically provisions the device with 2 kilobytes of information, in noisy environments, in less than 1 second.
- ***Estorage:*** A standard feature of Coyote, used as a provisioning mediator for low-duty cycle devices. EStorage allows for efficient communication for devices with ultra-low energy budget.

This provisioning process consists of several steps:

1. Establish a shared secret for the initial communication, ensuring that the involved controller actually talks to the device involved, and not the neighbor's. This is usually done with out-of-band signaling by the user (e.g. by typing in the PIN printed on the device or by Bluetooth® pairing, etc.)
2. Using the initial shared secret, provide the following information, with the initial communication channel (CrossCast or Bluetooth®):
    - Network access information, if applicable. For example, Wi-Fi® access point credentials. This step may cause the switch from the initial communication mode to the new one (IP over Wi-Fi®).
    - Channel names to be used for communicating with peers, and credentials for authenticating (such as public keys).
    - Other required information, such as specific device characteristics or settings.
3. Testing the newly established communication channel, names and credentials.

In a typical consumer scenario, the device does not have any user input hardware and initially knows only its unique ID(s), such as PIN printed on the device (or Bluetooth® pairing PIN), which is sufficient for Step 1.

The following SDK API calls perform the provisioning:

On the controller side:

*int provision_send(uint8_t *pin, int pinlen, uint8_t CSID[16], uint8_t *mypubkey, int mypubkeylen, uint8_t *devicepubkey, int *devicepubkeylen);*

[*devicepubkey* and *devicepubkeylen* are returned values.]

On the device side:

*int provision_get(uint8_t *pin, int pinlen, uint8_t *mypubkey, int mypubkeylen, uint8_t *CSID, uint8_t *controllerpubkey, int *controllerpubkeylen);*

[*CSID*, *controllerpubkey* and *controllerpubkeylen* are returned values.]

COMMUNICATION BASICS

Coyote consists of many redundant bare-metal servers located at premium colocations in North America, Europe and China, connected to multiple Tier-1 bandwidth providers.

The communication is based on UDP over IPv4, thus covering all existing consumer markets. The following table illustrates basic features of IoT/IP compared to the familiar TCP/IPv4 stack:

Feature	IoT/IP	TCP/IP
Address Space	128 bits	32 bits
Address Authority	Self-assigned by devices	Infrastructure-assigned
Address Exposure	Private	Public
Compulsory Encryption Between Routers	256-bit	None
NAT Piercing	Native	None
Storage	Short term and medium term	None

The SDK is OS-neutral and has been tested on multiple firmware, handset, desktop and server platforms. The memory footprint is very small, enabling integration when the available space is very tight (130Kb code and 10Kb RAM on ARM architectures.)

CONDUIT

Conduit enables real-time, low-latency, end-to-end communications. The SDK provides a rich functionality interface beyond the basic datagram mode, including reliable messaging, end-to-end encryption, connection hibernation, etc. The full description is available in the *iDevices® SDK User Guide*.

The Conduit API provides for fast integration for all application models. The device can be a traditional server or client, or it can operate in custom modes. Connections can be datagram or reliable.

The calls include:
iot_connection_create();
iot_send();
iot_receive();
iot_connection_close();

ESTORAGE

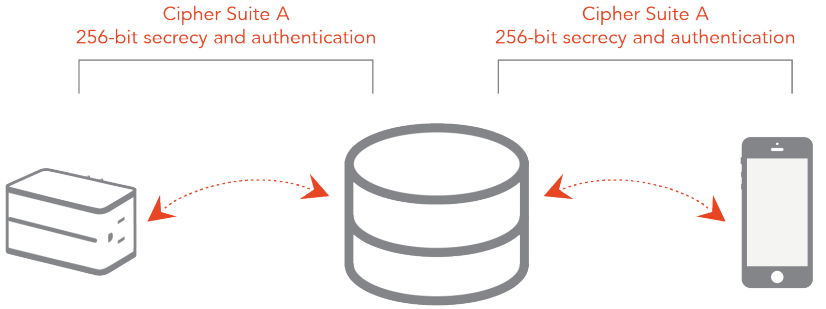
EStorage enables short-term (hours) to medium-term (days) retention of key-value records.

EStorage is typically used for:
- Storing device log records for further analysis
- Capturing communication from ultra-low duty cycle devices
- Video streaming (both live and recording)
- Firmware updates

The SDK provides additional functionality including reliable get and put, authenticated updates, etc. The full description is available in the *iDevices® SDK User Guide*.

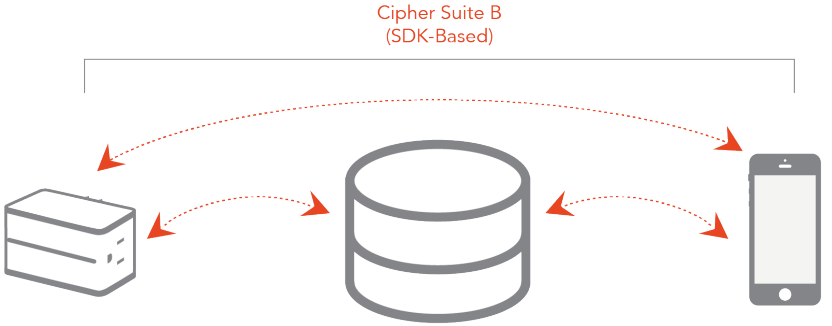
DEVICE TO SERVER SECURITY

The platform includes compulsory client<>IoT server encryption with 256-bit ciphers with forward secrecy:



END-TO-END SECURITY

Conduit communications have additional SDK-provided, end-to-end encryption with forward secrecy with a *different* cipher suite than the one used for client-server communication.



AUTHENTICATION

All authentication scenarios are provided for:

Mode	Accessory	Controller	Use Case	Actions
2-Way Authenticated	Knows C's public key. Has fixed key pair.	Knows A's public key. Has fixed key pair.	Private pairing	Each side verifies other by sending nonce. C creates ephemeral key pair, sends public key to A. A creates sesion key, sends to C via ephemeral key.
1-Way Authenticated	Knows C's public key. Has no fixed key pair.	Doesn't know A's public key. Has fixed key pair.	Trusted controller. Many accessories talking to a trusted backend server (firmware updates).	A verifies C with nonce, creates own key pair, sends public key to C. C creates session key, sends to A via A's public key.
1-Way Authenticated	Doesn't know C's public key. Has fixed key pair.	Knows A's public key. Has no fixed key pair.	Many random controllers talking to a trusted accessory (weather station).	C verifies A with nonce, creates own key pair, sends public key to A. A creates session key, sends to C via Cs public key.
Ad Hoc	Doesn't know C's public key. Has no fixed key pair.	Doesn't know A's public key. Has no fixed key pair.	VoIP/video conferencing	C creates ephemeral key pair, sends public key to A. A creates session key, sends to C via ephemeral key.

Application Examples

To illustrate Coyote integration scenarios, several different applications are described below, ranging from battery-operated sensors to live video streaming.

ULTRA-SHORT DUTY CYCLE

Combination of CrossCast provisioning, EStorage and Conduit services can enable ultra-short duty cycles, resulting in low-energy budgets and extending battery life of full-power and full-range Wi-Fi® devices to many years. These are typically environmental sensing (temperature, humidity, vibrations, etc.) One such example would be a Water Leak Detector. It is a battery-operated, low cost Wi-Fi® device (with full power radio and range) that reports water presence every few hours. The negative messages are recorded to EStorage, and a separate process periodically scans these for non-reporting (malfunctioning) devices. The positive messages are additionally relayed via Conduit to the instant-messaging alert service. The projected battery life on 2 AAA batteries is 6+ years.

DUAL PROTOCOL DEVICES

Coyote, due to its small footprint, can be implemented as an additional transport layer, either in parallel with or under the existing one, to provide complete transparency to the application layer.

For example, the iDevices® Switch product falls in this category. While it implements Apple HomeKit™ protocol, it can use Coyote if the native remote access transport of HomeKit™ is not available. Any switching between the two is transparent to the user.

HANDSET TO HANDSET RELAY

There are many Bluetooth® only devices that are locally operated with handset apps, and due to nature of Bluetooth® connections, only one handset can access any device at one time.

For the iDevices®, iGrill®, a Bluetooth® grilling thermometer, a relay feature was added to the original Bluetooth®-only app. This enables all handsets paired with a particular iGrill® to access it remotely, as long as at least one paired handset is in Bluetooth® contact with the iGrill®. There were no changes to the user interface, and connectivity switched automatically. For example, if handset A has a local Bluetooth® connection with the iGrill® acting as a relay, and handset B is remote, both have the same access. If B moves close to the iGrill® and A moves away, beyond Bluetooth® range, the switch is automatic (B becomes relay) without any interruption.

VIDEO STREAMING

The device has a USB connector and can accept modern USB cameras. The video stream is segmented and encoded in EStorage records with forward error correction. The controller, or multiple controllers, can access either the live video stream, or previously recorded footage.

Contact iDevices To Learn More

If you're interested in how Coyote can help your brand get connected, contact us at:

iDevicesinc.com/Coyote
IoT@iDevicesinc.com

or call:

800.277.3381