



# sean the flex guy TV - episode 2

## Flash and the City Conference

May 14-16 NYC, NY

- Sessions are going to be held at timeless New York City land marks
- 3-Legged Dog (3LD) Technology Center and Theatre
  - Cutting-edge space designed to facilitate the production of new media and large-scale experimental artwork
- City track completely new conference format
- Awesome extra-curricular activities

### Two Tracks:

- **Technical**
  - Focused on the code and wiring side of application architecture
    - Components
    - Frameworks
    - design patterns
- **Inspirational**
  - Creative and eye-opening sessions
    - UX
    - Design

### Speakers:

- Adam Flater
- Andy Powell
- Ben Stucki
- Chad Udell
- Chris Allen
- Chuck Freedman
- Darron Schall
- Dee Sadler
- Duane Nickull
- Elad Elrom
- Grant Skinner
- James Ward
- Jeff Tapper
- Jeffry Houser
- Jesse Freeman

- Jesse Warden
- Joel Hooks
- Juan Sanchez
- Kevin Hoyt
- Kevin Suttle
- Lee Brimelow
- Marco Casario
- Peter Elst
- Phillip Kerman
- Ryan Stewart
- Scott Janousek
- bitchwhocodes
- Yakov Fain

### **Pre-Conference Workshops**

- Mobile and Devices
- Beginning ActionScript
- Flex 4.0
- AIR 2.0

## **AIR 2.0**

### **Overview**

Adobe AIR 2 beta 2 was released on February 2, 2010.

Adobe is not yet disclosing the final release date for AIR 2. It's expected it to be in the first half of 2010.

The AIR 2 Beta 2 will expire on September 1st, 2010.

- Enhanced support for interacting with printers (beta 2)
- Support for TLS/SSL socket communication (beta 2)
- Support for the detection of mass storage devices
- Advanced networking capabilities like secure sockets, UDP support, and the ability to listen on sockets
- Support for native code integration
- The ability to open a file with its default application
- Multi-touch and gesture support
- New APIs for access to raw microphone data
- Webkit update with HTML5/CSS3 support
- Global error handling
- Improved cross-platform printing

Improved security and support for enterprise and government standards.

## Networking

- **NetworkInfo**
  - Accesses low-level network information
- Info about network interfaces
  - local IP address
  - wired or wireless networks
- **DNSResolver**
  - Performs DNS and reverse DNS lookup
  - Dispatches `DNSResolverEvent`'s.
  - `ARecord`, `MXRecord`, `SRVRecord`, `ResourceRecord`, `PTRRecord`

## Native Processes

Open files with their default application with AIR 2.0.

- **flash.desktop.NativeProcess**
  - command line integration
  - general launching capabilities on the operating system
  - monitor the standard input, output and error of the process
- **flash.desktop.NativeProcessStartupInfo**
  - basic information used to start processes
- **flash.events.NativeProcessExitEvent**
  - dispatched when processes exit
- Communicates with the native application using standard input (STDIN) and standard output (STDOUT) streams
- **ProgressEvent.STANDARD\_OUTPUT\_DATA**
- **ProgressEvent.STANDARD\_INPUT\_PROGRESS**
- Update the application descriptor's `supportedProfiles` element to use the `extendedDesktop`
  - `<supportedProfiles>extendedDesktop</supportedProfiles>`

## openWithDefaultApplication

- Opens files with the application associated with the operating system
- Can't communicate with the launched process after it's open
- Security restrictions for executables and shell scripts
- If called on a file that does not have a default application a runtime error is thrown
- Directories are opened with default file explorer (Finder, Windows Explorer)

## URLFilePromise

- Drag files from the AIR app and copy to the local computer
- Similar to dragging an image from your web browser to your desktop

## StorageVolumeInfo

- **StorageVolumeInfo**
  - keeps track of changes to the mass storage devices
  - `storageVolumeInfo.getStorageVolumes`
    - singleton instance of the `StorageVolumeInfo` object
  - Returns vector of `StorageVolume` objects corresponding to the currently mounted storage volumes.
- **StorageVolume**
  - `.drive`
  - `.name`
  - `.icon`
  - `.fileSystemType` ("FAT", "NTFS", "HFS", or "UFS")
  - `.rootDirectory`
    - `.nativePath`
  - `.isWritable`
  - `.isRemovable`
- **StorageVolumeChangeEvent**
  - `storageVolumeMount`
  - `storageVolumeUnmount`

## Microphone API

- Manipulate and record sound directly from a microphone or the line-in on the user's sound card
- Previously a remote server such as the Adobe Flash Media Server was needed to access any of these inputs
- Access the uncompressed PCM `ByteArray` data from the sound card as if you were reading it from a file on the file system using the `Sound.extract()` method
- Raw data returned is formatted as a `ByteArray` of single-channel (mono) float value, uncompressed PCM samples
  - Add effects (simple echo effects, audio gates)
  - Search for certain audio patterns
  - Save raw data to the hard drive
  - Encapsulate audio data as a WAV file
- Create **`SampleDataEvent.SAMPLE_DATA`** event listener on each **`Microphone.getMicrophone()`** instance
  - These events contain the raw PCM data
- **`Microphone.names`**
  - input sources returned by the OS

## MultiTouch

- Flash Player detects if device has touch capability
  - multiple points of contact
  - specific events for different gestures

- **GESTURE**
    - multiple points of contact and specific events for different gestures (such as rotation and pan)
  - **NONE**
    - none at all, handled as MouseEvent
  - **TOUCH\_POINT**
    - basic single point of contact (such as tap)
- Set the Multitouch class to the type of user gesture that the device is capable of capturing
- Start listening to gesture events
- **TouchscreenType**
  - **NONE**
  - **FINGER**
  - **TOUCH\_POINT**
  - **STYLUS**
  - **GESTURE**
- **Multitouch**
  - inputMode (set it to MultitouchInputMode constant)
- **MultitouchInputMode**
  - enumeration class that holds the three types of multi-touch hardware options
    - **NONE**
    - **FINGER**
    - **TOUCH\_POINT**
    - **STYLUS**
    - **GESTURE**
- **TransformGestureEvent**
  - **GESTURE\_ZOOM**
  - **GESTURE\_SWIPE**
  - **GESTURE\_PAN**
  - **GESTURE\_ROTATE**
- **TouchEvent**
  - dispatched when the Flash player detects user gestures
    - **TOUCH\_BEGIN**
    - **TOUCH\_END**
    - **TOUCH\_MOVE**
    - **TOUCH\_OUT**
    - **TOUCH\_OVER**
    - **TOUCH\_ROLL\_OUT**
    - **TOUCH\_ROLL\_OVER**
    - **TOUCH\_TAP**

## WebKit with the SquirrelFish Extreme

- AIR 2.0 uses the same branch of WebKit as the Safari 4 beta
- **CSS Transitions**
  - animate from an old value to a new value over time
- **CSS Transformations**
  - scale, rotate, and skew blocks of elements
- **CSS Animations**
  - set timings for fades, rotation, expansion, collapses, and others
- **CSS Gradients**
  - linear and radial
- **Webkit CSS selectors**
  - Access the DOM faster and easier using the Selectors API
    - Allows you to select elements within a document using CSS

## UDP sockets

- Universal Datagram Protocol
- Provides a way to exchange messages over a stateless network connection
- No guarantees that messages are delivered in order or even that messages are delivered at all
- OS's network code usually spends less time marshaling, tracking, and acknowledging messages
- UDP messages typically arrive at the destination app with a shorter delay than TCP messages
- UDP socket communication is helpful when you must send real-time information such as:
  - position updates in a game
  - sound packets in an audio chat application
- Use when data loss is acceptable, and low transmission latency is more important than guaranteed arrival
- For almost all other purposes, TCP sockets are a better choice
- Send and receive UDP messages with the DatagramSocket and DatagramSocketDataEvent classes

### To send or receive a UDP message

1. Create a DatagramSocket object
2. Add an event listener for the data event
3. Bind the socket to a local IP address and port using the bind() method
4. Send messages by calling the send() method, passing in the IP address and port of the target computer
5. Receive messages by responding to the data event
  - The DatagramSocketDataEvent object dispatched for this event contains a ByteArray object containing the message data.

**Keep in mind the following considerations when using UDP sockets:**

- A single packet of data cannot be larger than the smallest maximum transmission unit (MTU) of the network interface or any network nodes between the sender and the recipient.
  - All of the data in the ByteArray object passed to the send() method is sent as a single packet. (In TCP, large messages are broken up into separate packets)
- There is no handshaking between the sender and the target.
  - Messages are discarded without error if the target does not exist or does not have an active listener at the specified port
- When you use the connect() method, messages sent from other sources are ignored.
  - A UDP connection provides convenient packet filtering only.
  - It does not mean that there is necessarily a valid, listening process at the target address and port
- UDP traffic can swamp a network.
  - Network administrators might need to implement quality-of-service controls if network congestion occurs.
  - (TCP has built-in traffic control to reduce the impact of network congestion)