Andriod Debug Bridge

Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components:

- A client, which runs on your development machine. You can invoke a client from a shell by issuing an adb command. Other Android tools such as the ADT plugin and DDMS also create adb clients.
- A server, which runs as a background process on your development machine. The server manages communication between the client and the adb daemon running on an emulator or device.
- A daemon, which runs as a background process on each emulator or device instance.

You can find the adb tool in <sdk>/platform-tools/.

When you start an adb client, the client first checks whether there is an adb server process already running. If there isn't, it starts the server process. When the server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients—all adb clients use port 5037 to communicate with the adb server.

The server then sets up connections to all running emulator/device instances. It locates emulator/device instances by scanning odd-numbered ports in the range 5555 to 5585, the range used by emulators/devices. Where the server finds an adb daemon, it sets up a connection to that port. Note that each emulator/device instance acquires a pair of sequential ports — an even-numbered port for console connections and an odd-numbered port for adb connections. For example:

```
Emulator 1, console: 5554
Emulator 1, adb: 5555
Emulator 2, console: 5556
Emulator 2, adb: 5557
and so on...
```

As shown, the emulator instance connected to adb on port 5555 is the same as the instance whose console listens on port 5554.

Once the server has set up connections to all emulator instances, you can use adb commands to access those instances. Because the server manages connections to emulator/device instances and handles commands from multiple adb clients, you can control any emulator/device instance from any client (or from a script).

Enabling adb Debugging

In order to use adb with a device connected over USB, you must enable USB debugging in the device system settings, under Developer options.

On Android 4.2 and higher, the Developer options screen is hidden by default. To make it visible, go to Settings > About phone and tap Build number seven times. Return to the previous screen to find Developer options at the bottom.

On some devices, the Developer options screen may be located or named differently.

Note: When you connect a device running Android 4.2.2 or higher to your computer, the system shows a dialog asking whether to accept an RSA key that allows debugging through this computer. This security mechanism protects user devices because it ensures that USB debugging and other adb commands cannot be executed unless you're able to unlock the device and acknowledge the dialog. This requires that you have adb version 1.0.31 (available with SDK Platform-tools r16.0.1 and higher) in order to debug on a device running Android 4.2.2 or higher.

For more information about connecting to a device over USB, read Using hardware device

Connecting to STB using ADB

- To make a connection between STB and PC it is necessary to:
- Open command line on PC (by pressing Windows button + R)
- Run cmd command
- Drag adb.exe to console and press Enter
- Connect adb host to device:

```
adb connect #.#.#.#
connected to #.#.#.#:5555
```

If the adb connection is ever lost:

- Make sure that your host is still connected to the same network your STB is.
- Reconnect by executing the “adb connect” step again.
- Or if that doesn't work, reset your adb host and then start over from the beginning:

```
adb kill-server
```

Syntax

You can issue adb commands from a command line on your development machine or from a script. The usage is:

```
adb [-d|-e|-s <serialNumber>] <command>
```

If there's only one emulator running or only one device connected, the adb command is sent to that device by default.

If multiple emulators are running and/or multiple devices are attached, you need to use the -d, -e, or -s option to specify the target device to which the command should be directed.

Commands

The table below lists all of the supported adb commands and explains their meaning and usage.

Category	Command	Description	Comments
Target device	-d	Direct an adb command to the only attached USB device	Returns an error if more than one USB device is attached
Target device	-e	Direct an adb command to the only running emulator instance	Returns an error if more than one emulator instance is running
Target device	-s <serialNumber>	Direct an adb command a specific emulator/device instance, referred to by its adb-assigned serial number	See Directing Commands to a Specific Emulator/Device Instance
General	devices	Prints a list of all attached emulator/device instances	See Querying for Emulator/Device Instances for more information
General	help	Prints a list of supported adb commands	
General	version	Prints the adb version number	
Debug	logcat [option] [filter-specs]	Prints log data to the screen	
Debug	bugreport	Prints dumpsys, dumpstate, and logcat data to the screen, for the purposes of bug reporting	
Debug	jdwp	Prints a list of available JDWP processes on a given device	You can use the forward jdwp:<pid> port-forwarding specification to connect to a specific JDWP process
Data	install <path-to-apk>	Pushes an Android application (specified as a full path to an .apk file) to an emulator/device	
Data	pull <remote> <local>	Copies a specified file from an emulator/device instance to your development computer	
Data	push <local> <remote>	Copies a specified file from your development computer to an emulator/device instance	
Ports and Networking	forward <local> <remote>	Forwards socket connections from a specified local port to a specified remote port on the emulator/device instance	Port specifications can use these schemes: tcp:<portnum> local:<UNIX domain socket name> dev:<character device name> jdwp:<pid>
Ports and Networking	ppp <tty> [parm]...	Run PPP over USB, <tty> — the tty for PPP stream	Note that you should not automatically start a PPP connection
Scripting	get-serialno	Prints the adb instance serial number string	See Querying for Emulator/Device Instances for more information
Scripting	get-state	Prints the adb state of an emulator/device instance	

Category	Command	Description	Comments
Scripting	wait-for-device	Blocks execution until the device is online — that is, until the instance state is device	
Server	start-server	Checks whether the adb server process is running and starts it, if not	
Server	kill-server	Terminates the adb server process	
Shell	shell	Starts a remote shell in the target emulator/device instance	See Issuing Shell Commands for more information
Shell	shell [shellCommand]	Issues a shell command in the target emulator/device instance and then exits the remote shell	See Issuing Shell Commands for more information

More detailed information you can find on official website Android Debug bridge