Streaming Data Synchronously to the Handset

In this unit, you will learn how to make synchronous calls to a server from a Flash Lite 2.1 client.

Adobe

Objectives

After completing this unit, you should be able to:

- Understand the nature of XML socket communications
- Understand the server side requirements for using an XML socket
- Connect to a server side XML socket in Flash Lite 2.1 using `XMLSocket()`
- Build a chat application

Note: the `XMLSocket()` *class used in this unit is a Flash Lite 2.1 feature, and is not available in Flash Lite 2.0 .*

Introducing XML Sockets

- A socket is a persistent network connection between two system endpoints.
- Because it is persistent, it is fundamentally different than the request-response (asynchrounous) oriented connections, such as HTTP and FTP, most commonly used in internet development.

Understanding Flash Lite socket connections

Key points to understand about socket connections include:

- Better performance: because the connection remains open, rather than begin re-created for each use (as with HTTP and FTP)
- Higher resource use: continuously available, so continuously in memory; must close when no longer needed to relieve memory and increase security

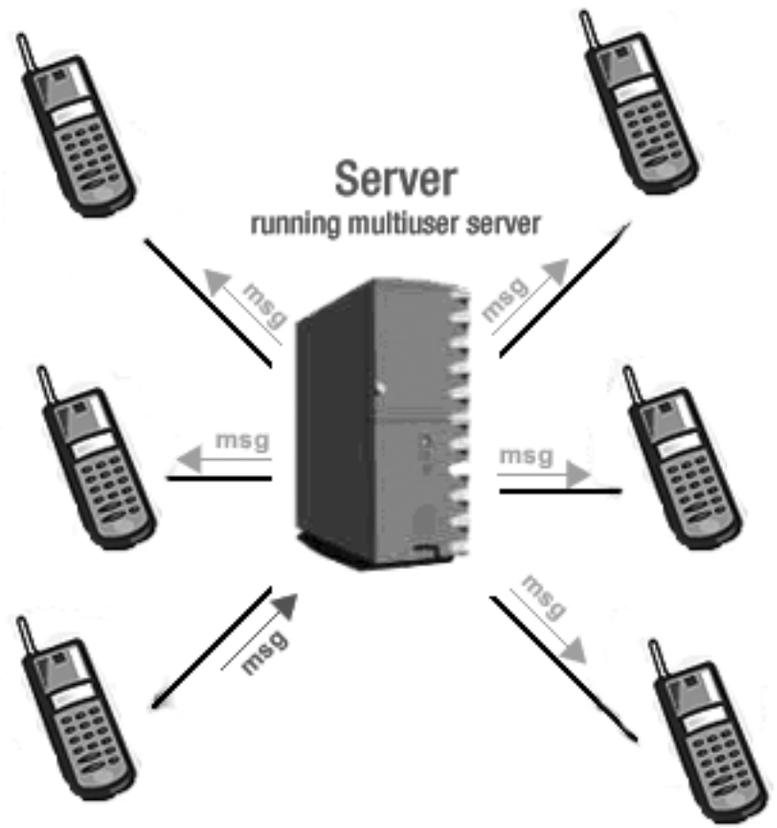
Working with socket servers

To support `XMLSocket()` connections, the targeted server must run a persistent in-memory process ("daemon" or "service") which understands the industry-standard protocol used by this class:

- XML messages are sent over a full-duplex TCP/IP stream socket connection.
- Each XML message is a complete XML document, terminated by a zero (\0) byte.
- An unlimited number of XML messages can be sent and received over a single, persistent `XMLSocket()` connection.

Working with socket servers

Multi-User Socket Server



Adobe

Working with socket servers

- This server process manages communication with the `XMLSocket()` object running in each connected client, and presents some form of data service to the client.
  - For example, one client may send data to the socket server, which then pushes the data back out to all other connected clients, creating a chat service.
  - Or, the server may send a continuous stream of stock price updates, sports scores etc.

Using allowed ports

- Server network-connection points are called "ports".
  - Each port is identified by a 16-bit integer value, up to 65535.
  - The first 1024 port number are sometimes referred to as "well-known" ports, and are largely reserved for use with widespread standard protocols. For example: HTTP (port 80), FTP (port 21), POP (port 110), SMTP (port 25), and TELNET (port 23).
  - To avoid the traffic and security concerns related to well-known ports, `XMLSocket()` can only connect to port numbers higher than 1024.

Working within Flash Lite player security

SWF files loaded from a domain server into the Flash Lite player may only connect to services available from ports in same domain, unless a cross-domain policy file (`crossdomain.xml`) in in place in the target domain.

*Note: The debug Flash Lite player and emulator does **not** enforce domain security, so connections available during development are not necessarily available running live.*

Introducing an XML socket server: Jabber

- Flash Lite is a very strong tool for developing multiuser mobile phone applications, but it must be paired with other technologies to implement the server side of the system.
- In this course, you will use a pre-existing socket server available at http://www.jabber.org. Jabber is a streaming XML protocols enabling any two entities on the Internet to exchange presence and messages in near real-time.
- Jabber uses a client-server architecture, not a peer-to-peer architecture as some other messaging systems do. So, all Jabber data sent from one client to another must pass through at least one Jabber server.

Note: Jabber clients are free to negotiate direct connections, for example to transfer files, but those "out-of-band" connections are first negotiated within the context of the client-server framework.

- Jabber protocols are 100% XML, which enables structured, intelligent messaging between human users and also between software applications.

Understanding Jabber client behavior

- A Jabber client connects to a Jabber server on a TCP socket over port 5222 (servers connect directly to each other over port 5269).
- This connection is persistent ("always on") for the life of the client's session on the server.
- The client does not need to poll for messages, as an email client does. Rather, any message intended for delivery is immediately pushed out to its target client, if connected.
- The Jabber server tracks whether a target client is online, and if not, stores messages for delivery when it connects again.

Jabber servers

The following Jabber servers work with this Flash Lite course application.

- jabber.com
- 12jabber.com
- jabber.anywise.com
- jabber.minus273.org
- jabber.palomine.net

Jabber clients

- Pre-built Jabber chat clients are available for both Windows and Apple operating systems.
- In this course, we will exchange messages between a pre-built client, and a Flash Lite chat client we will build.
  - Jabber Client list: http://www.jabber.org/software/clients.shtml
  - Exodus: http://www.jabberstudio.org/
  - Adium: http://www.adiumx.com/
  - iChat: http://www.apple.com/macosx/features/ichat/
  - Psi: http://psi-im.org/

Walkthrough 1: Setting Up a Jabber Account

In this walkthrough, you will perform the following tasks:

- ▶Set up a Jabber account
- ▶Test the account with the Exodus Jabber client

The Exodus message window

1. Close Exodus.

Using XMLSocket in Flash Lite 2.1

The `XMLSocket()` object is not initially connected to any server. You must create the object, establish a connection, then pass and receive XML data.

Using the XMLSocket class

- Flash Lite 2.1 supports socket connections through the `XMLSocket()` class. XML sockets are a good choice for low-latency, near-continuous communication applications, such as chat clients.

Using the XMLSocket class methods

;

`.connect(url, port)`	**open a persistent connection to a socket server running at the specified URL and port number**
`.onConnect(success)`	an event broadcast with a `true` or `false` argument when a connection is made or fails
`.send(xml)`	sends an XML `String` or `XML` object to the connected socket server
`.onXML(xml)`	event broadcast with `XML` object argument holding data returned from the socket server
`.close()`	Closes the XML socket

Connecting to the socket server

Once you have created an `XMLSocket()` object, establish a socket connection by invoking its `connect()` method, passing two parameters: the IP address or domain of the target server, and the port number to which you'll connect.

```
var server:String = "jabber.com";
var socket:XMLSocket = new XMLSocket();
socket.connect(server, 5222);
```

Connecting to the socket server

The `XMLSocket()` object's `onConnect` event will broadcast when the attempted connection succeeds, or fails. A `Boolean` parameter will be passed to an assigned event handler, with the value `true` if successful, or `false` if not.

```
var server:String = "jabber.com";
var socket:XMLSocket = new XMLSocket();
socket.connect(server, 5222);
socket.onConnect = Delegate.create(this, doConnect);
function doConnect(connected:Boolean):Void
{
 if(connected){
  // socket connected, start sending
 }
 else{
  // failed to connect
 }
}
```

Disconnecting from the socket server

To conserve system overhead, and keep the client more secure, close the socket object once it's no longer in use. The `close()` method takes no parameters, and fires an `onClose` event when the socket has been closed.

```
socket.close();
socket.onClose = function():Void
{
 trace("connection closed");
}
```


Adobe

Walkthrough 2: Creating an XMLSocket connection

In this walkthrough, you will perform the following tasks:

- Create a new `XMLSocket()` object
- Connect to the jabber.com XML socket server

Passing XML Data through Jabber

- Flash clients can connect to the Jabber server as a normal Jabber client, and speak the normal Jabber XML protocol.
- Jabber also supports "transports" which allow you to extend the capabilities of the server to include AIM, IRC, Yahoo, and MSN protocols, allowing your Flash clients to communicate to anyone on these networks.

Identifying as a Flash based Jabber client

In order to tell the Jabber server you are using a Flash client, you must pass it this XML document after connecting ,using the `send()` method of the `XMLSocket()` object.

```
<?xml version="1.0"?>
<flash:stream to="[Jabber Server]" xmlns="jabber:client"
  xmlns:flash="http://www.jabber.com/streams/flash"
  version="1.0"/>
```

Sending data through XMLSocket

The `send()` method of the XMLSocket class passes either an `XML` object, or an `String` of valid XML data, to its connected server. An asynchronous response will be received by any assigned `onData` or `onXML` event handers.

Adobe

Sending an XML String

A `String` of valid XML - in a format required by the connected server - may be passed to the `send()` method of the `XMLSocket` object as an argument.

```
var socket:XMLSocket = new XMLSocket();
... // connect and get response

socket.send("<?xml version='1.0'?>
  <login username='Fred' password='123 />");
```

Receiving data through XMLSocket

When XML is received through a connected `XMLSocket` object, two events are fired: `onData` and `onXML`. The `onData` event fires first, before the data is formatted into an `XML` object. The `onXML` event fires second, after the returned data is parsed into an `XML` object.

Using the onData event

- The `XMLSocket.onData` event fires when raw data has been pushed from the server, terminated by a zero (0) byte.
  - By default, the returned data will be parsed into an XML object. Once this is done, the `XMLSocket.onXML` event fires.
  - You can override the default `XMLSocket.onData` event handling, by implementing a custom event handler to intercept data before it is parsed as XML. This is a useful if you're transmitting arbitrarily formatted data packets, and you'd prefer to manipulate the data directly when it arrives, rather than have Flash Lite parse the data automatically. It is also useful if you wish to view the returned data for testing and debugging purposes.

Using the onXML event

The `XMLSocket.onXML` event fires when the data pushed from the server has been parsed as XML. An event handler assigned to this event will receive one parameter, an `XML` object created within Flash Lite, from the XML string returned through the `XMLSocket` object.

```
socket.onXML = Delegate.create(this, onSocketData);
function onSocketData(p_response:XML):Void
{
 // p_response is an XML object, built from
 // the XML String passed to the onData
 // event handler
}
```

Walkthrough 3: Sending and Receiving XML Data

In this walkthrough, you will perform the following tasks:

- Let the Jabber server know it is talking to a Flash client
- Handle and view data pushed from the server

Assembling XML data

- The XML data to be sent may become verbose. It can be cumbersome to work with long String variables in code.
- Two approaches make working with long XML packets more manageable.

Using compound String concatenation

XML protocols will often require that long String values have user-provided values, contained in variables, be written into the String. Compound concatenation, using the append operator (+=) may make this easier.

```
var id:Number = 101;
var price:Number = 450.00;
var bikeName:String = "VooDoo";

var xml:String = "<bikes>";
xml += "<bike id='" + id + "'>";
xml += "<price>" + price + "</price>";
xml += "<bikeName>" + bikeName + "</bikeName>";
xml += "</bike>";
xml += "</bikes>";

trace(xml); // displays "<bikes><bike id='101'>
  <price>450.00</price><bikeName>VooDoo</bikeName></bike></
  bikes>"
```

Using the XML and XMLNode API

An XML objects attributes and nodes may be generated through a series of method calls on the XML object.

Creating elements from the XML object

There are two ways to create new `XMLNode` objects ("elements"). You may either use the `createElement(name)` and `createTextNode(value)` methods of an XML object. Or, you may use the `XMLNode()` class constructor.

Regardless of how each node is created, each must be added to its parent element, or the `XML` object itself if it is the top-level node, using the `appendChild()` method

Building an XML object from the XML object

The `createElement(name)` and `createTextNode(value)` methods of the `XML` object may be used to create each node. Each node is then placed inside its parent node using the appendChild(node) method of its parent.

```
var xml:XML = new XML();
var root:XMLNode = xml.createElement("bikes");
var bike:XMLNode = xml.createElement("bike");
bike.attributes.price = 450.00

var bikeName:XMLNode = xml.createTextNode("VooDoo");

bike.appendChild(bikeName);
root.appendChild(bike);
xml.appendChild(root);
trace(xml);
// displays
// <bikes><bike price="450"> VooDoo</bike></bikes>
```

Building an XML object using the XMLNode constructor

- `XMLNode` objects can be explicitly created, and appended as desired to other `XMLNode` objects, to assemble an XML object.
- The first argument passed to the constructor will be `1` for an Element and `3` for a Text node. The second argument will be the name of the node.

Building an XML object using the XMLNode constructor

Nodes are appended to their parent using the `appendChild(node)` method.

```
var xml:XML = new XML();
var root:XMLNode = new XMLNode(1, "bikes");
var bike:XMLNode = new XMLNode(1, "bike");
bike.attributes.price = 450.00

var bikeName:XMLNode = new XMLNode(3, "VooDoo");

bike.appendChild(bikeName);
root.appendChild(bike);
xml.appendChild(root);

trace(xml);
// displays
// <bikes><bike price="450"> VooDoo</bike></bikes>
```

Walkthrough 4: Sending and Receiving Messages

In this walkthrough, you will perform the following tasks:

- ▶Capture the user input for username and password.
- ▶Assemble XML login information using compound String concatenation
- ▶Send, and handle the login response from the server

Adobe

Summary

- A socket connection remains open for two-way communication, unlike HTTP and FTP connection, which close once complete
- Jabber.com is an XML socket server supporting real-time chat
- XML socket communication is enabled in Flash Lite 2.1 (not 2.0) through the `XMLSocket` class
- An XML socket connection is created using the `connect(url, port)` method of an `XMLSocket` object
- An XML socket connection is closed using the `close()` method of the `XMLSocket` object
- A `onConnect(success)` event broadcasts once the connection is made
- XML is sent by passing either an XML `String` or an `XML` object to the `send(xml)` method of a connected `XMLSocket` object
- When XML data is returned both an `onData(data)` and `onXML(xml)` event are broadcast

Summary

- The `onData(data)` event is passed the raw, unformatted returned data as an argument
- The `onXML(xml)` event is passed the returned data as an argument, formatted as an `XML` object
- Long XML packets can be assembled through either compound `String` concatenation, with variables, or through the `XML` and `XMLNode` class API's
- `XMLNode` objects can be either directly created, or created using the `createElement(name)` or `createTextNode(text)` methods of the `XML` class