Flex Camp Boston

7th December 2007

Who am I?

Peter Farland Flex SDK

Background:

Java development with a focus on application servers, data services and various protocols, true type fonts, the SWF file format and a bit of vector graphics

Products:

Developer on several Allaire/Macromedia/Adobe products: ColdFusion, Flash Remoting, Flex, Flex Data Services (FDS), LiveCycle Data Services (LCDS)

Overview

1. Versions of Flex and LCDS
2. Configuration tips for LCDS
3. Debugging Flex and LCDS applications
4. Deployment considerations for LCDS

Versions

A general guide

“For a given LCDS server, use the Flex SDK released at that time, otherwise, a later Flex SDK.”

Compile-Time Compatibility

rpc.swc	fds.swc
Flex 2.0.1	FDS 2.0.1
Flex 2.0.1 HF2	LCDS 2.5
Flex 2.0.1 HF3	LCDS 2.5.1
Flex 3	LCDS 2.5.1

- Compile-time compatibility depends on rpc.swc and fds.swc present in the library path, and the compiler's awareness of the services-config.xml file format
- Don't forget to add the fds_rb.swc resource bundle to the locale directory too

Runtime Compatibility

Client	Server
Flex 2.0.1	FDS 2.0.1
Flex 2.0.1 HF2	LCDS 2.5
Flex 2.0.1 HF3	LCDS 2.5.1
Flex 3	LCDS 2.5.1

- Runtime client-server compatibility depends on the messaging format exchanged between Flex and LCDS, controlled by the client channels, the server endpoints and the message broker
- The LCDS requirements generally apply to the advanced Data Management feature. RPC services like remoting and the proxy will work with more client-server configurations

Determining Versions

SWCs

- SWCs are zip archives and can be expanded with WinZip, UnZip, Stuffit Expander, etc...
- The <versions> section of catalog.xml reports the build of compc that was used to build the swc
- Recently, we've added a version.properties to our swc archives to specify the exact build number when the swc was built

Determining Versions

JARs

- JARs are also zip archives and can be expanded
- The Implementation-Version entry in the MANIFEST.MF file reports the build in which the JAR was built
- Some of the Flex SDK JARs report the build information in a version.properties file, but going forward the MANIFEST.MF will be the place to look for version information

JVM Versions

A general guide
"Test several updates of Java for performance differences. When in doubt, try to use the latest update of Java."

Java 1.4.2

- Certain Java 1.4.2 updates have IO problems which seriously affect Flex compiler performance and cause issues for RTMP in LCDS... avoid 1.4.2_02 through 1.4.2_05, and 1.4.2_10

Java 5 (1.5.0)

- Avoid 1.5.0_07
- xalan.jar greatly speeds up services-config.xml parsing

Configuration

- The Flex compiler is aware of the LCDS services-config.xml file format due to /lib/flex-messaging-common.jar
- The –services command line configuration option generates AS3 code to:
  - embed a subset of the services-config.xml as XML to support destination channels, clustering and managed properties
  - ensure Channel classes are linked-in to the SWF and initialized
  - ensure DataService lazy associations are registered

Compiling without "-services"

- Flex APIs RemoteObject, proxied WebService or HTTPService, you do not need to specify a –services command line option if you programmatically create a ChannelSet

```
import mx.messaging.ChannelSet;
import mx.messaging.channels.AMFChannel;

...

private function applicationInit(event:Event):void
{
    var channelSet:ChannelSet = new ChannelSet();
    var channel:AMFChannel = new AMFChannel(null,
        "http://{server.name}:{sever.port}/myapp/messagebroker/amf");
    channelSet.addChannel(channel);
    myRemoteObject.channelSet = channelSet;
}
```

ChannelSet Usages

- Multiple Channels in a ChannelSet simply provide a failover mechanism (see later)
- You should try to use a single ChannelSet for all services in your client – this simplifies failover, server session state, etc.
- You should only authenticate once per client (and you should use ChannelSet.setCredentials() to authenticate instead the service setCredentials() APIs)
- You can use the SWFs URL to determine whether a secure channel should be used at runtime (e.g. if https, use SecureAMFChannel; if http, use AMFChannel)

Poor Man's Load-Balancing

- The general idea is to programmatically create a ChannelSet that contains a list of failover URLs but the order is randomized on creation
- You can use Math.random() to help create a random list

LCDS Runtime Configuration

- Each "component" of the LCDS message broker can be programmatically created (instead of being statically declared in services-config.xml)
- You should be able to use the Javadoc for LCDS and create your own bootstrap service that dynamically creates new destinations on startup
- For example, you may want to read a hibernate configuration file and determine what data management service destinations you need to create
- A non-bootstrap example may be that you want to create a Chat service and dynamically create rooms for private conversations (see LCDS samples for example code)

Optimize Configuration

- Be sure to remove unused services, adapters, destinations, channel-definitions, security-constraints, login-adapters, clustering settings, redeploy watcher settings, etc... from services-config.xml
- For deployment, be sure to remove redeploy watchers and turn off debugging level logging
- Smaller configurations are easier to manage and speed up compilation
- Configuration parsing is based on XPath - this has proven to be slower on JDK 1.5.0 (xalan.jar in the classpath should resolve this issue)

Incremental Compilation

- Flex Builder can use active or inactive incremental compilation depending on the number of projects in your workspace
- Typically the last edited project is retained in memory (active), other projects are cached to disk (inactive)
- There's some overhead in restoring the incremental compilation state from the cache, so the fastest compilation scenario is active incremental compilation

Debugging

- For Flex applications, the Debugger and Profiler are great tools for tracking down problems
- ...but for data drive applications, it's a little more involved
- There are considerations for compile time, and at runtime for both the client and server
- The most common compile-time mistake is not having references to data types retrieved at runtime and thus not linked in to the SWF
- The situation is harder to spot when the missing data type implements flash.utils.IExternalizable (the infamous "one of the parameters is incorrect" error)

Client Debugging

- For data services, a quick way to get debug info is to add <mx:TraceTarget /> to your MXML and watch the Flex Builder debug console (or flashlog.txt with a debug Flash Player)
- Log output is based the mx.logging.Log API which reports events to listening targets – if you programmatically create targets, you can do more such as filter on categories etc.
- To capture raw HTTP information use a client side sniffer like Paros Proxy, Charles, ServiceCapture, etc.
- The benefit of a client side sniffer like Paros Proxy is that you can also debug HTTPS requests

Server Debugging

- On the server, /WEB-INF/flex/services-config.xml has a section to control logging. Set the target logging level to "Debug" and adjust the categories to log
- There are also server side HTTP sniffers (see LCDS's embedded JRun for /bin/sniffer.exe)
- You could write your own implementation of flex.messaging.log.Target, but typically it's easiest to start your J2EE server on the command line and rely on the default ConsoleTarget.

Browser Issues

- Be on the look out for known issues with certain browsers:
  - HTTPS responses with no-cache headers cause issues with MSIE 6
  - HTTP responses with no-cache headers with chunked and gzip encoding can cause issues
  - Response bodies are returned as empty when non 200 HTTP status-codes are received

Deployment

- For deployment, you should pre-compile your application and deploy SWFs instead of MXML/AS source
- ...so you don't need to deploy the Flex Webtier compiler with LCDS WAR files
- From /WEB-INF/web.xml, remove:
  - the flex.class.path <context-param>
  - the FlexMxmlServlet, FlexSwfServlet, and FlexInternalServlet <servlet>'s and any related <servlet-mapping>'s
  - the FlexTagLib <taglib>
- From /WEB-INF/flex, delete /jars, /libs, /locale, *.ser, *.css, flex-*.xml, flash-*.xml, mxml-manifest.xml
- From / WEB-INF/lib, delete bootstrap*.jar

Secure Channels and Firewalls

- With LCDS 2.5.1, you can configure secure client channels and normal server endpoints. The client connects securely to your firewall but requests inside the DMZ can be normal HTTP requests
- This works because normal endpoints can still accept secure requests

```
<channel-definition id="my-secure-amf"
  class="mx.messaging.channels.SecureAMFChannel">
    <endpoint url="https://..."
      class="flex.messaging.endpoints.AMFEndpoint"/>
</channel-definition>
```

RTMP Channel Port Binding

- For an RTMP endpoint you can configure the <bind-address> and <bind-port> to let the endpoint bind to a different address:port than the client channel
- This lets you point clients at an external TCP load-balancer that forwards connections to the bind address:port of the RTMP endpoint

Virtual Hosting

- If your deployment is on a virtual host, be mindful of the resources available for your endpoints
- On Linux, socket handles come out of the same fixed pool of file handles (along with file handles, pipes, etc)
- Typically web hosting companies have things fine-tuned for traditional web applications that use short-lived connections – a gotcha you have to watch out for...

Final Notes

- Keep up-to-date with the SDK nightly builds on Adobe Labs
- Make noise - log bugs at http://bugs.adobe.com/flex
- Flex community – http://www.flex.org/community
- This presentation will be up on my blog at: http://blogs.adobe.com/pfarland

Thank you!