

Calling Remoting Service destinations from Flash or Java applications

Note: This content is an addition to [Using the Remoting Service](#).

The NetConnection API of Flash Player provides a way to call Remoting Service destinations from a standard (non-Flex) Flash application or from ActionScript in a Flex application if desired. The AMFConnection API in LiveCycle Data Services ES gives you a Java API patterned on the NetConnection API but for calling Remoting Service destinations from a Java application. You can use either of these APIs with LiveCycle Data Services ES, BlazeDS, or third-party remoting implementations.

Call a destination from a Flash application

You can use the Flash Player `flash.net.NetConnection` API to call a Remoting Service destination from a Flash application. You use the `NetConnection.connect()` method to connect to a destination and the `NetConnection.call()` method to call the service. The following MXML code example shows this way of making Remoting Service calls with `NetConnection` instead of `RemoteObject`:

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%"
  creationComplete="creationCompleteHandler();" >
  <mx:Panel id="mainPanel" height="100%" width="100%">
    <mx:HBox>
      <mx:Label text="Enter a text for the server to echo"/>
      <mx:TextInput id="ti" text="Hello World!"/>
      <mx:Button label="Send" click="echo()"/>
      <mx:Button label="Clear" click='ta.text = ""'/>
    </mx:HBox>
    <mx:TextArea id="ta" width="100%" height="100%"/>
  </mx:Panel>
  <mx:Script>
    <![CDATA[
      import flash.net.NetConnection;
      import flash.net.ObjectEncoding;
      import flash.net.Responder;
      private var nc:NetConnection
      private function creationCompleteHandler():void {
        // Create the connection.
        nc = new NetConnection();
        nc.objectEncoding = ObjectEncoding.AMF0;
        // Connect to the remote URL.
        nc.connect("http://[server]:[port]/yourapp/messagebroker/amf" );
      }
      private function echo():void {
        // Call the echo method of a destination on the server named remoting_AMF.
        nc.call( "remoting_AMF.echo", new Responder( resultHandler, faultHandler ),
          ti.text );
      }
      private function resultHandler(result:Object):void {
        ta.text += "Server responded: " + result + "\n";
      }
      private function faultHandler(fault:Object):void {
        ta.text += "Received fault: " + fault + "\n";
      }
    ]]>
  </mx:Script>
</mx:Application>
```

Call a destination from a Java application

The AMFConnection API is a new Java client API in the flex-messaging-core.jar file that makes it possible to work with Remoting Service destinations from a Java application. To compile Java classes that use the AMFConnection API, you must have both the flex-messaging-core.jar and flex-messaging-common.jar files in your class path.

The AMFConnection API is similar to the Flash Player flash.net.NetConnection API, but uses typical Java coding pattern rather than ActionScript coding pattern. The classes are in the flex.messaging.io.amf.client* package in the flex-messaging-core.jar file. The primary class is the AMFConnection class. You connect to remote URLs with the AMFConnection.connect() method and call the service with the AMFConnection.call() method. You catch ClientStatusException and ServerStatusException exceptions when there are errors. Here is an example of how you can use AMFConnection to call a Remoting Service destination from a method in a Java class:

```
public void callRemoting()
{
    // Create the AMF connection.
    AMFConnection amfConnection = new AMFConnection();

    // Connect to the remote URL.
    String url = "http://[server]:[port]/yourapp/messagebroker/amf";
    try
    {
        amfConnection.connect(url);
    }
    catch (ClientStatusException cse)
    {
        System.out.println(cse);
        return;
    }

    // Make a remoting call and retrieve the result.
    try
    {
        // Call the echo method of a destination on the server named remoting_AMF.

        Object result = amfConnection.call("remoting_AMF.echo", "echo me1");
    }
    catch (ClientStatusException cse)
    {
        System.out.println(cse);
    }
    catch (ServerStatusException sse)
    {
        System.out.println(sse);
    }
    // Close the connection.
    amfConnection.close();
}
```

The AMFConnection API automatically handles cookies similarly to the way in which web browsers do, so there is no need for custom cookie handling.