



# METAOPTION

A Quick Article on Flex Architecture and  
Programming  
[By: Flex Team METAOPTION]

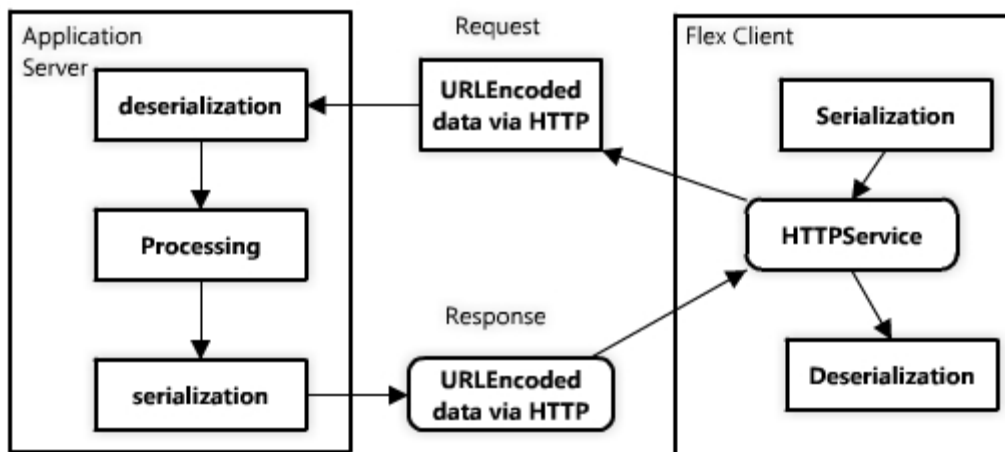
## What is Flex

Flex is a highly productive, free open source framework for building and maintaining expressive web applications that deploy consistently on all major browsers, desktops, and operating systems. It provides a modern, standards-based language and programming model that supports common design patterns. MXML, a declarative XML-based language, is used to describe UI layout and behaviors, and ActionScript™ 3, a powerful object-oriented programming language, is used to create client logic. Flex also includes a rich component library with more than 100 proven, extensible UI components for creating rich Internet applications (RIAs), as well as an interactive Flex application debugger.

## Flex Architecture Fundamentals - Main Means of Communication between Flex and the Application Server: -

### HTTPServices

This class makes HTTP(S) requests (mostly GET or POST) which may transport various content types, from simple URLEncoded variables to more complex data like XML. Basically, this is the choice for Representational State Transfer (REST) web services architecture. Most of the time, an HTTPService object is used to communicate with a simple script or page like a JSP page, an ASP page or a PHP script. Obviously, the value of the HTTPService url property is the URL of this page



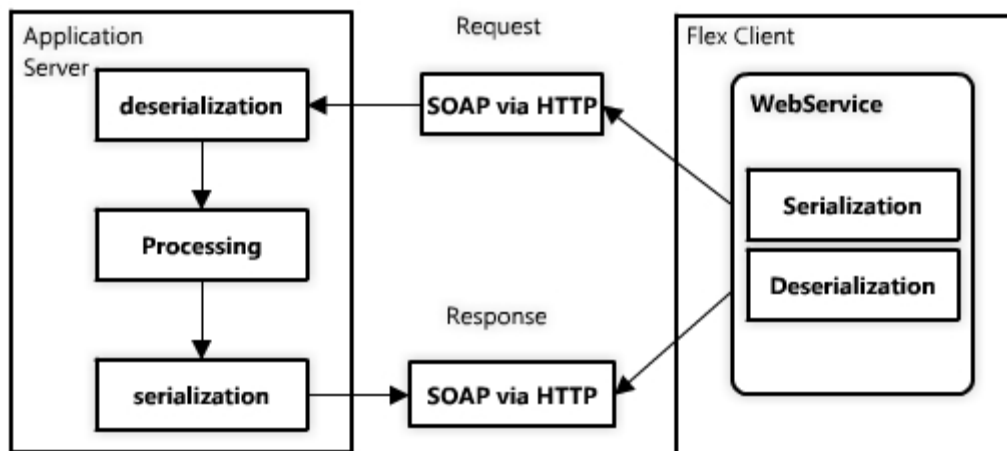
The transported data is sent and received as plain text, meaning that we have to map the received data to the corresponding ActionScript data type on the Flex side, and to the corresponding server-side technology data type on the server side.

## WebServices

The `WebService` class sends and receives SOAP messages over HTTP. Naturally, this is what we use to communicate with SOAP WebServices. To connect to this `WebService` and call its operations (i.e. its methods), you'll have to hook this object to the Web Service's Web Service Definition Language (WSDL) by setting its URI as the `wsdl` property value.

All ActionScript primitive types and some built-in ActionScript complex types are automatically mapped from and to SOAP/XMLschema data. This process is handled on the Flex side, by the `WebService` class. This is quite an improvement over `HTTPService`-based communications where we have to do that job ourself, or use third party libraries.

However, you'll still have to implement your own mapping procedure for custom classes.



## RemoteObjects

The `RemoteObject` class is responsible for sending and receiving ActionScript Message Format (AMF) data. This technique is also called Remoting.

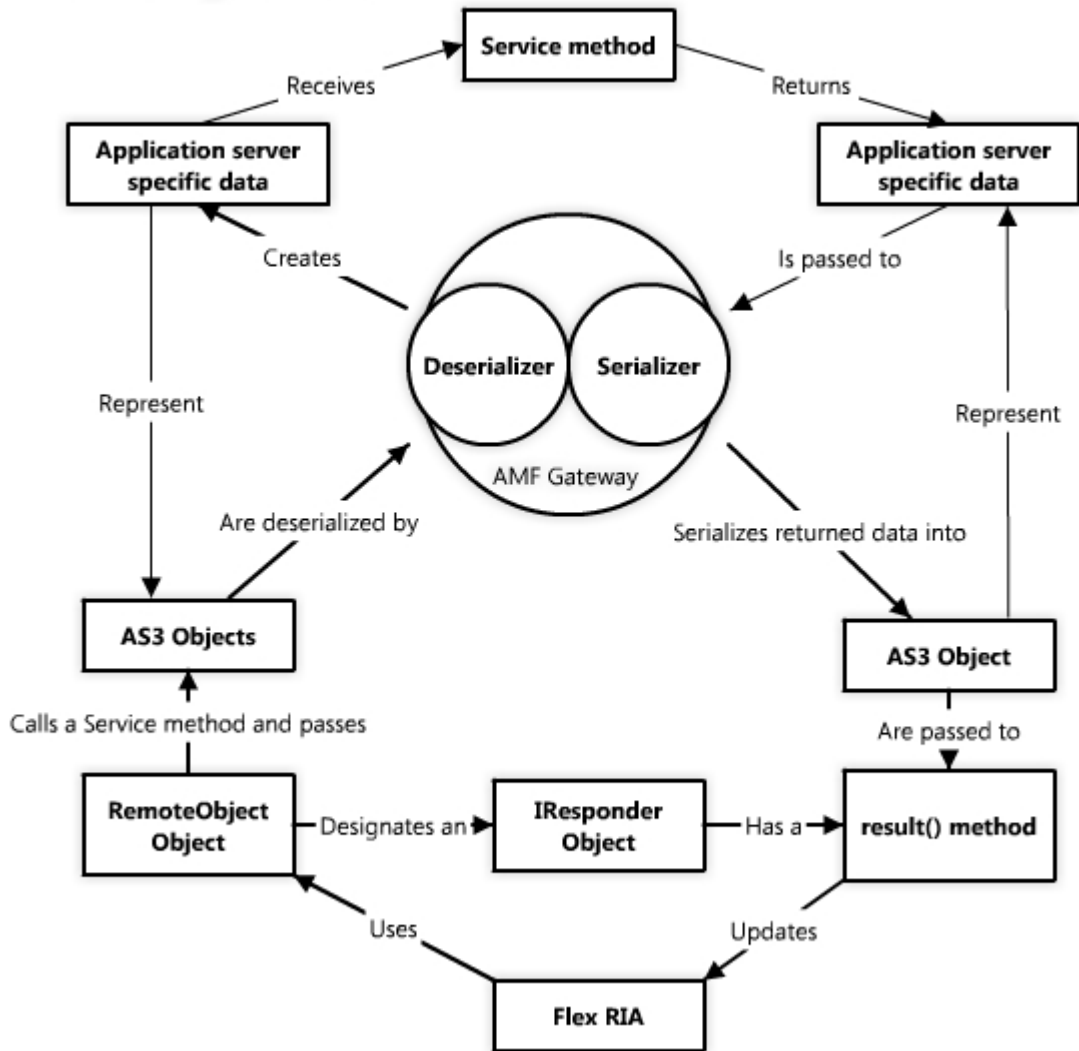
AMF is not a transfer protocol: it is binary ActionScript. AMF3 is the binary format for AS3, whereas AMF0 was the binary format for AS1 and AS2. Note that AMF3 specifications were open quite recently.

In this case, AMF is transported over HTTP, but it can also be transported over real time communication protocols. Exchanging data using AMF is dramatically faster than any other mean of communication. Take the test: try James Ward's benchmark application.

Besides, it allows the Flex client to send and receive ActionScript strongly typed objects. With Remoting, the ActionScript serialization/deserialization process takes place on the server side. Of course, you'll have to have a server side tool that enables this AMF serialization/deserialization: this is what we call an AMF Gateway (or Remoting Gateway).

With Remoting, you can transport built-in primitive and complex data (as with a Webservice) but also custom typed objects. This way, you're able to automatically map a server side Value Object to and from an ActionScript Value Object. To do so, you simply have to add a RemoteClass Meta data to your ActionScript ValueObject Class Definition.

### The Remoting data flow



### Flex Programming Environment:-

The Programming Environment that we are using is Adobe Flex Builder 3. It accelerates Flex application development. It is an Eclipse based development tool enabling intelligent coding, interactive step-through debugging, and visual design of the user interface layout, appearance, and behavior of RIAs. Flex Builder 3 includes the complete Flex framework, including compilers, a component library, and debuggers.

Flex Builder 3 is available in Standard and Professional editions. Both editions allow import of assets from Adobe Creative Suite® 3 software, making it easy for designers and developers to work together. Flex Builder 3 Professional further adds powerful data visualization capabilities, the new Advanced Datagrid, memory and performance profilers, and support for automated functional testing for developing business-critical applications. Key features include:

1. Powerful coding tools- Flex Builder 3 is a powerful Eclipse based IDE that includes editors for MXML, ActionScript™, and CSS, as well as syntax coloring, statement completion, code collapse, interactive step-through debugging, and more.
2. Rich visual layout (Enhanced in Flex Builder 3) - Visually design and preview user interface layout, appearance, and behavior using a rich library of built-in components. Extend the built-in components or create new ones as needed.
3. Interactive data visualization (Enhanced in Flex Builder 3) (Professional edition only)
4. Create data dashboards and interactive data analysis by simply dragging and dropping a chart type and linking it to a data source using the Flex charting library. Use the powerful new Advanced Datagrid to enable users to explore complex data.
5. Skinning and styling - Customize the appearance of an application using CSS and graphical property editors. Quickly set the most commonly used properties and preview the results in Design view.
6. Integration with Adobe Creative Suite 3 - New Flex Skin Design Extensions for Adobe Flash®, Illustrator®, Photoshop®, and Fireworks® make it fast and easy to import ready-to-use creative assets from Adobe Creative Suite® software directly into Flex Builder 3. Use the new Adobe Flex Component Kit for Flash CS3 Professional to create interactive, animated content in Flash that can then be exported as a Flex component.
7. Code refactoring - The new Flex Builder 3 refactoring engine allows developers to quickly navigate through code or restructure code by renaming all references to a class, method, or variable.
8. Native support for Adobe AIR - Flex Builder 3 provides the fastest way to create applications for the Adobe AIR™ runtime, including all the tools required to build, debug, package, and sign Adobe AIR applications. Adobe AIR lets you quickly develop RIAs for the desktop using the same skills and codebase you use to build RIAs for the browser.
9. Powerful testing tools -The Flex Builder 3 memory and performance profilers enable developers to improve application performance by providing tools to monitor and analyze memory consumption and CPU cycles. Support for automated functional testing tools such as HP QuickTest Professional (formerly Mercury QuickTest Professional) is also now available in Flex Builder 3 Professional.

10. Web service introspection -Flex Builder 3 can now retrieve a WSDL and generate ActionScript proxies to make calls and serialize/deserialize strongly typed objects. New code hinting is available for making web service calls and dealing with responses.

### **Data Access and Interconnectivity:-**

I) Accessing Server-Side Data with Flex Adobe® Flex® data access components use remote procedure calls to interact with server environments, such as PHP, Adobe ColdFusion, and Microsoft ASP.NET, to provide data to Adobe Flex applications and send data to back-end data sources. For this the following ways can be used:-

1. Using HTTPService components – This can be used as -  
`<mx:HTTPService id="userRequest"  
url="http://server/myproj/request_post2.php" useProxy="false"  
method="POST">`
2. Using WebService components - This can be used as -  
`<mx:WebService id="userRequest"  
wsdl="http://localhost:8500/flexapp/returnusers.cfc?wsdl">`
3. Using RemoteObject components - This can be used as -  
`<mx:RemoteObject id="userRequest" destination="ColdFusion"  
source="flexapp.returnusers">`
4. Explicit parameter passing and parameter binding -This can be used as -  
`var params:Object = new Object();  
params.param1 = 'val1';  
myService.send(params);`
5. Handling service results - This can be used as -  
`<GetQuoteResponse  
xmlns="http://ws.invesbot.com/">  
<GetQuoteResult><StockQuote xmlns="">  
<Symbol>ADBE</Symbol>`

II) Representing Data -Data representation is a combination of features that provide a powerful way to validate, format, store, and pass data between objects.

Adobe® Flex® provides the following set of features for representing data in our applications: data binding, validation, and formatting. These features work in conjunction

with the Adobe® LiveCycle™ Data Services ES and Vega features for working with remote data. Together, they allow us to perform the following tasks:

1. Pass data between client-side objects.
2. Store data in client-side objects.
3. Validate data before passing it between client-side objects.
4. Format data before displaying it.

III) Binding Data -Data binding lets one pass data between client-side objects in an Adobe® Flex® application. Binding automatically copies the value of a property of a source object to a property of a destination object when the source property changes.

IV) Storing Data -We use the data model feature to store data in an application before it is sent to the server, or to store data sent from the server before using it in the application. A data model is an ActionScript object that contains properties that you use to store application-specific data. Communication between an Adobe® Flex® application and the server is required only to retrieve data not yet available to the Flex application and to update a server-side data source with new data from the Flex application.

V) Validating Data - We use the Adobe® Flex® data validation mechanism to validate the data in an application. Flex provides predefined validators for many common types of user-supplied data, such as date, number, and currency values.

VI) Formatting Data - Data formatters are user-configurable objects that format raw data into a customized string. We can use formatters with data binding to create a meaningful display of the raw data bound to a component. This can save our time by automating data formatting tasks and by letting you easily change the formatting of fields within our applications.

### **Benefits of using Flex:-**

1. Eliminate the "busy work" associated with the existing business process. We needed to find a development platform that could help us come close to performing a data visualization miracle! The existing process (see Figure 1) was not only labor intensive and inefficient, it also left room for human error because reports were generated manually by re-keying financial information into a static presentation.
2. Integrate into our existing processes and technical systems. Standards are good! That is why millions of dollars are spend by various industries to create and adhere to them. Flex is not a back-end development platform. Flex is a technology used to render the "view" or "presentation layer" of RIAs. Making Flex applications work within the existing environment is a relatively painless process,

- because Flex is designed to allow standards-based communications between the Flex applications and other systems.
3. Run the scorecard on a variety of devices. Perhaps the most obvious way that our use of Flex meets a project requirement is that Flex applications are executed within Macromedia Flash Player. As we know, Flash Player has evolved a great deal since the days of overwhelming "skip intro" animations. For the sake of our project we consider Flash Player as a cross-platform runtime engine for RIAs. To run Flex applications, the client must have Flash Player 7.x installed.
  4. Provide real-time access to our most current data. Although this requirement is partially addressed in the second bullet item above, we didn't really touch on how this functions technically and what advantages we have gained by using Flex. While there are a wide variety of applications that enable real-time data reporting, none is as flexible when it comes to the presentation layer as an RIA running in the Flash client runtime. The advantages that Flex brought to us in this area are found in the myriad of ways that Flex applications can send, receive, and process data on the client.
  5. Be secure. The entire Executive Information System (EIS) dashboard that we've been talking about in this overview is designed to present critical business data to users that have the proper authentication credentials. Some users, such as the company CEO and other executives, have access to view all the data that the system offers. Other users can view only data that is relevant to their geographic location or their practice area.

In short, the system provides role-based security that can be customized to accommodate different types of users. While our Flex application doesn't manage the user authentication process, it does manage the content displayed to the user based on security rules delivered to the Flash interface from our back-end system. The advantage Flex provides in this regard is obvious: We can easily change user views of data and screen elements based on rules received from the application security model.