iControl Concept to Implementation (iC2I): The Introduction.

Don MacVittie, 2008-04-04

iControl is built upon the theory of granularity – the idea that small routines, each of which does a single atomic or nearly atomic task – will give you maximum flexibility. And it does. You can do quite a bit with iControl that can't be found elsewhere in the product because other portions of the product (like the UI) use several iControl calls lumped together to complete a task.

But granular means "product or technology focused", not "business or task focused". This is not necessarily a bad thing – we can't specialize for your business if we want to sell to many businesses – but it can mean there is a learning curve to your development. The Java Wrappers project aims at removing some of the extraneous technical overhead and giving you streamlined access to objects in the BIG-IP, but sometimes that's not enough.

So this series of articles will focus on a single business requirement or system administration task and follow through achieving it in iControl. That will mean some articles in the series will be short Tech Tips, and some will be longer Tutorials, but all will provide you with some solid steps to pursue to achieve a given goal. All but this introductory article that is. This article will be the only bit of required reading for the series, and is going to go over some basic items that are common to every iControl program, and make certain you have those down before doing anything else.

The first round of articles in this series will be relatively simple tasks that are common to all BIG-IP installations, and as time goes on we'll delve into more and more complex situations. The process of writing these articles will generate code, that early on will be largely from the Java Wrappers project, and later on, where relevant, will be used to improve the Java Wrappers, meaning that you may be able to gain the benefit of older articles if you are a Java programmer simply by downloading the newest version of the Java Wrappers project.

But we're getting ahead of ourselves already. First we should set up those common functions that you must have to use iControl to begin with. Which really isn't much.

This series will share a few things, primarily the communications mechanism to work with the iControl API. So we'll build a base class that handles communications for all the other articles in this series. A pretty simple little class, but necessary to make things smoothly.

So the base class holds members that remember the information about the BIG-IP you wish to communicate with.

```
package iC2i;

public class BigIpBase {
    protected String username_ = null;
    protected String password_ = null;
    protected String address_ = null;
    protected String fullUrl = null;

    public BigIpBase(String username, String password, String address) {
        username_ = username;
        password_ = password;
        address_ = address;
        fullUrl = "https://" + username + ":" + password + "@" + address + "/iControl/iControlPortal.
    }
```

```
    public BigIpBase() { }
}
```

That's it for the base class, just a common method for handling communications. Thus whenever an instance of one of our classes is created it will be able to get in touch with the BIG-IP via what is set in this base class' constructor.

Of course there are times in any system when you want to create an instance that is not tied to a BIG-IP, that's a simple case and is also explicitly defined in the class.

The variable fullUrl just builds the SSL URL that we will use to communicate with iControl, telling the API your username and password, and calling the iControl Portal, the CGI script that is the interface to all API calls on the BIG-IP. If your username and password are invalid or don't have rights to some functionality, when you first call the BIG-IP you'll get an exception, but we'll handle that on a case-by-case basis.

Note that if you really wanted to get carried away with this class, you could create purely virtual methods for getList() and objectStatus(), since nearly all objects in iControl support them. To us this is overkill and removes our ability to give these generic names something more meaningful in the context of the object being referenced, so we won't, but it wouldn't hurt anything if you chose to.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com