

Accessing Data Across Subdomains

This document describes how a TIBCO General Interface™ application communicates with web servers across subdomains. There are two general scenarios for accessing a General Interface™ application, local provisioning and remote provisioning. Local provisioning requires no configuration, if default browser security settings are used. Remote provisioning requires either configuring proxy communication or modifying default browser security settings.

Version 4: May 9, 2007

Scope: TIBCO General Interface 3.4



<http://www.tibco.com>

Global Headquarters

3303 Hillview Avenue

Palo Alto, CA 94304

Tel: +1 650-846-1000

Toll Free: 1 800-420-8450

Fax: +1 650-846-1005

© 2006-2007, TIBCO Software Inc. All rights reserved. TIBCO, the TIBCO logo, The Power of Now, and TIBCO Software are trademarks or registered trademarks of TIBCO Software Inc. in the United States and/or other countries. All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

Table of Contents

Local Provisioning	3
Disabling Firebug in Firefox	3
Remote Provisioning.....	4
Configuring Proxy Communication	5
Configuring the Server	5
Configuring the Application	6
Modifying Internet Explorer Security Settings.....	8

Local Provisioning

With local provisioning, the application files and the TIBCO General Interface runtime framework are located on the same machine as the web browser. When the application is accessed, the browser loads required files into cache using a file URL, for example `file:\\C:\tibco\gi\GI_Builder.html`. With the default browser settings, there are no restrictions on communicating with web servers outside the subdomain. *Figure 1: Local Provisioning* illustrates the local provisioning scenario.

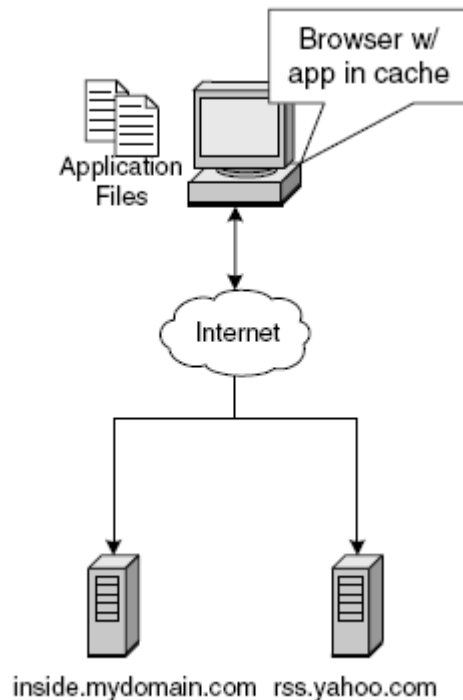


Figure 1: Local Provisioning

Disabling Firebug in Firefox

When using Firefox, cross-domain access is allowed when running from the file system. This allows easy testing of remotely provisioned services from within General Interface Builder. However, if the Firebug add-on is enabled for local files, a permission denied exception is thrown. To bypass this error, disable Firebug in Firefox (Tools > Firebug > Disable Firebug for Local Files).

Remote Provisioning

With remote provisioning, the deployed application files and the TIBCO General Interface runtime framework are located on a web server. A browser on a remote machine accesses the files using an HTTP URL and loads files into cache. When an HTTP URL is used, default browser security settings prevent your application from communicating with web servers outside the subdomain. *Figure 2: Remote Provisioning—Before Configuration* illustrates the remote provisioning scenario before any configuration is performed.

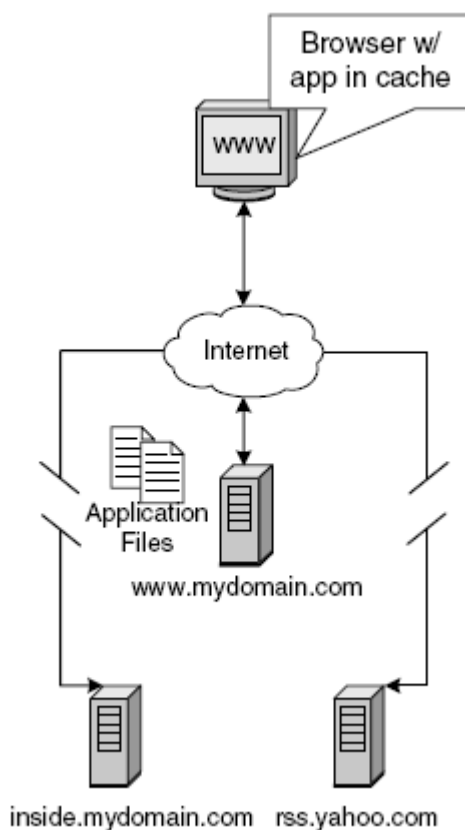


Figure 2: Remote Provisioning—Before Configuration

Figure 2 shows a TIBCO General Interface application deployed to `www.mydomain.com`. Without configuration, the application can only communicate with `www.mydomain.com`. The browser prevents communication with other subdomains, such as `inside.mydomain.com` and `rss.yahoo.com`. However, the same application in a local provisioning scenario can communicate directly with web servers outside the subdomain.

Configuring Proxy Communication

To enable proxy communication, you configure both the application and the web server where the application is deployed. This allows the web server to broker interactions between the application and web servers outside the subdomain. *Figure 3: Remote Provisioning—After Proxy Services Configuration* illustrates the remote provisioning scenario with proxy services configured.

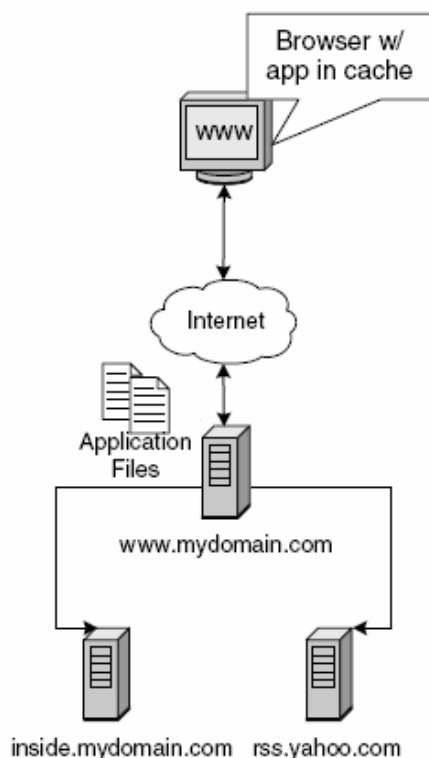


Figure 3: Remote Provisioning—After Proxy Services Configuration

Configuring the Server

Steps for configuring the web server vary according to vendor.

- **Apache** Install and configure the mod_proxy module. Then, use the ProxyPass directive to route responses through the TIBCO General Interface application domain server, such as `www.mydomain.com`, rather than a web server outside the subdomain. For details, see the Apache HTTP Server documentation at <http://httpd.apache.org>.
- **Microsoft IIS** Install and configure the Microsoft Internet Security and Acceleration (ISA) Server. Write an ISAPI Filter to perform reverse-proxying or use an equivalent third-party product. The software must intercept an HTTP Request made to the TIBCO General Interface application domain server, such as `www.mydomain.com`, copy the entire HTTP header packet, change the HTTP Host header and perform a new HTTP Request using the WinHttp API. For details, see: <http://www.microsoft.com/isaserver/default.mspx>.
- **BEA WebLogic** Register the proxy servlet in your Web Application deployment descriptor. Define an initialization parameter for the servlet and map the servlet to a URL pattern. For details, see: http://e-docs.bea.com/wls/docs81/plugins/http_proxy.html.
- **Custom** Write a server process, such as a servlet, .NET service, or web service, that acts as a proxy between the TIBCO General Interface application domain server and other subdomains.

Configuring the Application

There are two ways to retrofit a General Interface application so that it requests data from servers via a proxy. The simplest way is to modify any absolute URL in the project to point to the proxy URL. Absolute URLs may exist in many types of project source files including JavaScript, service mapping rules, and component serialization files. Use your favorite text editor to perform a global search and replace in the project source files for strings matching "http://" or "https://".

The second way to retrofit a General Interface application so that it requests data through a proxy is more complicated but provides a mechanism for easily switching back and forth between proxied and non-proxied requests. This is accomplished by defining a function that converts a non-proxied URL to a proxied URL and then modifying any request before it is sent to use the proxied URL. Consider the following code sample as shown in Figure 4.

Figure 4: Proxy Sample Code

```

jsx3.Package.definePackage("eg.proxy", function(proxy) {

    // switch, if true all URLs are converted to the proxied format
    proxy.PROXY = (window.location + "").indexOf("file") != 0;

    // the domain of my proxy host
    proxy.PROXY_HOST = "proxy.eg.com";

    // the path prefix of the proxied URLs
    proxy.PATH_PREFIX = "proxy/";

    /**
     * Converts a non-proxied URI to a proxied URI if PROXY is true.
     * <p/>
     * <code>http://www.domain.com/service/op</code> will be converted to
     * <code>http://PROXY_HOST/PATH_PREFIX/www.domain.com/service/op</code>
     *
     * @param strURI {String} the URI to convert
     * @returns {String} the converted URI
     */
    proxy.convertURI = function(strURI) {
        if (proxy.PROXY) {
            var uri = new jsx3.net.URI(strURI);
            if (uri.getHost() != proxy.PROXY_HOST &&
                (uri.getScheme() == "http" || uri.getScheme() == "https")) {
                return jsx3.net.URI.fromParts(
                    uri.getScheme(), null, proxy.PROXY_HOST, null,
                    proxy.PATH_PREFIX + uri.getHost() + uri.getPath(), null, null
                ).toString();
            } else {
                return strURI;
            }
        } else {
            return strURI;
        }
    };

    /**
     * Open all requests with this method to ensure that URLs are properly converted for proxy.
     */
    proxy.openRequest = function(strURL) {
        var objRequest = new jsx3.net.Request();
        objRequest.open("GET", proxy.convertURI(strURL));
        return objRequest;
    };

    /**
     * Open all services with this method to ensure that URLs are properly converted for proxy.
     */
    proxy.openService = function(strRulesURL, strOpName) {
        var objService = new jsx3.net.Service(strRulesURL, strOpName);
        objService.setEndpointURL(proxy.convertURI(objService.getEndpointURL()));
        return objService;
    };
});

```

Modifying Internet Explorer Security Settings

An alternative to configuring proxy services is to modify Internet Explorer security settings on each client machine that accesses the application. This option is only available for Internet Explorer. For Firefox, you must configure proxy communications. See [Configuring Proxy Communication](#).

When the security settings are set to accept the subdomain where the application is deployed as a trusted site, the browser allows direct communication with web servers outside the subdomain. *Figure 5: Remote Provisioning—After Browser Configuration* illustrates the remote provisioning scenario with modified browser settings.

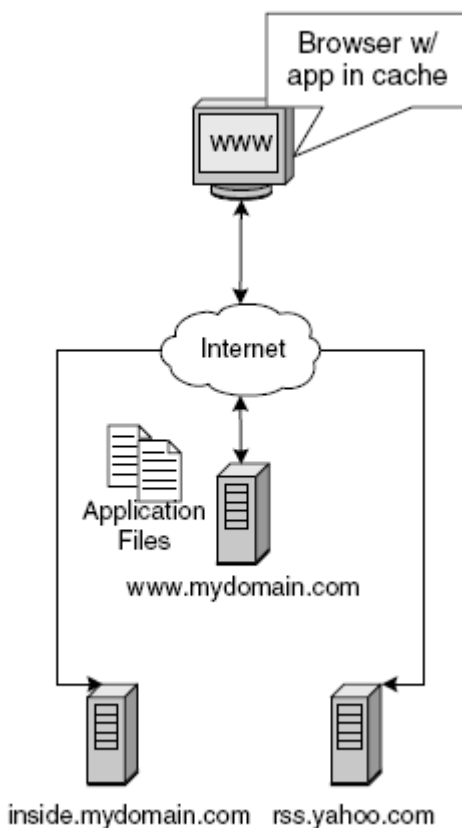
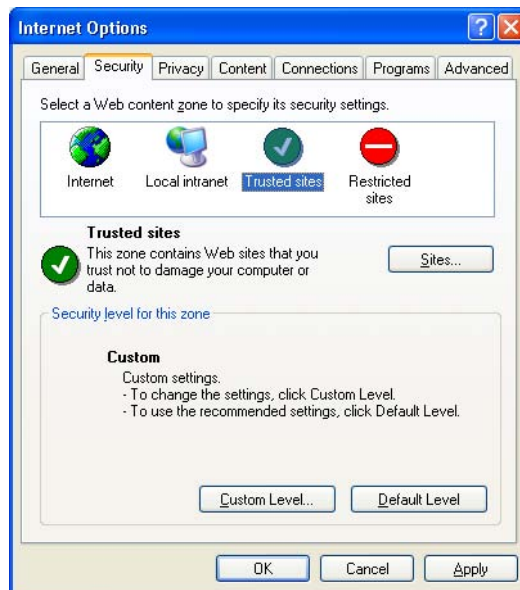


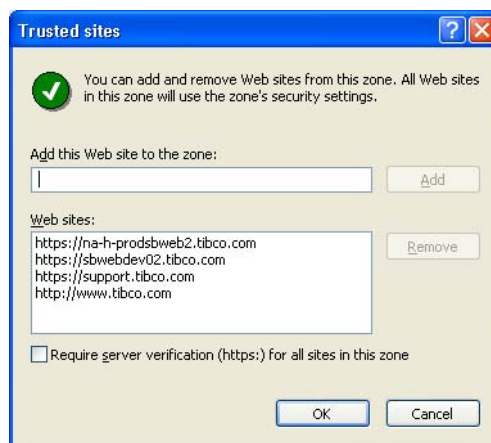
Figure 5: Remote Provisioning—After Browser Configuration

To modify browser settings on a client machine:

1. Exit TIBCO General Interface Builder.
2. In Internet Explorer, select **Tools > Internet Options** from the browser menu. The Internet Options dialog displays.
3. Click the **Security** tab, and then click the **Trusted Sites** zone.



4. Click the **Custom Level** button.
5. Enable the **Access data sources across domains** setting, then click **OK**.
6. Click the **Sites** button.



7. Type the name of the subdomain where the application is deployed in the **Add this Web site to the zone** field, and then click **OK**.