Business User Interface API Requirements

T

|  |  |
| --- | --- |
| **Product Suite** | **TIBCO BusinessEvents** |
| **Product** | **TIBCO BusinessEvents WebStudio** |
| **Product Version** | **5.2** |
| **Initial Author** | **Gabe Grigorescu** |

Copyright Notice

COPYRIGHT© 2014 TIBCO Software Inc. This document is unpublished and the foregoing notice is affixed to protect TIBCO Software Inc. in the event of inadvertent publication. All rights reserved. No part of this document may be reproduced in any form, including photocopying or transmission electronically to any computer, without prior written consent of TIBCO Software Inc. The information contained in this document is confidential and proprietary to TIBCO Software Inc. and may not be used or disclosed except as expressly authorized in writing by TIBCO Software Inc. Copyright protection includes material generated from our software programs displayed on the screen, such as icons, screen displays, and the like.

Trademarks

Technologies described herein are either covered by existing patents or patent applications are in progress. All brand and product names are trademarks or registered trademarks of their respective holders and are hereby acknowledged.

Confidentiality

The information in this document is subject to change without notice. This document contains information that is confidential and proprietary to TIBCO Software Inc. and may not be copied, published, or disclosed to others, or used for any purposes other than review, without written authorization of an officer of TIBCO Software Inc. Submission of this document does not represent a commitment to implement any portion of this specification in the products of the submitters.

Content Warranty

The information in this document is subject to change without notice. THIS DOCUMENT IS PROVIDED "AS IS" AND TIBCO MAKES NO WARRANTY, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TIBCO Software Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

For more information, please contact:

TIBCO Software Inc.  
3303 Hillview Avenue  
Palo Alto, CA 94304  
USA

Table of Contents

Change Log 4

1.1 Related Documents 4

2 Introduction 5

2.1 Purpose 5

2.2 Scope 5

2.3 Audience 5

2.4 Acronyms and Vocabulary 5

3 Requirements 6

3.1 Platform 6

4 Architecture 6

4.1 High Level Architecture 6

4.2 Components 6

5 Design 7

6 Project Milestone and Deliverables 7

# Change Log

.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Name | Description |
| 3/17/2014 | 0.1 | Gabe Grigorescu | Initial Document Creation |
| 3/20/2014 | 0.2 | Vikram Patil | Added some use cases and examples. |
| 5/20/2014 | 0.3 | Ryan Hollom | Add info on CORS |
|  |  |  |  |

## Related Documents

|  |  |  |
| --- | --- | --- |
| Document Title | Revision Number | Author |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

## Purpose

This document is intended to start capturing questions, decision points, high level components, and initial requirements and design around APIs we’ll expose for our Business User Interface, which are part of TIBCO BusinessEvents WebStudio.

## Scope

* Rationale and motivation for such APIs
* Functional requirements of technology
* Architectural pieces of the technology
* Design and implementation constraints

## Audience

* Developers
* Architects
* Quality Assurance
* Product Marketing
* Product Management and Support.

## Acronyms and Vocabulary

The table below describes some of the Acronyms used in this documentation

* BE – TIBCO BusinessEvents.
* BUI – Business User Interface
* CORS – Cross-Origin Resource Sharing

# Requirements

Need ability to enable our field and customers to create their own UIs, embed those UIs into their own systems…

Currently we offer WebStudio as an OOB Business user interface (BUI) to manage the lifecycle of Business artifacts (i.e. DT, RTI & eventually Process). Lifecycle management is at the heart of this product, i.e. everything goes through various states of the cycle.

Requirement as i understand,

Currently we provide everything OOB. There might be users, who would prefer to create their own UI, but still want our lifecycle management support. Or just want API’s to fetch/create business artifacts.

To facilitate that, we need to expose our complete artifact lifecycle management functionality via API's. The end user can then leverage these API's to create various UI components w/o lifecycle support.

**Use case** to cater to above requirement,

A user has an existing application and needs to show a specific DT in one of the sections of the application.

**Solution** - He can build a custom UI with any HTML5 toolkits available and use the exposed API's to show the DT. Example flow of this small module will be something like (with lifecycle support),

1. API call to login/authenticate
2. API call to Checkout a project
3. API call to Fetch DT
4. Render the data onto the custom UI.

**Note** - If you observe, i have maintained the lifecycle of artifact, i.e. fetch cannot happen without checkout and checkout cannot happen without authentication. Alternatively without any need for lifecycle support, it could be,

1. API call to login/authenticate
2. API call to Fetch DT
3. Render the data onto the custom UI.

In this flow the end user will be outside of the lifecyle support and will not be able to leverage those feature.

Any other use cases?

## Platform

Html5..

# Architecture

## High Level Architecture

Back end, RMS for now…

Yes, having our existing RMS system has lot of benefits, since we get a whole lot of things out of box, which otherwise will be quite tricky/time consuming to manage, distributed caching, backing, store, etc. to name a few.

Additionally, we already have all API’s as part of our RMS server, which currently our GWT based WebStudio application consumes. Need to check if we can expose the existing API’s with some amount of changes.

## Cross Origin Resource Sharing (CORS)

CORS (<http://www.w3.org/TR/cors/>) is a W3C spec that allows cross-domain communication from the browser to the server. As the WebStudio resources are contained within RMS itself (i.e. the embedded Tomcat instance), the client and server are within the same domain, so CORS support is not needed.

However, if the RMS APIs are exposed to external UIs, CORS support must be added on the server, and the client must properly format the request to use CORS.

### Creating CORS Request

Chrome, Firefox, Opera, and Safari all use the XMLHttpRequest2 object. Internet Explorer uses the XDomainRequest object, which is functionally similar to the XMLHttpRequest2 object. The SDK can provide APIs to handle the differences in these objects, in order to provide a general communication layer with error handling, etc.

For instance, a JavaScript API can be created as follows to setup a CORS request:

**function** createCORSRequest(method, url, addUserName){

**var** xhr = **new** XMLHttpRequest();

xhr.withCredentials = **true**;

**if** ("withCredentials" **in** xhr){

xhr.open(method, url, **true**);

} **else** **if** (**typeof** XDomainRequest != "undefined"){

xhr = **new** XDomainRequest();

xhr.open(method, url);

} **else** {

console.log("Null Request");

xhr = **null**;

}

**if** (addUserName) {

**var** uname = sessionStorage["username"];

xhr.setRequestHeader("username", uname);

}

**return** xhr;

}

### withCredentials

By default, CORS requests do not send or set any cookies. WebStudio requires certain cookies to be passed, for instance the WS\_SESSION\_ID, in order to pass the proper authentication context to the server. In order to include cookies as part of the request, the XMLHttpRequest’s .**withCredentials** property must be set to true (as seen in the example API above).

Further, the server must also enable credentials by setting the **Access-Control-Allow-Credentials** response header to “true”.

### Preflight request

The request defined above results in some extra communication between the client and server, called a preflight request. The preflight request essentially asks for permission to make the actual request, and is made as an HTTP OPTIONS request. Our RMS server must be able to respond to this request. Currently there are some areas that check only for GET or POST requests. The server side might need to be altered to support this preflight request.

### Preflight response

The preflight response then contains information regarding what is valid in the actual request.

These headers include (but are not limited to):

* **Access-Control-Allow-Origin**: Every CORS request contains an Origin header, which specified the origin of the request. The **Access-Control-Allow-Origin** specifies which domains are valid for this request, and can either specify an individual value (i.e. <http://test.tibco.com>), or can use the wildcard ‘\*’. Note, however, using the wildcard or simply echoing the request Origin value can lead to security issues. The best practice is to check the Origin value in the preflight request against a known list of valid values, and then respond appropriately. It remains to be seen how this can be done for a custom client UI.
* **Access-Control-Allow-Credentials**: Must be set to true in order for cookies to be sent in the CORS request.
* **Access-Control-Allow-Headers**: This is a comma separated list specifying the valid custom headers to be sent in the request. In our case, this would include **username, password, content-type,** and perhaps others.

Once the preflight request gives permissions, the browser makes the actual request. In other words, a CORS request will actually make two requests from client to server. The actual request is handled in the traditional way, by looking at the responseXML, responseText, etc.

NOTE: For completeness, the HttpServletResponse from RMS would add the above headers similar to the following:

httpServletResponse.addHeader("Access-Control-Allow-Origin", "http://192.168.0.4:8080");

httpServletResponse.addHeader("Access-Control-Allow-Credentials", "true");

httpServletResponse.addHeader("Access-Control-Allow-Headers", "username, password, content-type");

## Components

S

# Design

# Project Milestone and Deliverables

This section describes the Project tasks and items to be delivered. Maybe this is too much for early stages.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | | | Status | Comments |
|  | | |  |  |
|  |  | |  |  |
|  |  |  |  |  |