TIBCO ActiveMatrix BusinessWorks™ Plug-in for Mobile Integration Developer's Guide

Software Release 6.2.0
December 2016

Two-Second Advantage®

TIBCO®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO Hawk, TIBCO Rendezvous, TIBCO Runtime Agent, TIBCO ActiveMatrix BusinessWorks, TIBCO Administrator, TIBCO Designer, TIBCO ActiveMatrix Service Gateway, TIBCO BusinessEvents, TIBCO BusinessConnect, and TIBCO BusinessConnect Trading Community Management are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and 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.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2014-2016 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

TIBCO Documentation and Support Services 4
About BWMI Client Application Development 5
Communicating with MI Service 6
Developing a Mobile Integration Process 10

TIBCO Documentation and Support Services

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, visit:

https://docs.tibco.com

Product-Specific Documentation

The following documents for this product can be found in the TIBCO Documentation Library:

- TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration Installation
- TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration User's Guide
- TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration Developer's Guide
- TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration Expresso Server Guide
- TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration Release Notes

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to the following web address:

https://www.tibcommunity.com

About BWMI Client Application Development

Using the TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration, the developers can build mobile client applications (iOS and/or Android) that can receive notifications sent by MI Service through APNS and/or GCM server. Based on the push notifications received by the mobile client applications, it needs to communicate back to the MI Service. To accomplish this, appropriate MI REST/JSON services are called to ensure reliable notifications delivery.

This section focuses on these communication interfaces (REST/JSON services) between the mobile application and MI Service. It also identifies the interaction points in the mobile client application and when to invoke them.

Interaction Stages in the Mobile Application with the MI Service

At various stages in the mobile application, the mobile application developer has to interact with the MI services. These stages are:

- Call the Registration Service, when the mobile client application is installed.
- Call the Registration Service if the ID has changed, when the mobile client application is launched.
- When the mobile client application is uninstalled.
- When the user turns off the notifications from iOS Notification Store.
- Send an Acknowledgement, when a user clicks on the notification.

You will get familiar with the interaction points such as, the REST/JSON services to be called, and the data to be exchanged. It is important that the mobile application developer implements each of these interaction points to achieve the required reliable notification behavior.

Client Application Development for iOS

As part of the mobile application development of iOS, the applicatiob developer must follow the required process. For details, refer to https://developer.apple.com/programs/ios/.

The application developer must also create a certificate for the mobile application, which is required to be configured with the MI Service for communicating with the APNS.

Client Application Development for Android

The application developer must follow the required process as part of the mobile application development of Android. For details, refer to http://developer.android.com/google/gcm/gcm.html.

Communicating with MI Service

The MI Service publishes certain service end points to the mobile client applications (iOS and Android) for consumption during the communication process.

The following end points are RESTful web services:

- **Device Registration Service Interface**: to register the device for receiving the notifications.
- **Device Deregistration Service Interface**: to deregister the device to stop receiving notifications from the Mobile Integration service.
- **Acknowledgment Service Interface**: to accept acknowledgments sent by the device after receiving the notifications.
- **Pending Notifications Service Interface**: to serve the pending notifications to the device.

Payload Keys

The following table lists the payload keys and what they denote.

Payload Keys	Definition
`nid`	Notification ID
`bid`	Signifies the Business ID specified for correlating the business process and notifications.
`faf`	FireAndForget flag indicates whether to manage or not to manage the notifications.
`badge`	Signifies the number of waiting notifications on the MI service. It is dynamically stamped in case of managed notifications, and as per the user specification for the notifications that are not managed.
`ns`	Signifies the notifications containing an array of the list of specified properties.

Device Registration Service Interface

Here you get familiar with Device Registrations Service Interface for the iOS and Android applications.

For iOS Mobile Application

Whenever the mobile application is installed or launched, it communicates with APNS to obtain its device token. The mobile application calls this BWMI service to communicate back the device token (deviceId) it has received from APNS:

- For a new device token
- When the device token has changed
- When the user turns on the notification through the IOS notification store configuration

Refer to the following REST/JSON data to communicate back to the MI service:

```
URI : http://<serverHost>:<serverPort>/devices
Method: PUT
Request Headers: Content-Type: application/json; charset=UTF-8
Request Body:
{
"deviceId":"d7a9876f73f84725914a11a25b89c3769f213d453b62ec7d38d241
b2fc10a8445",
```

```
       "type":"IOS",
       "user": {
               "phone": "+19883999901",
               "email": "abc@yahoo.com"
       }
}
Response Code:201 Created
Response body: None
```

For Android Mobile Application

Whenever the mobile application is installed or launched, it communicates with GCM server to obtain its Registration ID (deviceId). The mobile application then calls this MI service to communicate back the Registration ID it has received from the GCM Server in the following cases:

- For a new registration id
- When the registration id has changed
- When the user turns on the notification through the Android notification store configuration

Refer to the following REST/JSON data to communicate back to the BWMI service:

```
URI : http://<serverHost>:<serverPort>/devices
Method: PUT
Request Headers: Content-Type: application/json; charset=UTF-8
Request Body:
{

"deviceId":"GPA91bEthw8OzOYyDykm1Lvvcd3gP7oe52C9fiYobOVPems10fGmgQ
pkTpJIht9UplhsygL2zUwa2B5YjpsP3MexPb6hmv-2E9AkJKQyeBCrLvSvJL_FBser
PXskrx4LhVQeHZu3_maSIW6GkFVPtwNIzfxY-GfsIA",
         "type":"ANDROID",
         "user": {
                 "phone": "+19883999901",
                 "email": "abc@yahoo.com"
           }
}
Response Code:201 Created
Response body: None
```

Device Deregistration Service Interface

The mobile application calls this BWMI Deregistration service to communicate the inactive or invalid status of the mobile application whenever:

- The user turns off the notification through the IOS notification store configuration.
- The user turns off the device or uninstalls the application.

As a result of this deregistration service, the MI service ensures that no further notification is sent to this mobile application unless it receives a registration.

The following REST/JSON data is required to communicate back to the MI service.

For Mobile Client Application

```
Un-register Device
URI : http://<serverHost>:<serverPort>/devices
Method: DELETE
Request Headers: Content-Type: application/json; charset=UTF-8
Request Body:
{

          "deviceId":"d7a9876f73f84725914a11a25b89c3769f213d453b62ec
7d38d241b2fc10a8445",
          "type":"IOS|ANDROID",
          "user": {
                  "phone": "+19883999901",
                  "email": "abc@yahoo.com"
         }
}
```

```
Response Code: 200 OK
Response body: None
```

Acknowledgement Service Interface

The mobile application calls the Acknowledgement Service, when the user views the managed notification in the mobile application. Use the JSON data from the managed notification to construct the acknowledgement request.

The following REST/JSON data is required to communicate back to the MI service.

```
URI : http://<serverHost>:<serverPort>/acks
Method: PUT
Request Headers: Content-Type: application/json; charset=UTF-8
Request Body:
{"nid":"<notificationID>","did":"<deviceToken>","bid":"<businessID>"}
Response Code: 201 Created
Response body: None
```

Pending Notifications Service Interface

Whenever the mobile application is installed or launched, it communicates with APNS or GCM server to obtain its device token. The mobile application calls this MI service to communicate back the device token (deviceId) it has received from APNS or GCM server. This happens when the user:

- Clicks on the notification in the notification store and the mobile application is launched.
- Explicitly launches the mobile application.

The following REST/JSON data is required to communicate back to the MI service.

```
URI :
http://<serverHost>:<serverPort>/devices/{deviceToken}/notifications
Method: GET
Request Headers: None
Request Body: None
Response Code: 200 OK
Response Content-type: application/json;
Response body:
{
 "ns":[
[<notification id string>,<UTC date string>,<notification alert
text>,<optional businessId string>],
[<notification id string>,<UTC date string>,<notification alert
text>,<optional businessId string>],
[<notification id string>,<UTC date string>,<notification alert
text>,<optional businessId string>]
]
}
For example,
{
 "ns":[
["b8ba7a54-92dc-4978-b125-53bfe57daa6c","1365753980483","Boarding
announced at Terminal 1, Gate B67","GATE_CHANGE"],
["fd97b8b3-4aa3-47f3-8023-606e3c548d4d","1365753981820","Flight
canceled, "FLIGHT_CANCELLATION"],
["9d5e1138-3b2e-4f88-8185-46d6e38fa21b","1365753983879","20% discount
on all Round Trip travel bookings","OFFERS"]
  ]
```

Push Notification Payloads

The following are the iOS and GCM payloads notifications received by the mobile applications:

- **Payloads Delivered to the iOS Application Through APNS**: For iOS payloads, refer to Apple documentation for Apple Push Service.
- **Payloads Delivered to the Android Application Through GCM Server**: Android application receives the following payloads from the MI Service through GCM Server.

The application developers must handle the following payloads in their application.

- **For Managed Notification**: These notifications are managed by the MI Service.

```
{
"text":"Cash withdrawal",
"timestamp":"1374224245011",
"nid":"0a1868e6-04c9-4d3b-8125-2f9597f68e78",
"bid":"USER_TRANSACTION",
"faf":"false",
"badge":"1"
}
```

- **For UnManaged Notification**: These notifications are not managed by the BWMI service.

```
{
"text":"Salary deposited",
"faf":"true",
"sound":"default",
"badge":"1"
}
```

- **For Badge Update**: If there is no live notification on the server, then it sends this badge value as 0.

```
{
"badge":"1"
}
```

- For Pulled String of Notifications

```
{
"ns":[
["642d1380-6b98-4108-ba97-98e54483d8b9","1375952245016","Alert
2:27:25 PM"],
["78241424-0736-4301-b45f-920bd8ad84ed","1375952255016","Alert
2:27:35 PM"],
["0187a77f-b757-4b84-8768-c1ac2ddad133","1375952265015","Alert
2:27:45 PM"],
["7f015ae7-423b-4dd6-9bd2-3198d7fe534f","1375952275008","Alert
2:27:55 PM"]
]
}
```

Developing a Mobile Integration Process

The project in this tutorial focuses on how to use TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration with TIBCO ActiveMatrix BusinessWorks Enterprise. After completing the tutorial, you will be able to apply this methodology to your own projects.

Creating a BWMI Notification Process

This section guides walks you through creating and testing a process in TIBCO Business Studio. The process is named as `bwmiNotification`.

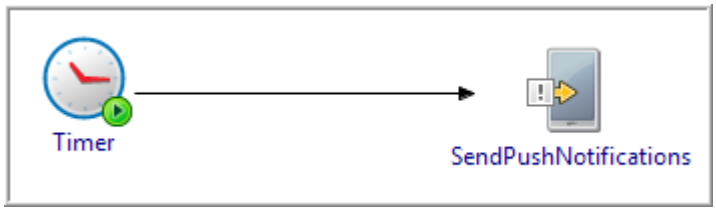
Prerequisites

Install:

- TIBCO ActiveMatrix BusinessWorks Enterprise
- TIBCO ActiveMatrix BusinessWorks Plug-in for Mobile Integration

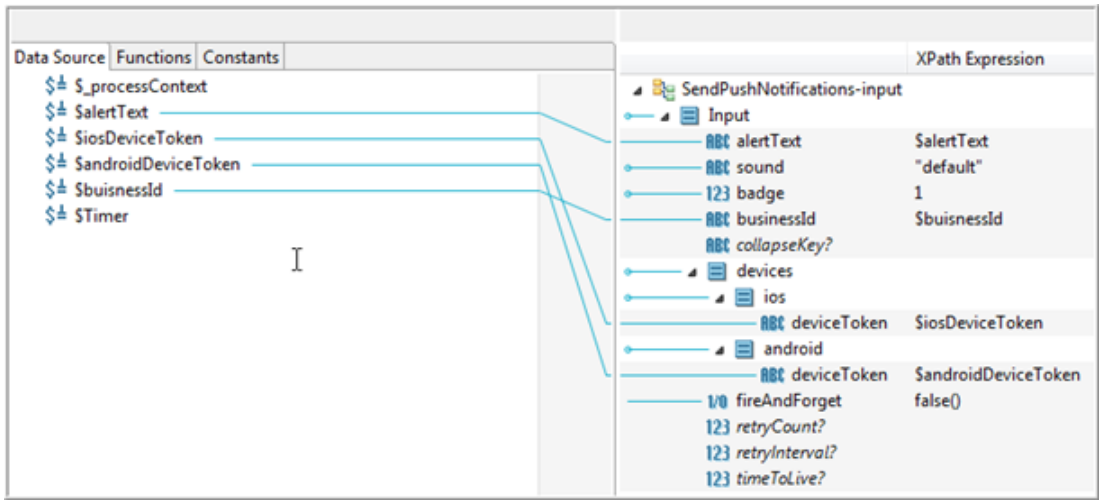
Procedure

1. Start `$TIBCO_HOME\studio\3.6\eclipse\TIBCOBusinessStudio.exe`.
2. To create a new project, select **File** > **New** > **Project** > **BusinessWorks** > **BusinessWorks Application Module**.
3. Specify `bwmiNotification` for the project name and click **Finish**.
4. In the **Process Editor** window, select and drop a **Timer** activity from the **General Activities** palette.
   To add an activity on to the Process Editor, click on the activity and drop it on the Process Editor. Do not drag-and-drop the activity.
5. Select the **SendPushNotifications** activity from the **MI** palette and drop next to the **Timer** activity.
6. Create a link from the **Timer** activity to the **SendPushNotifications** activity using the icon.

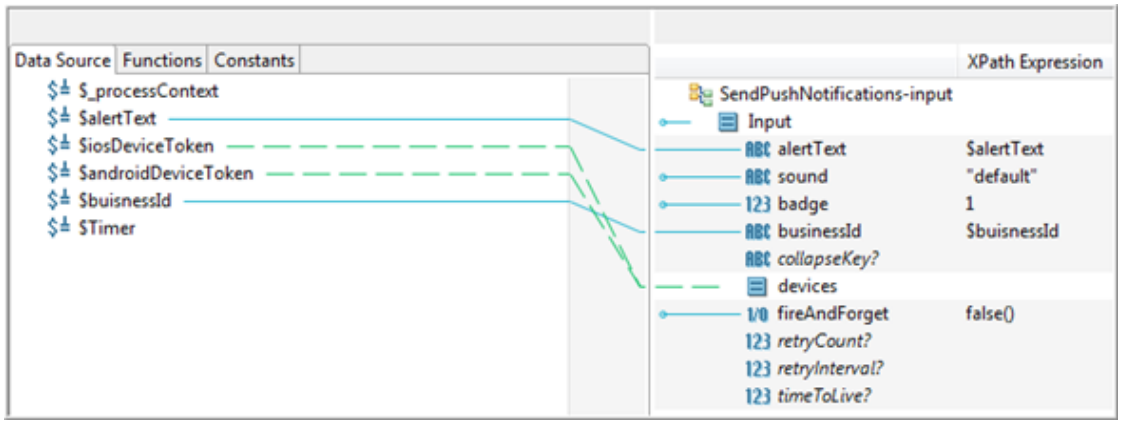


   You will find this image icon on the right of the Process Editor under the Palettes.
7. Select the **Timer** activity and in the **Properties** tab, select the **General** tab.
8. Select the **Run Once** check box, if you want the notification to be sent once.
   The **Run Once** check box when cleared, shows the **Time Interval** and **Time Unit** fields.
9. In the **Time Interval** field, specify the time (30 seconds).
   Default is 1. The recommended time interval is 30.
10. Select the **Interval Unit** as **Second** from the drop-down list.
    You can select the interval unit as per your requirement. By default it is seconds.

    Before creating this process, ensure that your device is registered and connected to the internet.
11. Select **SendPushNotifications** in the **Process Editor** and in the **Input** tab, add the device token and map it to the iOS or Android.



12. Select **alertText** in the **Data Source** and map it to the **alertText** in the XPath Expression editor.



13. In the **sound** field, add `"default"` to enable the notification alert sound.
14. Click the Debug icon to test the project.

Result

Your mobile application receives the notification and an acknowledgement is delivered to the TIBCO ActiveMatrix BusinessWorks Mobile Integration service.