TIBCO to Ballerina migration tool

# What is the problem we are trying to solve?

* Given a TIBCO Business Work (BW) project automatically generates a ballerina project that can then be used as a starting point in a migration.
  + In the best case scenario we should be able to automatically convert the whole project. Where this is not possible, the tool should gracefully generate placeholders just for specific parts it failed to convert which user can manually fill in.

# Who are we solving the problem for?

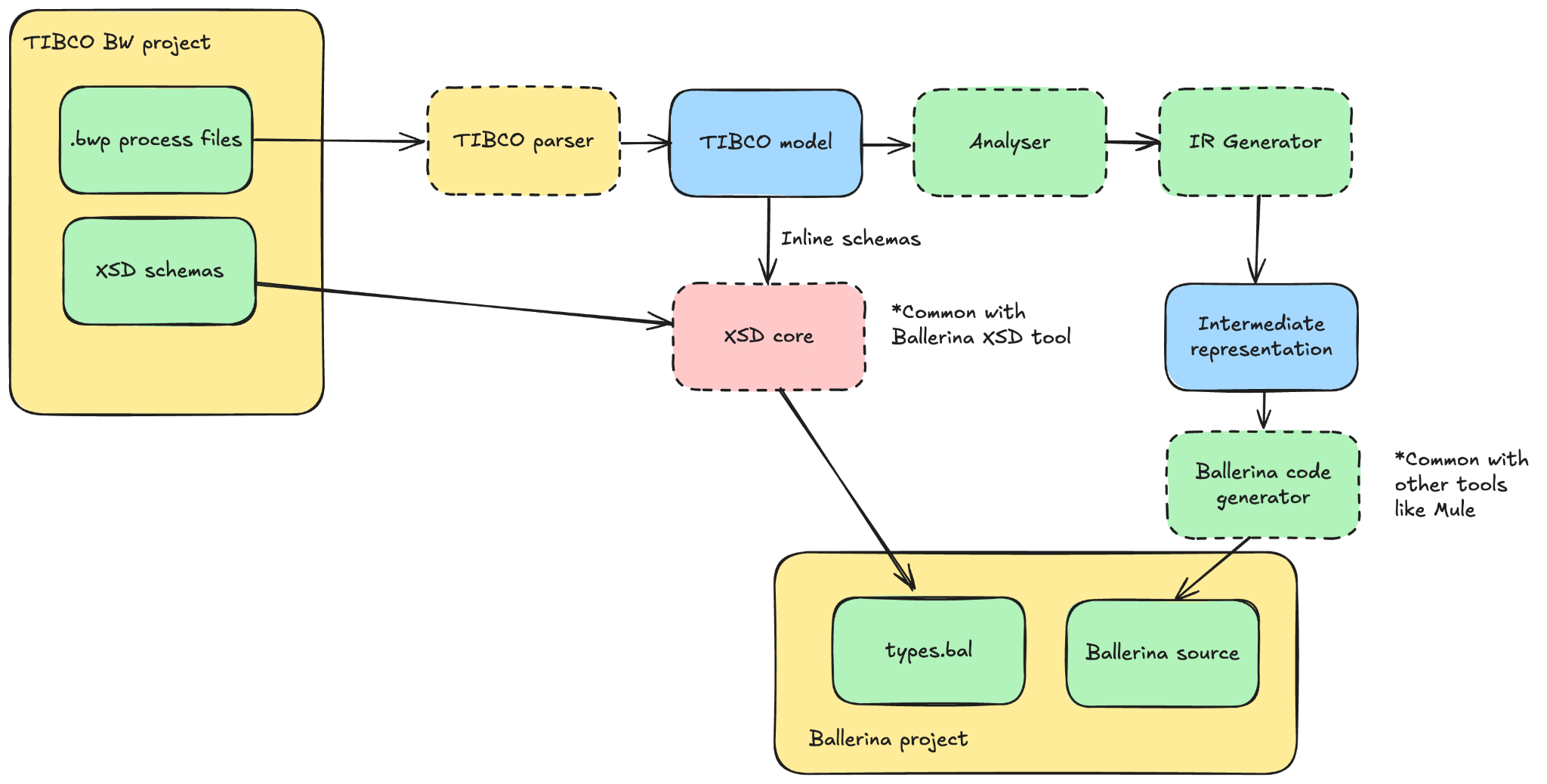
* People who are considering migrating from TIBCO BW to Ballerina

# Why should it be solved?

* Reduce friction for current users of TIBCO BW to migrate to Ballerina

# Proposed solution

## Architecture



* XSD Core converts a given XSD schema (given as string) to a Ballerina AST. This is shared between the ballerina XSD tool [1] and TIBCO conversion tool.
* Both intermediate representation and logic for generating Ballerina source using it is shared between TIBCO conversion tool and Mule conversion tool [2].
* At the moment we are targeting only Business work version 5.x

## Mapping between Tibco and Ballerina

*All examples here represents the CreditAppServiceExample[3]*

### Process starters / event handlers

* These would be represented as a service that directly call the “process start function”

service / on creditapp\_module\_ExperianScore\_listener {

resource function post creditscore(GiveNewSchemaNameHere input) returns ExperianResponseSchemaElement|http:NotFound|http:InternalServerError {

return creditapp\_module\_ExperianScore\_start(input);

}

}

* Process start function takes care of converting input data to the process into XML and converting the resulting XML to expected type

function creditapp\_module\_ExperianScore\_start(GiveNewSchemaNameHere input) returns ExperianResponseSchemaElement {

xml inputXML = checkpanic toXML(input);

xml xmlResult = process\_creditapp\_module\_ExperianScore(inputXML);

ExperianResponseSchemaElement result = convertToExperianResponseSchemaElement(xmlResult);

return result;

}

* + We are converting everything to XML because most of the business logic in TIBCO processes is written using XSLT and XPath. We generate on demand logic to convert XML to Ballerina records where needed.
* Processes that call this process (using invoke activity) will do so by directly calling this start function instead of invoking the service.
* Start function will invoke the “process function” (process\_creditapp\_module\_ExperianScore in the example)

### Process function

* ~~Process function will represent the activity graph in Tibco process using workers~~

~~function process\_creditapp\_module\_ExperianScore(xml input) returns xml {~~

~~map<xml> context = {};~~

~~worker startWorker {~~

~~xml|error result0 = receiveEvent(input, context);~~

~~if result0 is error {~~

~~result0 -> errorHandler;~~

~~return;~~

~~}~~

~~result0 -> StartToSendHTTPRequest;~~

~~}~~

~~...~~

~~worker activityExtension\_3\_worker {~~

~~error:NoMessage|xml inputVal = <- StartToSendHTTPRequest;~~

~~if inputVal is error:NoMessage {~~

~~return;~~

~~}~~

~~xml|error output = activityExtension\_3(inputVal, context);~~

~~if output is error {~~

~~output -> errorHandler;~~

~~return;~~

~~}~~

~~output -> RenderJSONToSendHTTPRequest;~~

~~}~~

~~...~~

~~worker errorHandler {~~

~~error result = <- startWorker | activityExtension\_3\_worker | activityExtension\_2\_worker | activityExtension\_4\_worker | activityExtension\_worker;~~

~~...~~

~~}~~

~~error:NoMessage|xml result = <- activityExtension\_worker;~~

~~return result;~~

~~}~~

~~~~

* ~~We will have separate workers for links and activities~~
  + *~~In theory it should be sufficient to have workers just for links but IMHO it makes code generator more complicated.~~*
* ~~Each “activity worker” (activityExtension\_3\_worker in above example) will call the activity function (activityExtension\_3). In case of error it will invoke errorHandler (representing error transitions in TIBCO). Otherwise it will pass the result of the activity to “link workers” corresponding to its destination links~~
* ~~Link workers pass the value it gets to any activity worker whose activity treats that link as a source.~~
* Process functions represent the control flow between activities. In order to generate this we will perform a depth first topological sort and call the corresponding activity functions.
* There is an “inner” process function that represents the “normal” flow and in case of an error it will immediately return the error

function activityRunner\_creditapp\_module\_MainProcess(xml input, map<xml> cx) returns xml|error {

xml result0 = check extActivity\_11(input, cx);

xml result1 = check extActivity(result0, cx);

xml result2 = check reply(result1, cx);

return result2;

}

* This would be called by the “outer” process function which in case of error passes the error directly to the error handler. This outer function will be called by the process start function.

function process\_creditapp\_module\_MainProcess(xml input) returns xml {

map<xml> context = {};

...

xml|error result = activityRunner\_creditapp\_module\_MainProcess(input, context);

if (result is error) {

return errorHandler\_creditapp\_module\_MainProcess(result, context);

}

return result;

}



### Activity function

* Actual logic of each TIBCO activity is represented as function like

function activityExtension\_3(xml input, map<xml> context) returns xml|error {

xml var0 = check xslt:transform(input, transformXSLT(xml `<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:tns6="/y54cuadtcxtfstqs3rux2gfdaxppoqgc/T1535409245354Converted/JsonSchema" xmlns:tns="http://tns.tibco.com/bw/activity/jsonRender/xsd/input/55832ae5-2a37-4b37-8392-a64537f49367" version="2.0">

<xsl:param name="Start"/>

<xsl:template name="RenderJSON-input" match="/">

<tns:InputElement>

<tns:dob>

<xsl:value-of select="$Start/tns6:DOB"/>

</tns:dob>

<tns:firstName>

<xsl:value-of select="$Start/tns6:FirstName"/>

</tns:firstName>

<tns:lastName>

<xsl:value-of select="$Start/tns6:LastName"/>

</tns:lastName>

<tns:ssn>

<xsl:value-of select="$Start/tns6:SSN"/>

</tns:ssn>

</tns:InputElement>

</xsl:template>

</xsl:stylesheet>`), context);

addToContext(context, "RenderJSON", var0);

return var0;

}

* We are using the context map to store variables and input will be the output of the last activity
  + *We can get rid of input parameter and instead make it part of the context as well*
* If we can’t convert the activity this will be a placeholder function with a comment containing the part of the .bwp file with the activity.

## To be finalized

### XPath support

* Current plan is to create a ballerina library wrapping an existing java library for this.
* We need minimum support for version 2.

### Configuration

* Currently we generate configurable variables for any configuration values needed, but need to explore how to use configuration already in the TIBCO project to fill in Config.toml file

### Activity conversion

* Currently we explicitly parse each activity and code gen activity function for that activity.
* This means in order to support any new activity we need to modify the tool’s source and rebuild it. Also this means we have to explicitly create mapping per each activity.
* Instead it is more desirable to have some sort of configuration file that does the mapping and tool implement a generic parsing and codegen logic on this configuration. It is also potentially easier to use AI tools to do the configuration generation if we have some sort of structured configuration file, rather than have it modify the tools code.
  + Ideally most of this should be shared between TIBCO and Mule migration tools.

# Tibco activity to ballerina mapping

* Activities marked as case by case means there is no equivalent ballerina library and most likely converter will have to generate code specific for each activity's configurations
* TBD activities should be possible at first glance but specific implementation is yet to be determined.

## Process starters

| **Palette** | **Process Starter** | **Ballerina equivalent** |
| --- | --- | --- |
| ActiveEnterprise Adapter | Adapter Subscriber | \* Will need to support case by case |
| Adapter Request-Response Server | A message or a request from an adapter is received. | \* Will need to support case by case |
| File | File Poller | file:Listner |
| General Activities | Timer | ballerina/task |
|  | Receive Notification | \* Will need to support case by case |
| HTTP | HTTP Receiver | http:Listner |
| JMS | JMS Queue Receiver |  |
| JMS Topic Subscriber | Either a JMS queue or topic message is received. | ballerinax/java.jms |
| Mail | Receive Mail | \* Will need to support case by case |
| Rendezvous | Rendezvous Subscriber | - Tibco proprietary messaging format |
| SOAP | SOAP Event Source | http:Listener |

## Other activities

| **Palette** | Comment | Ballerina equivalent |
| --- | --- | --- |
| Manual Work | The activities in the Manual Work palette are useful for automated business processes that have a few steps that require user interaction | None |
| FTP |  | ftp:Client |
| Checkpoint | The **Checkpoint** activity performs a checkpoint in a running process instance. A checkpoint saves the current process data and state so that it can be recovered at a later time in the event of a failure. |  |
| Custom Activity |  | Case by case |
| Engine Command | The **Engine Command** activity allows you to retrieve statistics and information about process definitions, process instances, and activities in the currently running process engine. | None |
| External Command |  | ballerina/os |
| Generate Error | The **Generate Error** activity generates an error and causes an immediate transition to any error transitions. | check an error value and jump to error handler |
| Get Shared Variable | The **Get Shared Variable** activity retrieves the current value of a [Shared Variable](https://docs.tibco.com/pub/activematrix_businessworks/5.16.1/doc/html/palette-reference/Shared_Variable.htm#shared_variable) or [Job Shared Variable](https://docs.tibco.com/pub/activematrix_businessworks/5.16.1/doc/html/palette-reference/Job_Shared_Variable.htm#job_shared_variable) resource. | Should be possible via context |
| Inspector | The **Inspector** activity is used to write the output of any or all activities and process variables to a file and/or stdout. | ballerina/io |
| JNDI Configuration | The JNDI Configuration shared configuration resource provides a way to specify JNDI connection information that can be shared by other resources. | Currently none but give this is java if needed we should be able to expose as a library |
| Job Shared Variable | A Job Shared Variable shared configuration resource allows you to store data for use by each process instance. | Via context |
| Lock Object | Lock Object shared configuration resources are used by Critical Section groups to ensure that only one process instance executes the activities in a critical section group at a time. | lock |
| Mapper | The **Mapper** activity adds a new process variable to the process definition. | Via context |
| Notify | The Notify activity allows a process instance to send data to a corresponding process instance containing a Wait activity or Receive Notification process starter. | TBD |
| On Event Timeout | The On Event Timeout process starter specifies a process to execute when a Wait For activity discards an incoming event due to a timeout. | TBD |
| On Notification Timeout | The **On Notification Timeout** process starter specifies a process to execute when a timeout is reached for storing notification data for a [Notify](https://docs.tibco.com/pub/activematrix_businessworks/5.16.1/doc/html/palette-reference/Notify.htm#notify) activity. | TBD |
| Receive Notification | The **Receive Notification** process starter starts a process when another process executes a [Notify](https://docs.tibco.com/pub/activematrix_businessworks/5.16.1/doc/html/palette-reference/Notify.htm#notify) activity with a matching key and Notify Configuration resource. | TBD |
| On shutdown | The On Shutdown process starter specifies a process to execute when the process engine shuts down, after all process instances are executed. | runtime:onGracefulStop |
| On Startup | The On Startup process starter specifies a process to execute when the process engine starts, before any incoming events are processed. | init |
| Rethrow | The **Rethrow** activity throws the exception caught by the [Catch](https://docs.tibco.com/pub/activematrix_businessworks/5.16.1/doc/html/palette-reference/Catch____.htm#catch) activity again. Use this activity when you want to propagate the exception to the next level. | check an error |
| Sleep | The **Sleep** activity suspends the process on the current transition for the given amount of time. | Depending on how we model activity flow can be represented by runtime:sleep |
| Write To Log |  | log |
| OnError | This activity provides error handling mechanism for all the errors that happen outside the job boundaries of all processes associated with the service resource. | At the end of each “activity function” check if error and if so jump to error handler |
| Java | The **Java** palette has activities and shared configuration resources for executing Java code as well as converting between Java objects and XML documents. | FFI |
| JDBC Palette |  | ballerinax/java.jdbc |
| Parse palette | The **Parse** palette provides shared configuration resources and activities for parsing and rendering formatted text. | Case by case (some has libraries like csv) |
| Policy Palette | The **Policy** palette enables you to specify security policies for inbound and outbound SOAP messages | TBD |
| RMI Palette | Remote Method Invocation (RMI) is a protocol for calling operations on remote Java objects. ActiveMatrix BusinessWorks can act as server for incoming RMI calls, or ActiveMatrix BusinessWorks can use RMI to retrieve a remote object. | TBD |

[1] <https://ballerina.io/learn/xsd-tool/>

[2] See email: [Architecture] Mule to Ballerina Mapping Tool Design and Implementation in architecture group

[3] <https://github.com/TIBCOSoftware/bw-samples/tree/master/TN2018/Apps/CreditAppService>

[4] <https://github.com/TIBCOSoftware/bw-samples/tree/master>