TIBCO™
Education Programs

M-BW6I: TIBCO ActiveMatrix BusinessWorks™ 6.x Development

Student Workbook

January 18, 2017

Two-Second Advantage®

TIBCO™

Important Information

Copyright Notice

COPYRIGHT© 2017 TIBCO Software Inc. All rights reserved. No part of this document may be reproduced in any form, including video recording, photocopying, downloading, broadcasting or transmission electronically, without prior written consent of TIBCO Software Inc. Copyright protection includes content in the material generated from software programs displayed on the screen, such as icons, screen displays, and the like.

Trademarks and Patents

All brand and product names are trademarks or registered trademarks of their respective holders and are hereby acknowledged. Technologies described herein may be covered by existing U.S. patents or U.S. patent applications that are in progress. Please consult the software product user documentation for details regarding applicable patents.

Confidentiality

Information contained in this material is confidential and proprietary to TIBCO Software Inc. and its affiliates and may not be modified, copied, published, disclosed, distributed, displayed or exhibited, in either electronic or printed formats without written authorization of an officer of TIBCO Software Inc.

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 OR NON-INFRINGEMENT.** 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.

Export

This material and related technical data are subject to U.S. export control laws, including without limitation the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations of other countries. You agree not to export or re-export this material in any form in violation of the applicable export or import laws of any jurisdiction.

Preface xv

Conventions Used in this Book xvi

Course Overview 1

Important Information. 2
Objectives 3
Agenda. 4
TIBCO Software Featured 5
BusinessWorks 6 Application Lifecycle 6
Developing BusinessWorks Solutions 7
BusinessWorks Application Details 8
Design Time - Recap 9
Developing Services 10
Multi-Platform Support 11
Test Your Skills Lab 12

Lab A: Test Your Skills 15

Create Application Module using Wizard 16

Unit 1 BWSX Business Case 19

Objectives 20
Business Context for BW Super Express (BWSX) 21
Project Analysis Best Practices 22
BWSX Business Process. 23
Query Order Process and Data Model. 24
Place Order Process 25
Architecture Pattern 26
QueryOrder: Process Pattern Mapping 27
PlaceOrder: Process-Pattern Mapping 28
BWSX Business Requirements 29
BWSX Implementation Details. 30
Best Practices - Project Design 31
Best Practices - Process Design 32
Best Practices - Mapping Data 33

Unit 2 Building RESTful Services 35

Objectives 36

REST Fundamentals. 37
REST Support in BusinessWorks 38
REST Architectural Elements 39
REST Architectural Elements and Implementations 40
Defining a RESTful Service 41
BusinessWorks REST Wizard 42
Completing REST Services Definition 43
Filling in the Details. 44
REST Operation Details 45
Testing a RESTful Service 46
Business Case Implementation 47
Process-Pattern Mapping - Lab B. 48
Interface Specification - Lab B 49
Review 50
Lab Overview 51
Lab B: Creating a RESTful Service 53
Define RESTful Service 54
Using Swagger UI 61
Using Parameters (Optional) 65

Unit 3 Debugging and OSGi 73
Objectives 74
The BusinessWorks Debugger 75
Setting and Using Breakpoints 76
Viewing Job Data 77
BusinessWorks Debug Environment 78
Setting up Debugging Details 79
BusinessWorks and OSGi 80
BusinessWorks Application Startup 81
Accessing the OSGi Console 82
Key OSGi Commands 83
Viewing Bundle and Engine Details 84
Endpoint Details 85
Review 86
Lab Overview 87
Lab C: Debugging Applications 89

Debug a Solution 90
Using OSGi Console 96

Unit 4 Configuring JDBC Drivers 101

Objectives 102
Transitioning JDBC Activity Based Projects. 103
Packaging an Application. 104
Connecting to a Database using JDBC 105
JDBC Driver: Setting up Design Time 106
Design Time JDBC Configuration 107
JDBC Driver: Run Time Changes 108
Working with a Database 109
Adding a JDBC Query 110
JDBC Connections in a Process 111
Database Schema - Lab D. 112
Interface Specification for Lab D 113
Review 114
Lab Overview 115

Lab D: Database Connectivity 117

Design Time Connectivity 118
Running JDBC Based Applications 127
Using Stored Procedures 131

Unit 5 Working with Properties 137

Objectives 138
Variables and Properties 139
Understanding Scope of Properties 140
Accessing Process Properties 141
Defining Process Properties 142
Process Properties vs. Process Variables 143
Process Property Features 144
From Process Properties to Module Properties 145
Module Property Details. 146
Types of Module Properties 147
Application Properties 148
Property Layers and Scope 149

Application Configuration Profiles 150
Variables 151
Review 152
Lab Overview 153
Lab E: Properties and Variables 155
Working with Module Properties 156
Using Module Properties 160
Types of Properties 172

Unit 6 Using Java Message Service 177
Objectives 178
Fundamentals of Messaging 179
Messaging Features 180
Message Processing Choices 181
Java Message Service (JMS) 182
TIBCO Enterprise Message Service 183
Publish-Subscribe Messaging 184
Point-to-Point Messaging 185
JMS Message Format 186
Client-Service Delivery Modes 187
TIBCO EMS Access Control 188
Administering TIBCO EMS 189
JMS In BusinessWorks 190
Java Projects in Business Studio 191
Review 192
Lab Overview 193
Lab F: JMS Messaging 195
Use JMS Solution 196
Build JMS Client 199

Unit 7 Adding Java Projects 205
Objectives 206
Business Studio: Additional Features 207
Creating Other Projects in Business Studio 208
Creating a Java Project 209
Java Perspective 210

Adding External JAR Files 211
Running Java Projects 212
BusinessWorks Projects in Java Perspective 213
Review 214
Lab Overview 215
Lab G: Java Projects 217
Adding a Java Project 218

Unit 8 Developing REST Clients 227
Objectives 228
Representational State Transfer (REST) 229
SOAP vs REST 230
JavaScript Object Notation (JSON) 231
REST and JSON Palette Overview 232
Invoke REST API 233
Parse JSON and Render JSON. 234
Conversion between XML and JSON 235
Normal Conversion 236
Badgerfish Conversion. 237
Business Case Implementation 238
Review 239
Lab Overview 240
Lab H: REST Clients 243
Build RESTful Client 244
Use a RESTful Client 251

Unit 9 Using Groups and Loops 257
Objectives 258
Types of Groups. 259
Grouping For Repetitive Actions 260
Using Scope. 261
Error Handling Details 262
Implementing Transactions 263
Using Checkpoints with Transactions 264
Process Inventory Definition 265
Grouping Best Practices 266

Interface Specification 267
Process-Pattern Mapping 268
Review 269
Lab Overview 270
Lab I: Loops and Transactions 273
Adding Transactions 274
Testing Transactions 285

Unit 10 Embedding Java Code. 297
Objectives 298
Java Nature for Application Module 299
Confirming Java Nature 300
Adding External Java Code to Module 301
Using Product Bundles 302
Using the Mapper 303
Variables in Mapper 304
Inline Schema Editor. 305
Business Case Implementation 306
Java Implementation. 307
Review 308
Lab Overview 309
Lab J: Using Java Code 311
Importing Existing Code 312
Get OrderID and Price 316

Unit 11 Building Plug-in Projects 325
Objectives 326
Java Code: The OSGI Way 327
OSGI Activator Classes 328
Using a Plug-In Project 329
Referencing a Plug-in Project 330
Using Non-OSGI Java Code. 331
Using Non-OSGi Code 332
Business Case Implementation 333
Java Plugins 334
Review 335

Lab Overview 336
Lab K: Plug-in Projects 339
Use Plug-In Project 340
Call External Method 349
Packaging JARs 353

Unit 12 Invoking Services 363
Objectives 364
BusinessWorks 6: Process Types 365
Sub-Process Types 366
Invoking WSDL Based Sub-Process 367
Dynamic Process Invocation 368
Dynamic Process: Name Generation. 369
Service Invocation: Internal 370
Invoking an External Service 371
Reference Binding: Review 372
Business Case Implementation 373
Process-Pattern Mapping to Test Invocation Options 374
Review 375
Lab Overview 376
Lab L: Process Invocation Techniques 379
Sub Process Invocation 380
Dynamic Sub Process 384
Using Binding Reference 388
Calling External Service 393

Unit 13 Using Conversation 397
Objectives 398
Synchronous Communication 399
Asynchrounous Communication 400
Asynchrounous Communication in Messaging 401
Asynchronous Communication in Services 402
Setting Up a Conversation 403
Joining a Conversation 404
The Conversation Tab 405
Activities Supporting Conversations 406

Conversation Details 407
Interface Specification 408
Process-Pattern Mapping to Test Conversations 409
Review 410
Lab Overview 411
Lab M: Use Conversation 413
Define a Conversation 414
Test Conversation 420

Unit 14 Implementing Event Handlers 431
Objectives 432
Events 433
Event Handlers 434
Changing Event Handler Scope 435
Multiple Event Handlers 436
Defining an Event Handler 437
Adding Multiple Event Handlers 438
Event Handler Business Logic 439
Interface Specification - Lab N 440
Process-Pattern Mapping to Test Event Handler 441
Review 442
Lab Overview 443
Lab N: Define Event Handlers 445
Add Event Handler 446

Unit 15 Using Compensation 459
Objectives 460
Compensation Logic 461
Compensation Handler 462
Explicitly Invoking Compensation Handler 463
Compensation Handler: Implicit Invocation 464
Rules for Compensation 465
Compensation Logic Details 466
Interface Specification for Lab O 467
Implicit Compensation - Process-Pattern Mapping 468
Explicit Compensation - Process-Pattern Mapping 469

Review 470
Lab Overview 471
Lab O: Compensating Transactions 473
Implicit Compensations 474
Explicit Compensations 481

Unit 16 Enterprise Deployment 493
Objectives 494
Enterprise Deployment Concepts 495
Enterprise Mode: Communication and Data Storage 496
TIBCO Enterprise Administrator. 497
Comparing Administrator and TEA. 498
Domain Communication and Runtime Components 499
Setting up TEA for BusinessWorks 500
TIBCO Enterprise Administrator Features 501
BusinessWorks Administration Views 502
Deployment Options 503
Deployment Choice 1: bwadmin 504
Deployment Choice 2: TIBCO Business Studio. 505
Deployment Choice 3: TIBCO Enterprise Administrator 506
Review 507
Lab Overview 508
Lab P: Application Deployment 511
Deploying an Application 512

Unit 17 Using TIBCO ActiveSpaces 523
Topics. 523
Objectives 524
TIBCO ActiveMatrix BusinessWorks Plug-Ins 525
Fast Data and TIBCO ActiveSpaces 526
Data Grid: An Introduction 527
ActiveSpaces Enables TIBCO Products 528
ActiveSpaces and BusinessWorks 529
Installing TIBCO ActiveSpaces 530
ActiveSpaces Components 531
Plug-in Capabilities 532

Installing BusinessWorks Plug-In for ActiveSpaces 533
Plug-in Installation Pre-Requisites 534
ActiveSpaces Palette Details 535
Review 536
Lab Overview 537
Lab Q: Enabling ActiveSpaces 539
Setup Environment 540
Install and Verify Plug-in 542

Unit 18 Working with Data 547
Topics 547
Objectives 548
How a Data Grid Works 549
Data Grid Terminology - Spaces and Metaspaces 550
ActiveSpaces Components in Pictures 551
Tuple Definition 552
Designing Applications with ActiveSpaces Plug-in 553
Adding a Metaspace 554
Space and Space Fields 555
Space Properties 556
Defining a Space Connection 557
Creating a Space Tuple 558
Retrieving Space Tuple with Get Operation 559
Retrieving Space Tuple with Take Operation 560
Command-Line Administration 561
GUI Based Administration 562
Viewing Metaspace and Spaces 563
Viewing Space and Member Data 564
Review 565
Lab Overview 566
Lab R: Perform Data Operations 569
Add Data 570
Retrieve Data 581

Appendix A Common Operating System Tasks 587
Starting Applications 588

Viewing Status of Processes 589
Terminating a Process 589
Obtain IP address of your machine 590
Basic commands from the Terminal Window 590
Using gedit to open/modify files 591
Copy/paste/rename files in File Browser 591
Create and use File Browser bookmarks 592
Naming a Terminal Window 592
Downloading and Extracting Student Lab Files 593
Managing Environment Variables 594
Working with Java Code 596
Setting Up Email 597
Screen Resolution 598
Running Virtual Machines in VMware Player 599

Preface

This workbook has concept explanations and the hardcopy version has space for taking notes. We encourage you to participate in all of the course exercises because they reinforce course content. You will find the answers to any review questions in the relevant section of the workbook.

Your instructor will provide you with a course evaluation form. Please fill this out and return it to the instructor at the end of the course. Your feedback is crucial to the successful evolution of this course.

For more information about our educational curriculum, please visit the TIBCO Education Programs web site: **http://www.tibco.com/services/educational**.

Conventions Used in this Book

This book uses some special conventions. Any cross references to related portions of the material will be *italicized*. New terms and acronyms will be defined as they occur.

The following conventions may be used to flag important additional information:

WARNING: This is a warning. It is used for critical information that must be followed to avoid serious consequences.

Analysis: This is an analysis. It is used to provide you with a summary or explanation of the step(s) you have completed or will complete next.

Note: This is a note. It is used to provide relevant information that pertains to the current task.

Tip: This is a tip. It is used to provide helpful information.

Course Overview

Welcome to *M-BW6I: TIBCO ActiveMatrix BusinessWorks™ 6.x. Development.*

This course gives you hands-on experience with TIBCO ActiveMatrix BusinessWorks 6.x. This brief introductory unit provides an overview of the course, including learning objectives, agenda and lab environment.

Topics

- *Important Information,* page 2
- *Objectives,* page 3
- *Agenda,* page 4
- *TIBCO Software Featured,* page 5
- *BusinessWorks 6 Application Lifecycle,* page 6
- *Developing BusinessWorks Solutions,* page 7
- *BusinessWorks Application Details,* page 8
- *Design Time - Recap,* page 9
- *Developing Services,* page 10
- *Multi-Platform Support,* page 11
- *Test Your Skills Lab,* page 12

Important Information

- **Copyright Notice**

  COPYRIGHT© 2017 TIBCO Software Inc. All rights reserved. No part of this document may be reproduced in any form, including video recording, photocopying, downloading, broadcasting or transmission electronically, without prior written consent of TIBCO Software Inc. Copyright protection includes content in the material generated from software programs displayed on the screen, such as icons, screen displays, and the like.

- **Trademarks and Patents**

  All brand and product names are trademarks or registered trademarks of their respective holders and are hereby acknowledged. Technologies described herein may be covered by existing U.S. patents or U.S. patent applications that are in progress. Please consult the software product user documentation for details regarding applicable patents.

- **Confidentiality**

  Information contained in this material is confidential and proprietary to TIBCO Software Inc. and its affiliates and may not be modified, copied, published, disclosed, distributed, displayed or exhibited, in either electronic or printed formats without written authorization of an officer of TIBCO Software Inc.

- **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 OR NON-INFRINGEMENT.** 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.

- **Export**

  This material and related technical data are subject to U.S. export control laws, including without limitation the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations of other countries. You agree not to export or re-export this material in any form in violation of the applicable export or import laws of any jurisdiction.

TIBCO TIBCO Education Programs *© Copyright 2000-2016 TIBCO Software Inc.* © TIBCO Software Inc.

Important Information

Please review this important copyright information before proceeding with the course.

Objectives

- Review BusinessWorks features and components
- Explore development and deployment details
- Explain usage of JDBC drivers
- Examine variables, properties and profiles
- Use JMS messaging with BusinessWorks 6
- Discuss RESTful service details
- Describe types of sub processes and their usage
- Demonstrate inter process communication

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Objectives

The objectives for this course are listed above.

Agenda

1. Review development concepts
2. Business Case
3. RESTFul Services
4. External database connectivity
6. Properties and Variables
7. Using Java with BusinessWorks
8. BusinessWorks and JMS
9. Sub-Process invocation techniques
10. Conversations and event handlers
11. Compensations
12. Adapters and Plug-ins

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Agenda

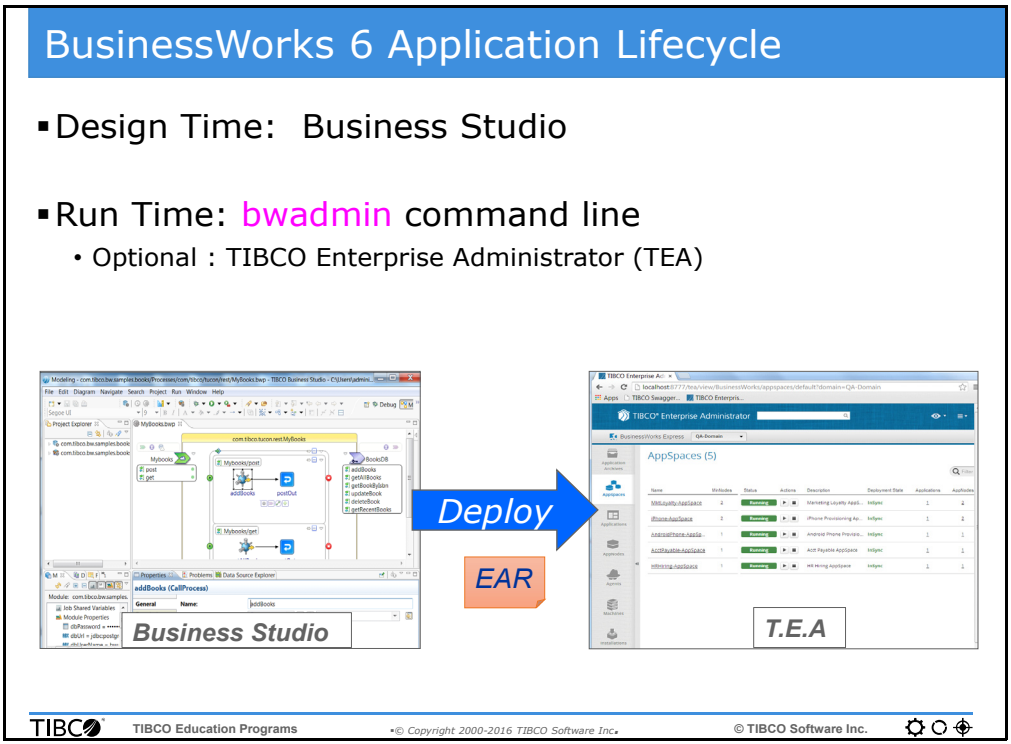
The topics above will be explored through a combination of instructor-led lectures and practical hands-on exercises.

TIBCO Software Featured

- Required:
  - TIBCO ActiveMatrix BusinessWorks 6.3.2
    - Studio for Designers 4.x
  - TIBCO Enterprise Administrator 2.2.0
    - Hotfix 3
- Additional software:
  - TIBCO Enterprise Message Service 8.3
- Third Party Software: Oracle JDBC drivers

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

TIBCO Software Featured



BusinessWorks 6 Application Lifecycle

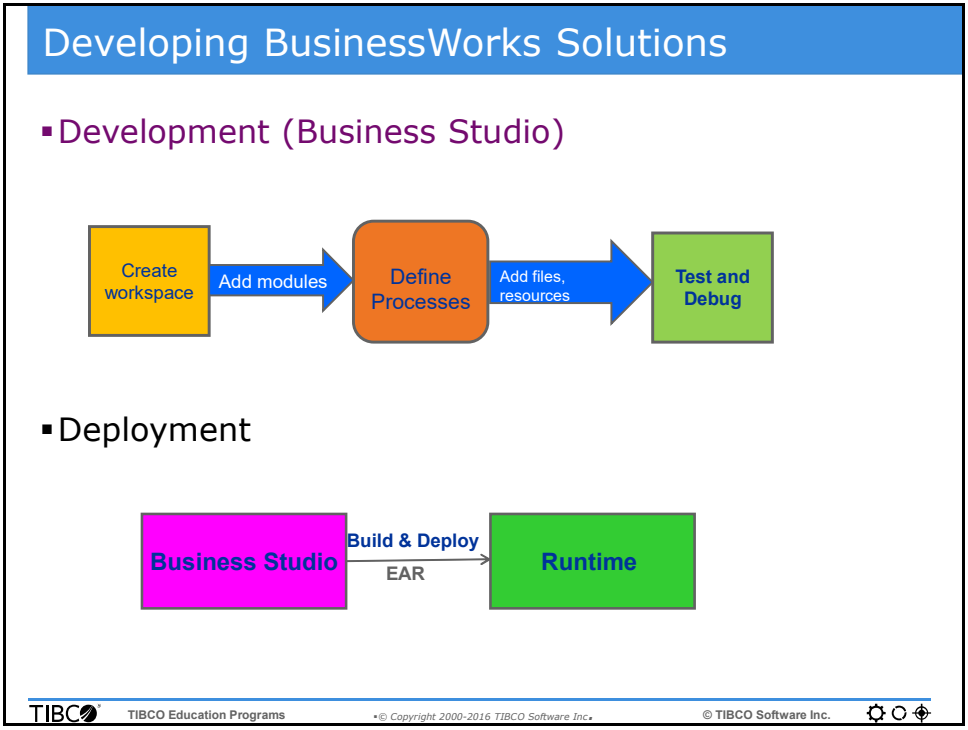
BusinessWorks applications are built using TIBCO Business Studio.

An application consists of one application module (1:1), and zero or more shared modules.

An application module contains the starter processes to run the application.

A shared module can contain other processes (sub-processes) and resources such as XML, XSD and WSDL files. It can also store connection information for external systems, such as HTTP connections, JDBC details or JMS settings.

An application is packaged as a single EAR file. An EAR file can be deployed to an AppSpace in local mode or Enterprise mode. Local mode is recommended for developers.



Developing BusinessWorks Solutions

The Application development methodology is shown above.

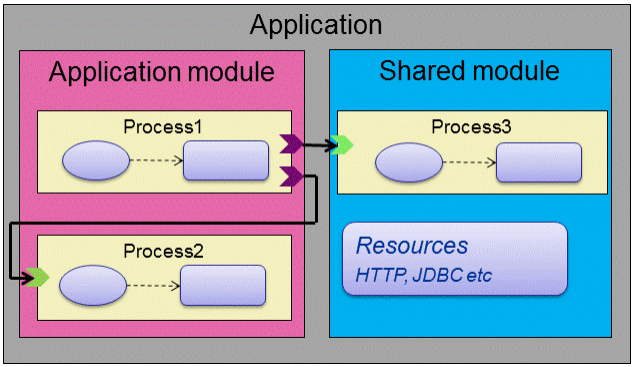
A workspace in Business Studio stores Application Modules and Shared modules. These modules are packages into applications (One Application = One Application Module).

Applications can be tested in Studio itself, using Run and Debug modes.

Once an Application is successfully tested, it is packaged into an EAR (Enterprise Archive) file. This file is deployed to an AppSpace. An AppSpace contains one or more AppNodes. The application is deployed to all AppNodes in the AppSpace.

BusinessWorks Application Details

- Applications developed using Studio for Designers
  - One Application Module
  - Zero+ shared modules



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

BusinessWorks Application Details

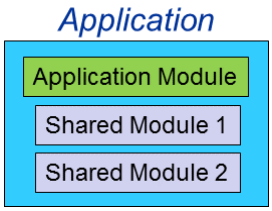
An Application Module and a Shared Module can store the same kind of project artifacts.

An Application module can hold starter processes, but a shared module cannot.

Resources, common processes and connections are normally created in a Shared Module, so that they can be used with other applications.

Design Time - Recap

- Workspace contains applications and modules
- Application Module
  - Packages
    - Processes
    - Resources
    - Schemas
- Process
  - Comprised of activities



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Design Time - Recap

Development in BusinessStudio is done in a workspace, which is a main folder containing one or more sub folders.

An application module or a shared module is a sub-folder in a workspace and is also referred to as a Project.

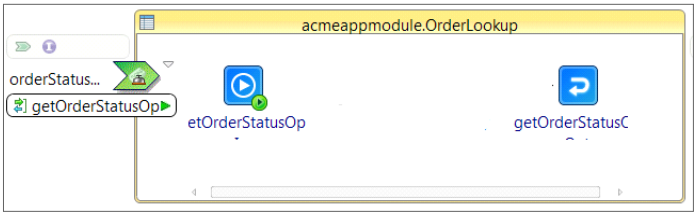
An application module is packaged along with any shared modules into a single application. Each application is a self contained piece of functionality, that is packaged into a single EAR file.

An application module or shared module contains processes and other resources required by the processes such as connections and schema files.

A process itself is a flow chart like representation of the business logic, which is executed when the application is run. The process flow consists of one or more activities which actually perform business functions as required.

Developing Services

- Supported: SOAP and REST
  - Implement multiple operations in one process
- Wizard-based development
  - Add WSDL, operations added
  - Response generated
    - Just provide implementation



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc. •10

Developing Services

Business Studio provides wizards to quickly build Web and RESTful services. For SOAP based web services, all you need is a WSDL file. The wizard provides starter process implementations for all operations in a WSDL file, with all inputs and outputs properly configured. A start process contains a preconfigured request and reply activity, so you simply provide the business logic.

Similarly, a RESTful service wizard uses an XSD file to select the parameters for a GET/POST/PUT or DELETE operation, and build a starter process for implementing the service. Multiple REST operations can be defined using a single process definition.

Multi-Platform Support

- Operating systems
  - Windows
  - Unix
  - Linux
- External application support
  - Adapters and plugins
  - Web Services
  - Messaging
    - EMS
    - FTL
    - Rendezvous

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Multi-Platform Support

BusinessWorks 6 is supported on the most common client platforms, including multiple Windows and Linux versions, and Mac OS X.

External application connectivity is provided by installing and using the correct Adapter or Plugin available for that product. Communication using TIBCO Messaging (Rendezvous, EMS or FTL) is also provided out-of-the-box.

Test Your Skills Lab

- Test and Deploy an Application
  - Application Module
  - Shared Module optional
- Design Process to:
  - Accept a CustID
  - Fetch details from customer table in the database

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Test Your Skills Lab

Notes

Notes

Lab A *Test Your Skills*

Overview

This lab provides students with additional practice working with TIBCO Business Studio. In this lab, students create a new business works application module using the wizard. They also define a process to send email to the customer with the order status and test the solution in test mode.

Objectives

- *Create Application Module using Wizard, page 16*

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3

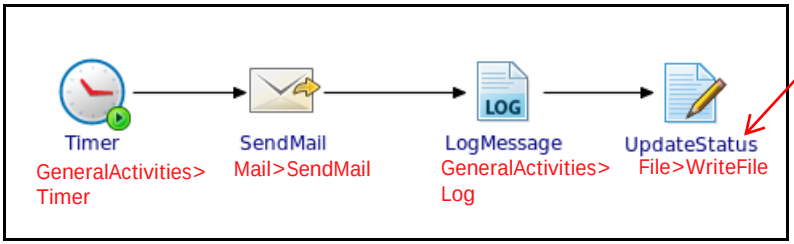
Directions

- Complete the exercises that follow

Lab A - Exercise 1: Create Application Module using Wizard

Overview

In this lab you will develop a simple Businessworks process that sends emails to the customer with the order status and test this process in Business Studio.



Names of A de las Activivities
WITHOUT SPACES

Steps

1. Configure Mozilla Thunderbird.

- Follow the instructions in *Setting Up Email,* page 597 to configure Mozilla Thunderbird.

2. Create new BusinessWorks Application module.

- ~~Launch Business Studio and use Workspace: `/home/tibco/bw6edu/workspace`~~

~~**Note:** Use the workspace that you created on Day1~~.

- Use wizard to define application module and application
- This also creates a default process

2.1. Open Business Studio and create the workspace
/home/tibco/bw6edu/TestSkillsLabA
2.2. Close the "Welcome" tab
Tip: The following Error may be displayed:
Failed to load JavaHL Library.
These errors were encountered: ....no libsvnjavahl-1 in java.library.path....
Solution: Click OK, and execute the following steps:
- Select Window> Preferences> Team> SVN> SVN interface Client
- Change default "javaHL(JNI) Not Available"
- Select SVNKit(Pure Java) SVNKit v1.7.9.9659 from the list
- Restart Business Studio
2.3. Click in the white area in Project Explorer
2.4. Right click>New>BusinessWorks Application Module
2.5. Name the Project:TestSkills, click Finish
2.6. In Project Explorer, go to TestSkills > Processes > testskills
Double click Process.bwp to open it
2.7. Right Click in the white area and select Add Activity > General Activities > Timer
In the General tab, confirm that Run Once is checked

3. Configure the process to send email.

- Add activities to the process and configure them
- Refer to process diagram on the top of the page

Activity SendMail
Properties, subtab Input:
from: "user01@edusvr"
to: "user01@edusvr"
subject: "Mail Test Your Skills"
***Drag any field to the expression
of the Body field to enable it. Overwrite.
e.g. string("Test eMail")

Activity UpdateStatus
Properties, subtab Input:
fileName: "/home/tibco/bw6edu/UpdateStatus.txt"
textContent: "Writing content to the file"

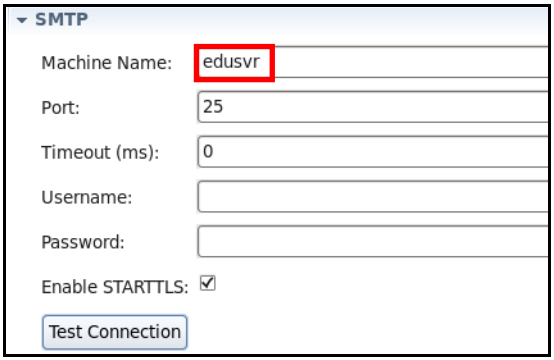
Activity Log
Properties, subtab Input:
message: "Message to Log"

4. Configure the SMTP resource for the SendMail activity.

4.1. Right click in TestSkills>Resources, New>SMTP Resource, Accept defaults, click Finish

4.2 Open the resource created SMTPResource go to its properties, modify:

- Host name: **`edusvr`**
- Port: **`25`**

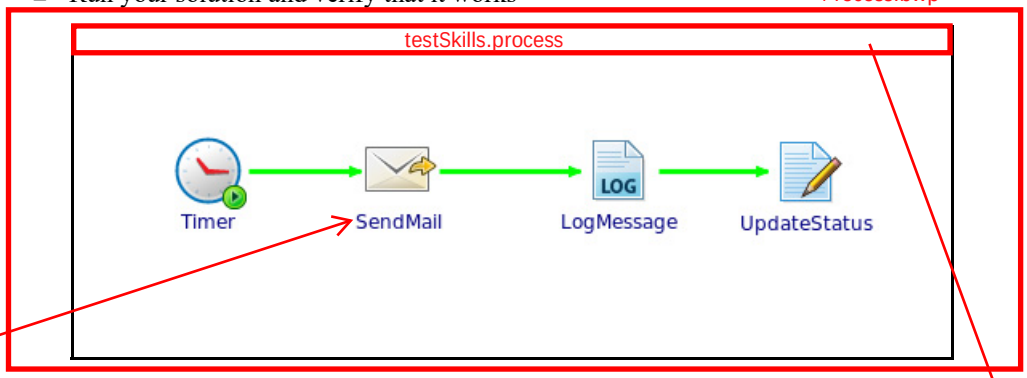


- Test connection and verify success

5. Test your application.

- Run your solution and verify that it works

Process.bwp



- Exit Business Studio

Para configurar la conexión SMTP a la Actividad SendMail, ir a las propiedades de la actividad Sendmail y en el tab General hacer click en el botón Create New Module Property para asociarla a la SMTP Property, después ir a TestSkills>Module Descriptors>Module Properties
En el Tab User, seleccionar la única Property: smtpProperty (si no aparece refrescar de alguna manera) y hacer click en el campo de la columna "default", hacer click en "..." y seleccionar el valor "testSkills.SMTPResource". Click OK. Con esto asignamos la conexión SMTP a la Property.

Además, puede ser necesario hacer click en la barra amarilla de la definición del proceso y abajo, en Properties, ir al subtab Process Properties y seleccionar el smtpResource Resource Template.

Cerrar BusinessStudio y Thunderbird

Notes

Unit 1 BWSX Business Case

This unit describes the business case for BWSX, which is implemented in this course.

Topics

- *Objectives,* page 20
- *Business Context for BW Super Express (BWSX),* page 21
- *Project Analysis Best Practices,* page 22
- *BWSX Business Process,* page 23
- *Query Order Process and Data Model,* page 24
- *Place Order Process,* page 25
- *Architecture Pattern,* page 26
- *QueryOrder: Process Pattern Mapping,* page 27
- *PlaceOrder: Process-Pattern Mapping,* page 28
- *BWSX Business Requirements,* page 29
- *BWSX Implementation Details,* page 30
- *Best Practices - Project Design,* page 31
- *Best Practices - Process Design,* page 32
- *Best Practices - Mapping Data,* page 33

Objectives

- Review business requirements for BWSX Ltd Co.
- Describe the solution architecture
  - Business processes
  - Architecture pattern
  - Process-Pattern mapping
- Discuss common best practices for project implementation

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Objectives

Business Context for BW Super Express

- BWSX wants to do a technology refresh
- Update key business processes
  - Enhance mobile shopping experience
- Selected processes for first phase:
  - Place order
  - Query order

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Business Context for BW Super Express (BWSX)

BW Super Express (BWSX) is a fictitious retail company. It wants to take advantage of the proliferation of smartphones and tablet devices. It wants to offer its customers unlimited shopping opportunities and introduce innovative marketing programs based on new technology.

In order to do this BWSX must update and automate its key business processes.

In this first phase, it has selected Place Order and Query Order business processes for update.

Project Analysis Best Practices

- Design using standard architecture practices
- Model the business requirements:
  - Business processes
  - Architecture pattern
  - Process-pattern mapping
- Develop exception handling scenarios
- Analyze load requirements
- Understand deployment environment
  - Resources, users, limitations

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Project Analysis Best Practices

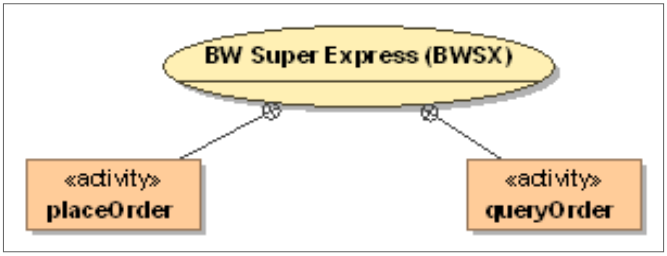
TIBCO has over 20 years experience in implementing integration projects. From this experience, TIBCO has created several best practices and recommendations. The most common recommendations are mentioned here.

A business process can be mapped to a classic architecture pattern for getting a quick understanding on the size and scope of the implementation. There are a few key architecture patterns, and more information can be found in the TIBCO Architecture Training series (ARC701, ARC702, ARC704).

When using TIBCO BusinessWorks 6, please remember to use modular development techniques. This means that a single large business process should be analyzed and broken up into smaller reusable modules where necessary.

BWSX Business Processes

- Context for first phase
  - Placing an order
  - Querying an order
- Map business *processes* to architectural *patterns*
  - Details in *ARC701: TIBCO Architecture Fundamentals*



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

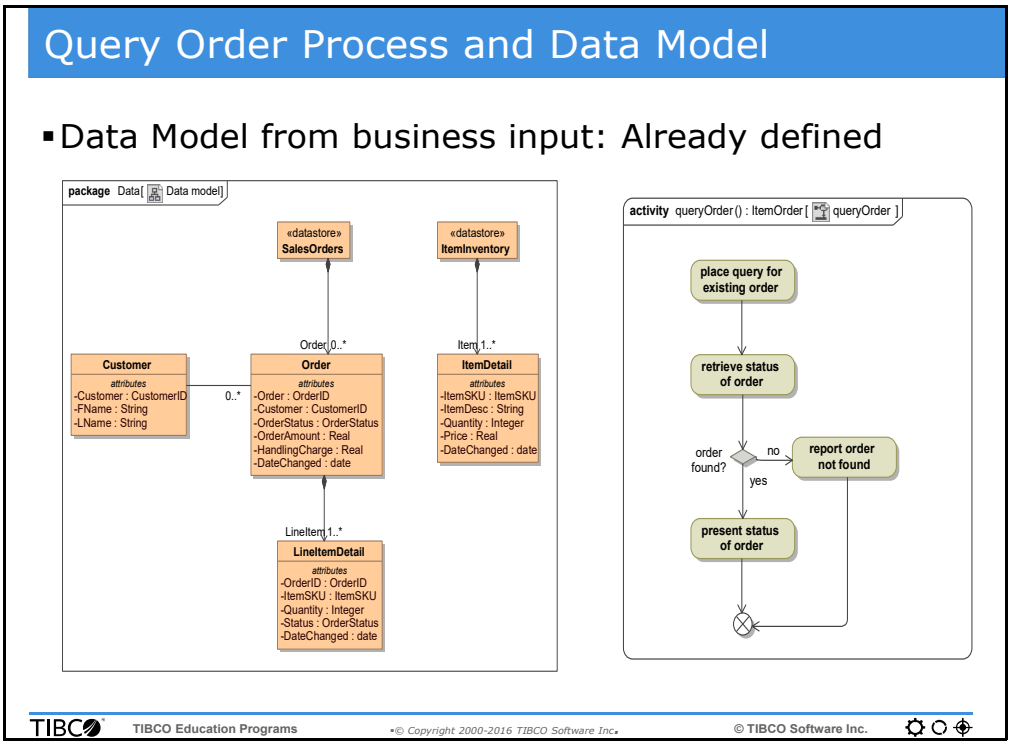
BWSX Business Process

The exercises in this course focus on the collaboration around two primary activities:

- Place Order
- Query Order

These activities and their associated data are explored in the following pages.

.

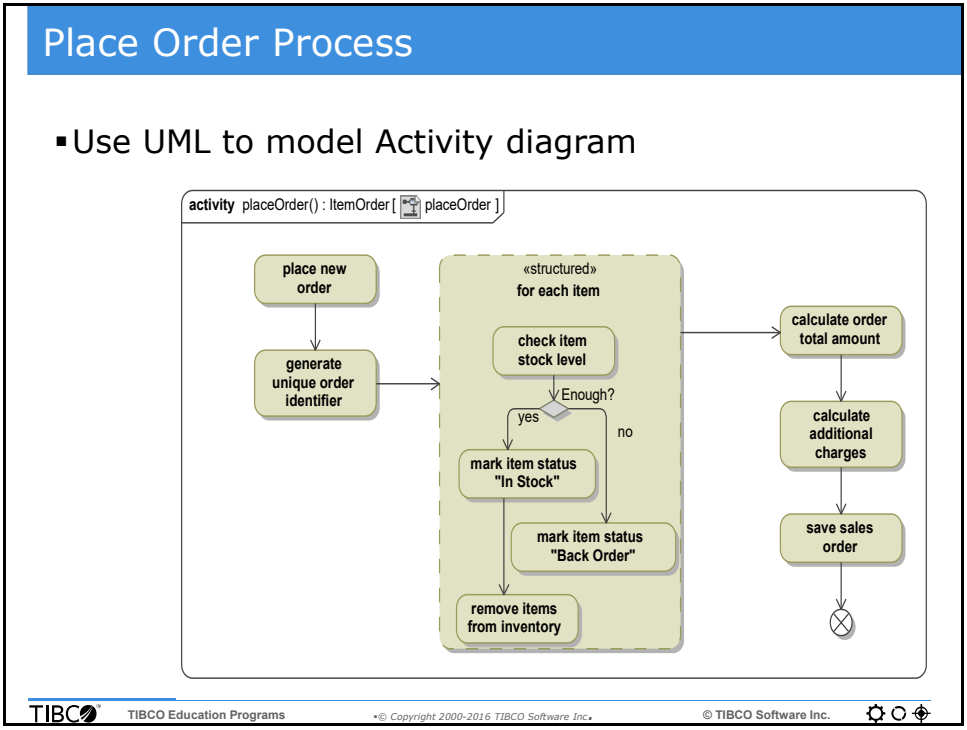


Query Order Process and Data Model

An order is placed by a customer and consists of various line items. The line items are held in inventory until selected to fulfill an order. The essential data and their associations are shown above.

The queryOrder process returns the status of previously submitted order, e.g. order completed. The activity diagram above shows this simple process.

.



Place Order Process

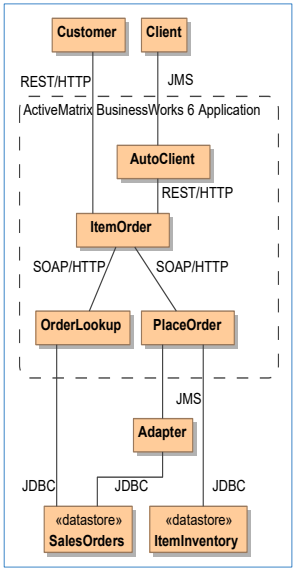
Placing an order is slightly more involved,

- It generates a unique identifier to represent the order
- The order contains line items that must be checked against the current stock inventory
- The line items must be taken from stock ready for shipping
- It calculates the total cost of the order, including handling charges, ready for accounting
- It saves the order for audit and tracking purposes

These activities are shown in the activity diagram above. Note that for simplicity the process does not consider items that are not held in stock.

Architecture Pattern

- Main Entities
  - REST interface
    - ItemOrder
  - SOAP interface
    - OrderLookUp
    - PlaceOrder
- Transports
  - HTTP
  - JMS
  - JDBC
- Datastores
  - Sales Orders
  - Item Inventory



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

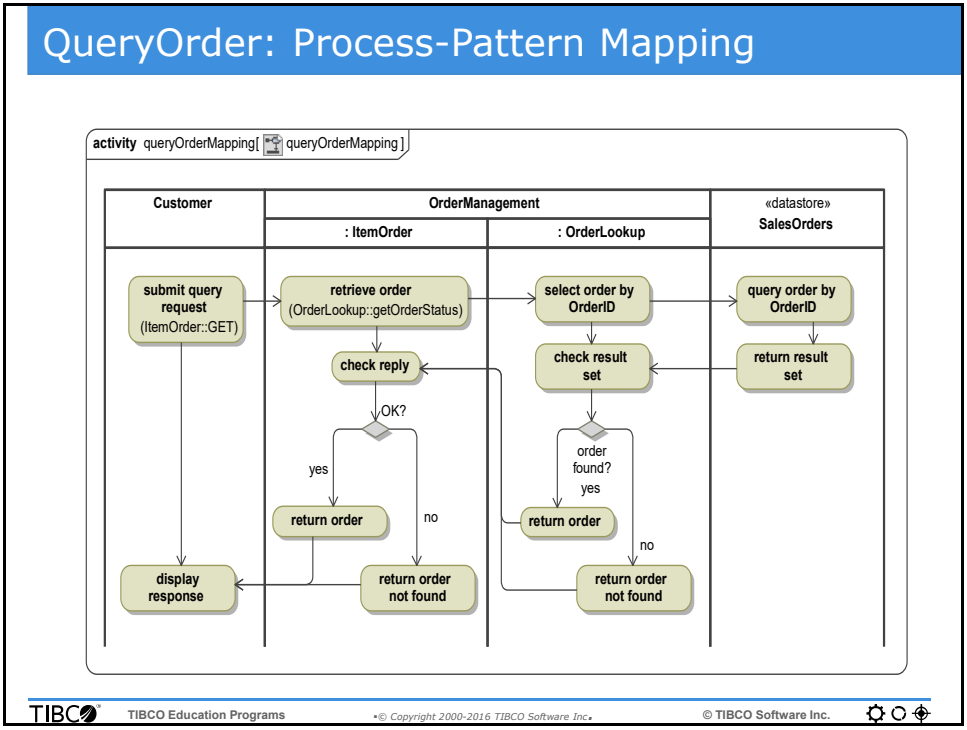
Architecture Pattern

The solution exposes a set of interfaces that can then be accessed by a variety of devices.

This diagram shows the main entities that are providing these interfaces:

- ItemOrder offers a REST interface
- AutoClient offers an XML/JMS interface
- OrderLookup and PlaceOrder offer a web service interface

There are two datastores: one for stock inventory and one for sales orders. These are accessed using JDBC communication.



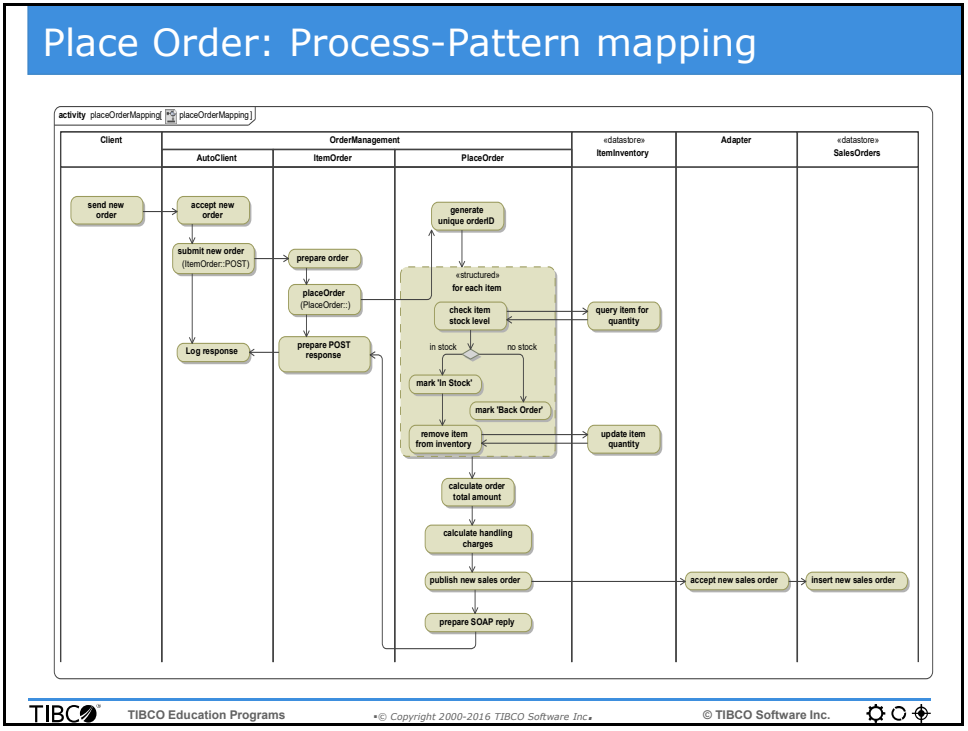
QueryOrder: Process Pattern Mapping

By mapping the process activities onto the architecture pattern, the work done by each entity is clearly shown.

The diagram above shows the mapping for queryOrder process.

Compare this with queryOrder activity diagram and you will see that some activities are now spread across several entities. For example, ‘retrieve status of order’ is spread across ItemOrder, OrderLookup and SalesOrders entities.

You will implement these components in the lab exercises that follow.



PlaceOrder: Process-Pattern Mapping

Here is the process-pattern mapping for the placeOrder process.

Compare this diagram with the process activity diagram to see how the work is spread across the entities.

You will implement these components in the lab exercises that follow.

BWSX Business Requirements

- Order Entry
  - Batch input
    - Using JMS
  - Web input
    - Using HTTP
    - RESTful service
  - Log all orders
- Integrate inventory DB
  - Single transaction
- Store customer orders
  - In sales order DB
  - Implemented later

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

BWSX Business Requirements

The BWSX business requirements can be met by,

Creating RESTful interfaces

- Place order
- Query order

Creating messaging interface

- Place order

Integrating with existing datastores

- Stock Inventory - coordination must be transactional to maintain consistency
- Sales Order

Since we are going to implement the pilot project, our application modules will use the keyword *ACME*.

BWSX Implementation Details

- Using TIBCO ActiveMatrix BusinessWorks
  - Design
    - WOA (Web Oriented Architecture) infrastructure
      - REST, SOAP
    - Built-in activities for process flows
      - JMS, JDBC, HTTP, REST
  - Deploy
    - Simple promotion from design to run-time environment
      - TIBCO Enterprise Administrator
  - Manage
    - Rules which automate runtime management
      - TIBCO Hawk

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

BWSX Implementation Details

Remember to use the capabilities of BusinessWorks 6 to provide flexibility in your implementation.

BusinessWorks 6 supports WOA implementations out of the box, so consider the use of RESTful services where they are appropriate. All TIBCO Messaging is supported, so consider using the right messaging for the right use case. You may need ultra-fast multi-thread processing (FTL), persistence and storage of logs and errors (EMS) and high fan out (Rendezvous).

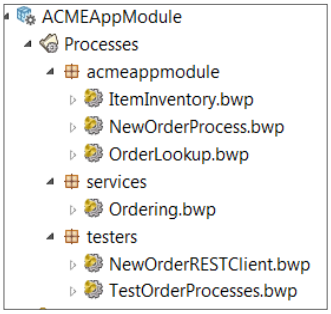
Many adapters and plug-ins for the most popular third party applications are provided by TIBCO. Consider using them before you decide to write your own connectivity application using Java code or C++ code.

Enterprise deployment uses TEA, so it is a good idea to have your Administrators gain familiarity with that environment. Similarly, administrators can learn TIBCO Hawk, to design robust monitoring for your applications.

Note: Rules which automate runtime management using TIBCO Hawk is covered in BW306 class.

Best Practices - Project Design

- Modular development
  - Build application modules
    - Service based
- Use Shared Modules
  - Common routines, XSD, connections
- Use multi-file approach for team development
- Use Revision Control System (RCS)
  - Secure, versioned development
- Use Project Template
  - Quick start development



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

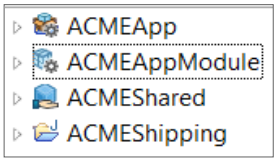
Best Practices - Project Design

Business Studio is an Eclipse based environment and it can easily connect to a version control system such as SVN.

You can also connected to an automated tool such as Maven or Ant for building EAR files and deploying changes.

Best Practices - Process Design

- Modular Development
  - *Shared modules* to share schemas/common routines
  - Break large processes into smaller sub-processes
  - Maintain readability



- Transitions: Use Boolean expressions
  - Choose process execution path
- Log and checkpoint business-critical data
- Name resources ***exactly* as indicated in the book**
  - Saves you time and helps troubleshooting
- Apply configuration settings and Save project frequently

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Best Practices - Process Design

Use Modular development practices when using BusinessWorks 6.

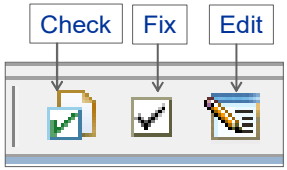
The application module usually contains the main process. It invokes sub-processes from the shared module when necessary. All resources and connections are also stored in shared modules.

You may have multiple shared modules, separated by the type of content (resource, connections and sub processes).

Also remember to save your changes frequently, as this commits changes from memory to disk. Remember to use the project clean option, if you see any inconsistency in development.

Best Practices - Mapping Data

- XPath Features
  - **ALL** output data from previous activities available to current activity
  - Strong data typing
  - Wizards to help complex mappings
  - **Fix-it** button to correct input definitions



- Use *XPath Editor* Functions in expressions
  - Drag-and-drop instead of typing
  - Edit XSLT directly
- Validate resources and debug errors
- Perform unit testing for all scenarios including errors

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Best Practices - Mapping Data

In a BusinessWorks process, all output data from an activity is available to all activities downstream in the process.

This reduces the need to pass values between activities and to keep multiple copies of data.

This behavior is quite different from a Java development environment, where data must be carefully passed around between activities.

The XPATH editor allows you to define data transformation using XPath 2.0 functions, and also a few TIBCO functions. Strong data typing is enforced, so use the mapper check and repair button to resolve any type casting issues. There are a few wizards available when mapping data structures, so use them to reduce the mapping effort. Always drag and drop where possible, instead of typing in variable names.

If you are familiar with XSLT, you can directly edit the XSLT generated by your XPath mapping.

If you expect specific values in your process, it is always advisable to design and run edge cases, where data is not within the range. This ensures that you built exception handling in the process.

Notes

Unit 2 Building RESTful Services

This unit describes the capabilities of Business Studio in building and testing RESTful services. It examines the basic concepts of a REST architecture.

Topics

- *Objectives,* page 36
- *REST Fundamentals,* page 37
- *REST Support in BusinessWorks,* page 38
- *REST Architectural Elements,* page 39
- *REST Architectural Elements and Implementations,* page 40
- *Defining a RESTful Service,* page 41
- *BusinessWorks REST Wizard,* page 42
- *Completing REST Services Definition,* page 43
- *Filling in the Details,* page 44
- *REST Operation Details,* page 45
- *Testing a RESTful Service,* page 46
- *Business Case Implementation,* page 47
- *Process-Pattern Mapping - Lab B,* page 48
- *Interface Specification - Lab B,* page 49
- *Review,* page 50
- *Lab Overview,* page 51

Objectives

- Explain fundamentals of REST
- Discuss architectural constraints for REST
- Describe how to define a RESTful service
- Examine operation configuration details
- Explain how to test RESTful services

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives

REST Fundamentals

- REST: Representational State Transfer
  - Architectural Style – not a standard
- Why REST: Build client friendly distributed systems
- Architectural constraints
  - Client-server
  - Uniform interface between components
  - Stateless
  - Cache (client-cache)

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

REST Fundamentals

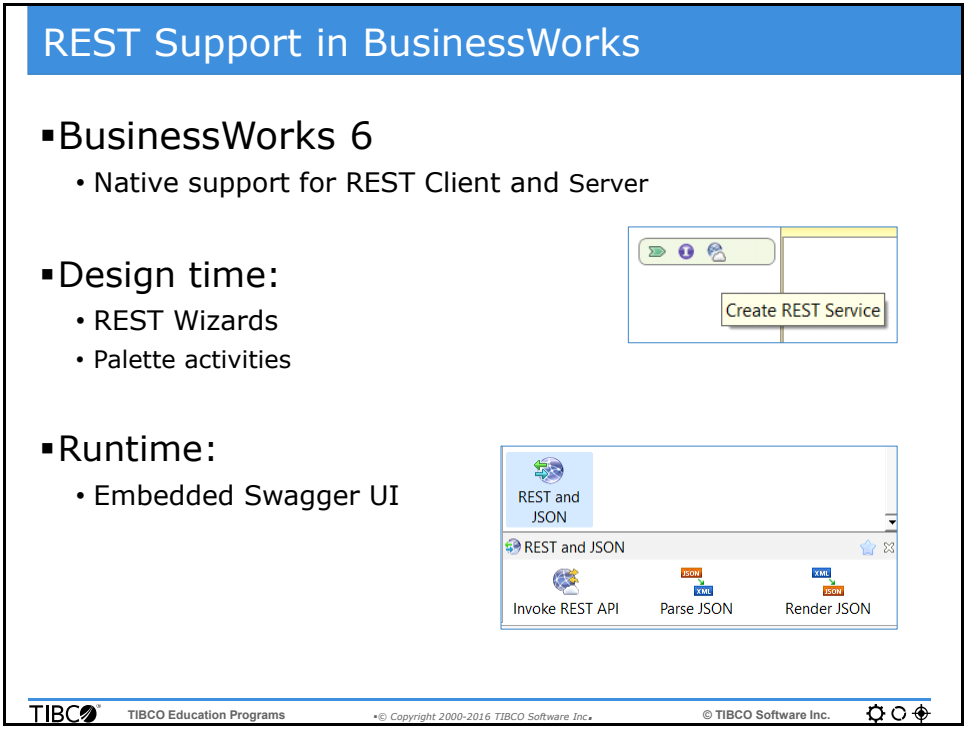
REST(sometimes spelled "ReST"): An architecture style for designing networked applications. It relies on a stateless, client-server, cacheable communications protocol -- and in virtually all cases, the HTTP protocol is used.

RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.

Client-Server: By separating client and server, the components can evolve independently, thus supporting the Internet-scale requirement of multiple organizational domains.

Stateless: Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client. Results in visibility, reliability, and scalability.

Cache: The client-cache-stateless-server style is used. A cache mediates between client and server and reuses responses which improves efficiency.

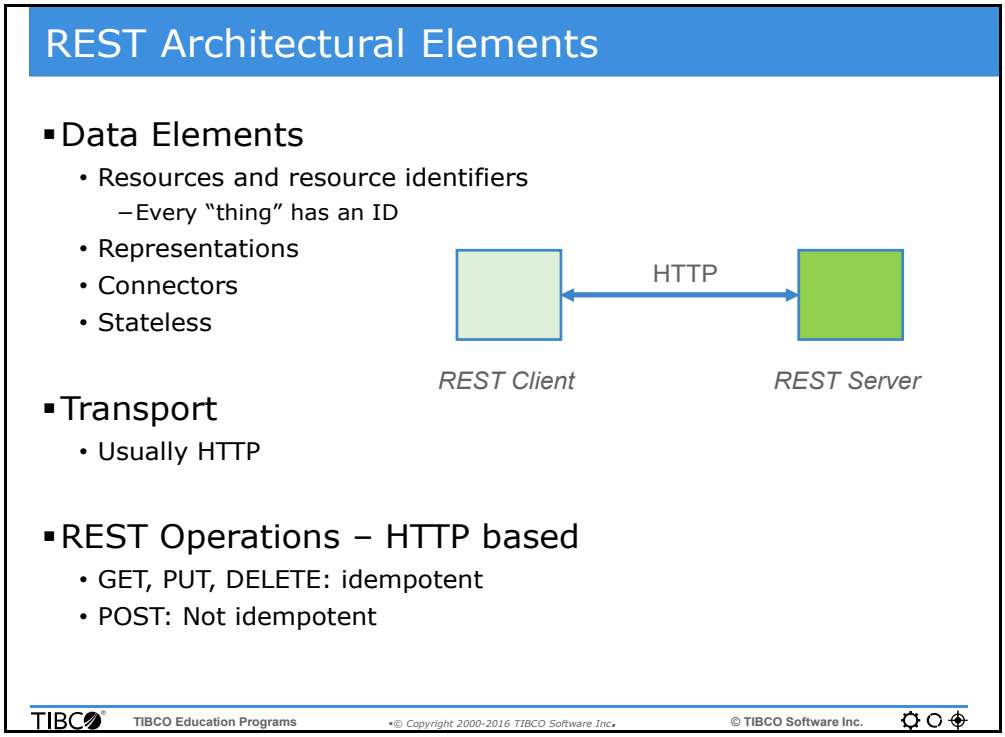


REST Support in BusinessWorks

Support for REST in BusinessWorks has evolved over the years.

BusinessWorks 5.x offers REST support through the installation of an additional plug-in. This plug-in creates a REST (HTTP/JSON) wrapper around an existing process and exposes it using the HTTP transport. Similarly, it provides a REST client which can interact with an HTTP/ JSON RESTful service. JSON content is converted to XML using Badgerfish or normal conversion mechanisms.

BusinessWorks 6 supports REST in its native form. Using wizards, you can define a RESTful service from the ground up, and add the BusinessWorks activities to complete the service definition. There is no need to define a wrapper process. You can also implement multiple REST operations in a single process. Additional features include: custom error codes, better JSON to XML conversion, custom status codes, path parameters, key-value parameters and custom HTTP headers.



REST Architectural Elements

Data Elements, which include resource, resource identifier, representation, representation metadata, resource metadata, and control data.

REST components perform actions on a resource by using a representation to capture the current or intended state of that resource and transferring that representation between components. A representation is a sequence of bytes, plus representation metadata to describe those bytes.

Connectors represent the activities involved in accessing resources and transferring representations. REST encapsulates different activities of accessing and transferring representations into different connector types.

GET: retrieve a representation of the resource

PUT: create a resource at a known URI or update an existing resource by replacing it

POST: create a resource when you don't know the URI, partial update of a resource, invoke arbitrary processing

DELETE: delete a resource

REST Architectural Elements and Implementations

- REST implementations commonly use:
  - HTTP (transport)
  - JSON (data format), sometimes XML
- Architectural Elements
  - Data Elements
    - Resources and resource identifiers
      - Every "thing" has an ID
  - REST Operations – borrowed from HTTP
    - GET, PUT, DELETE: idempotent
    - POST: Not idempotent

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

REST Architectural Elements and Implementations

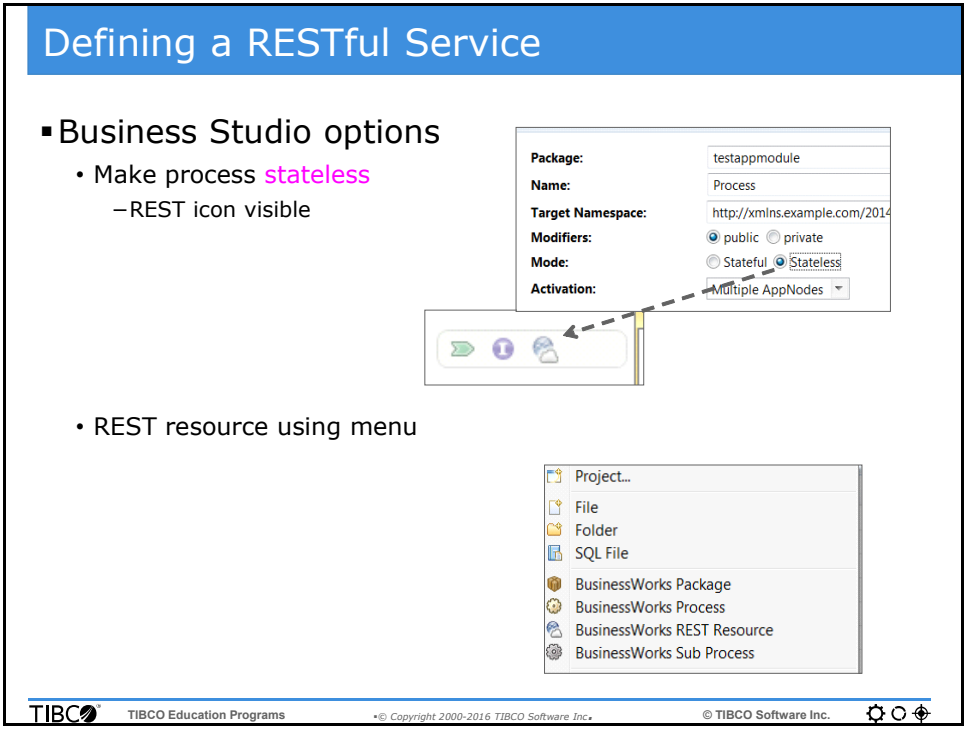
The adoption of REST by various applications and providers has dictated the de-facto standards that are being used with this architecture today. HTTP is most commonly used as a transport for RESTful services, and most RESTful data is sent and received in JSON format. In special cases, the data is also represented as XML. Data Elements have IDs. REST components perform actions on a resource by using a representation to capture the current or intended state of that resource and transferring that representation between components. A representation is a sequence of bytes, plus representation metadata to describe those bytes. Connectors represent the activities involved in accessing resources and transferring representations. REST encapsulates different activities of accessing and transferring representations into different connector types.

GET: retrieve a representation of the resource

PUT: create a resource at a known URI or update an existing resource by replacing it

POST: create a resource when you don't know the URI, partial update of a resource, invoke arbitrary processing

DELETE: delete a resource



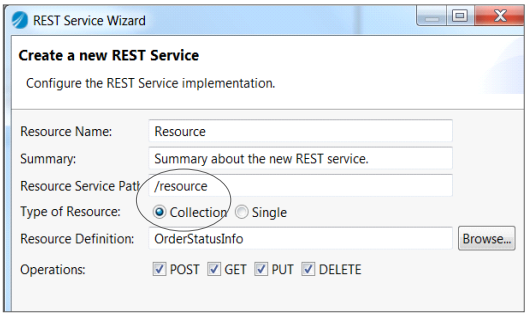
Defining a RESTful Service

You can access the process details by clicking on its boundary. Next, you can change the mode to stateless, which allows the process to be converted into a RESTful service. After doing this, an additional icon is displayed in the process service context. Clicking on this icon brings up the REST service wizard.

You can also right-click on an application module menu to bring up a different wizard which allows you to define a REST resource. Doing this automatically creates the process required to implement the RESTful service.

BusinessWorks REST Wizard

- Resource path
  - Use nouns
  - Examples:
    - /ItemOrder
    - /Customer
    - /Product



- Choose operations to be implemented

- XML Schema (XSD): For operation input/output
  - *Resource definition*

TIBCO TIBCO Education Programs *© Copyright 2000-2016 TIBCO Software Inc.* © TIBCO Software Inc.

BusinessWorks REST Wizard

ActiveMatrix BusinessWorks currently allows the following HTTP operations to be performed on resources: GET, PUT, DELETE, and POST.

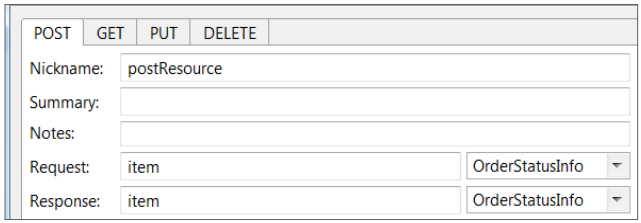
Collection vs.Single: For example, Orders could be a Collection Resource and an Order represents a Single Resource. The Schema Element used in the Resource Schema field determines the type of resource.

When using REST, resource paths are generally defined as nouns such as Order, Customer instead of using operation names such as GetCustomer, or SubmitOrder. The reason is that the single resource path (/ItemOrder) is able to support multiple operations by specifying the operation name as part of the HTTP request. For example /ItemOrder can support both GET and POST operations.

The REST wizard presented by Studio allows you to specify the request and response schemas for each operation that will be implemented by you. A tab represents each operation, and within it, you can specify request and response details as shown.

Completing REST Services Definition

- For each operation, update schema
  - Request schema
  - Response schema
- Pre-create schema or create new schema
  - Uses Eclipse schema editor



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Completing REST Services Definition

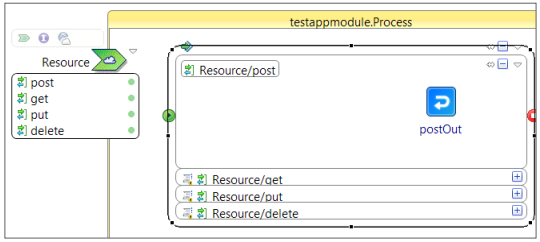
After you choose your REST operations to be implemented, you must click Next.

In this window, you must choose the right input and output schemas to be used with each operation.

One or more tabs are presented, and this depends on the number of operations you have chosen to implement in the previous screen. You can create new schemas in this screen as well.

Filling in the Details

- REST wizard defines response activities
  - Based on selected operations
- To do:
  - Implement operation
  - Provide response(s)



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

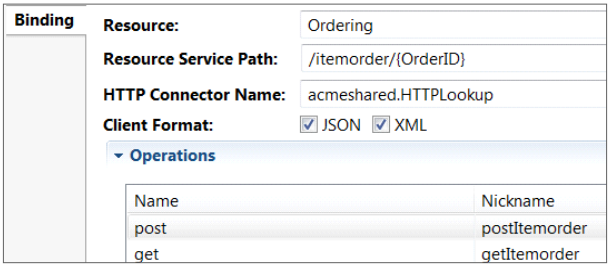
Filling in the Details

Once the REST wizard is finished, you will see a process diagram similar to the one shown above. The process contains multiple constructors, and each constructor implements a REST operation. The incoming request is diverted to the specific constructor at runtime.

Inside each constructor, a reply operation is already added, and this operation defines the schema needed to complete a successful response. Thus, enough starter details are provided, and you can now build the process required to send the appropriate response.

REST Operation Details

- View in Module > Components
  - Add parameters using { }
- Example:
  - URL : http://localhost/ItemOrder/123
    - If operation = GET, get details for orderid = 123
    - If operation = POST, submit new order for id = 123



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

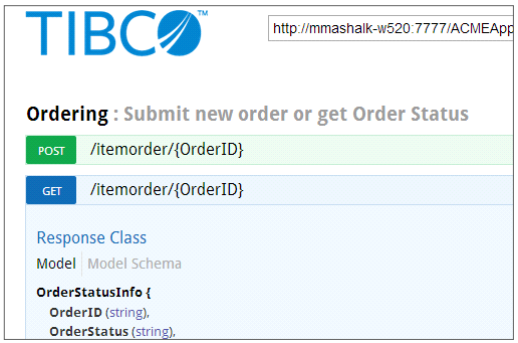
REST Operation Details

You can add parameters to a REST operation easily. To do this, you must locate the REST component definition in your module. The business logic and transport layers are decoupled, and hence the parameter definition is specified in the transport section of your module.

Go to module components and locate your REST operation. You can then add parameters using the {} tags as delimiters. You can add multiple parameters. Generally speaking, this method of submitting data is not recommended, since it makes the URIs inflexible.

Testing a RESTful Service

- Business Studio includes Swagger UI
  - Launched when RESTful service is run
  - URL: http://localhost:7777/AppName
- Optional
  - Test using browser



TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. • © TIBCO Software Inc.

Testing a RESTful Service

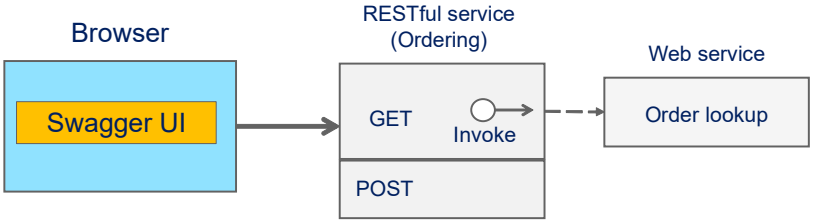
Business Studio uses the Swagger UI for REST documentation and testing. For more information about Swagger, go to: https://github.com/wordnik/swagger-ui.

Swagger UI is a dynamically generated browser-based documentation from a Swagger-compliant (REST) API. In Business Studio, the Swagger UI runs on port 7777. To access the Swagger UI, use the URL: http://host:7777/AppName.

After launching the Swagger UI, you can test all operations that you have defined for your RESTful service. Since the UI is dynamically generated, your UI options depend on your implementation. You can receive responses in JSON or XML format, and you can provide input data where necessary.

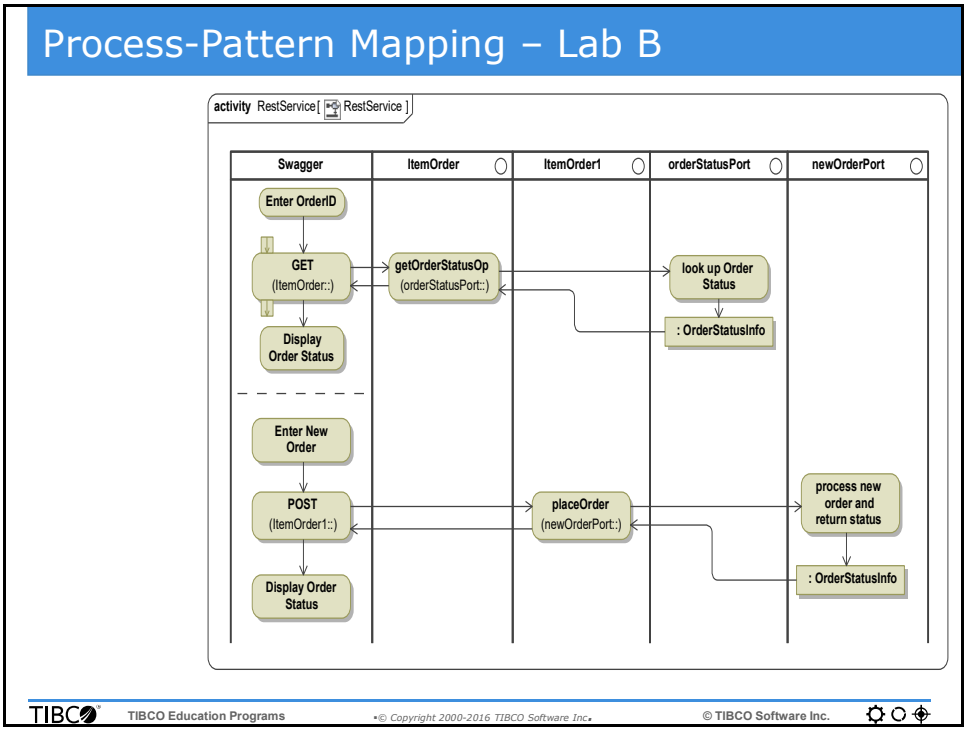
Business Case Implementation

- Define RESTful Service
  - GET operation invokes web service
  - POST operation returns dummy values
- Test using built-in Swagger UI
  - Browser access

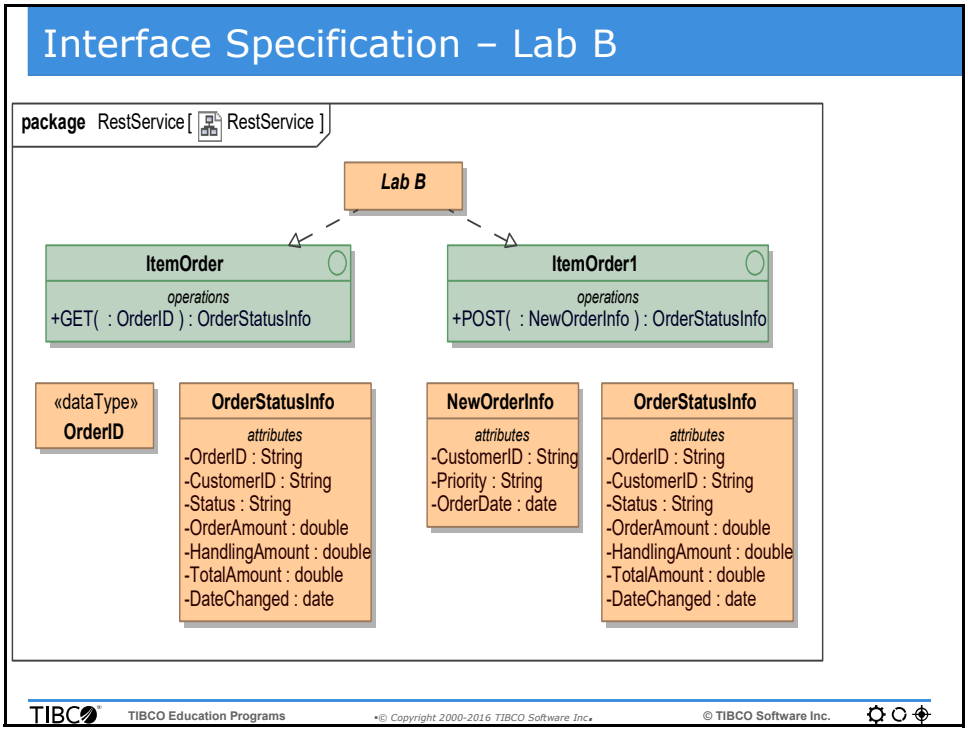


TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Business Case Implementation



Process-Pattern Mapping - Lab B



Interface Specification - Lab B

Review

- REST is an architecture style not a standard
- Key constraints: uniform interfaces, stateless
- REST Transport: HTTP, Format = JSON
- REST wizard provided for defining services
- REST Operations: GET, PUT, POST, DELETE
- Swagger UI provided for testing RESTful services

TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Design a RESTful service
  - Implement POST operation
  - Use lookup process to implement GET operation
- Test Service
  - Use Swagger UI
  - Provide parameters for operations

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab B Creating a RESTful Service

Overview

In this lab you will define a RESTful service that uses the web services you have defined in the earlier lab. You will use the REST Wizard. You will test the services using Swagger UI. You will also export your solution as a ZIP file.

Objectives

- *Define RESTful Service, page 54*
- *Using Swagger UI,* page 61

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Completion of the previous labs

Directions

- Complete the exercises that follow

Lab B - Exercise 1: Define RESTful Service

Overview

In this exercise you will create a RESTful service using the wizard provided by Business Studio. You will configure a GET and POST operation.

Steps

1. In TIBCO Business Studio, add a new package named services to your ACMEApp module.

- Launch Business Studio and use Workspace: `/home/tibco/bw6edu/workspace`
- Browse to *ACMEAppModule* > ***Processes***
  - Right-click and add a new ***BusinessWorks Package***
  - Name: **`services`**
- Click **Finish**

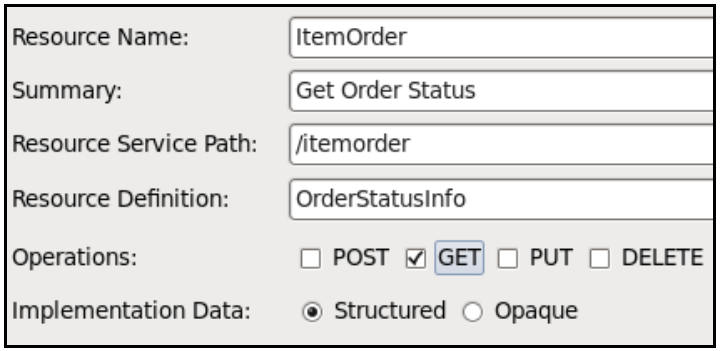
2. Define a new stateless process named Ordering inside this package.

- Right-click on **services** package and add a new ***BusinessWorks Process***
  - Process Name: **`Ordering`**
  - Keep other defaults (**public, empty process**)
  - Click **Finish**
- Click inside process boundary (yellow border) to view its properties
  - In the *Advanced* tab: Set **Mode** to **Stateless**
  - **Save** changes
- Notice a new icon appears in the process input options, for REST Service

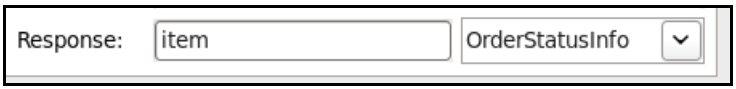
3. Configure process as a RESTful service with GET operation.

- Click on the **Create REST Service** icon

- Fill in these details
  - Resource Name: **`ItemOrder`**
  - Summary: **`Get Order Status`**
  - Resource Service Path: **/itemorder** (filled automatically)
  - Resource Definition (Browse): **ACMEOrderItemSchema/OrderStatusInfo**
  - Operations: **GET** (checked)
  - Implementation Data (default): **Structured**
  - Confirm before you continue



  - Click **Next**
- In the next screen, configure the GET operation in the *GET* tab
  - The operation has no input (request)
  - It is set to return **OrderStatusInfo**

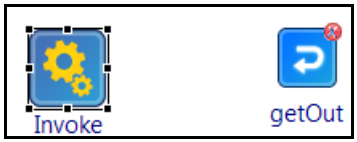


  - Verify this and click **Finish**

4. Add activity in the GET operation to invoke OrderLookup process.

ACMEAppModule>Processes>acmeappmodule>OrderLook.bwp

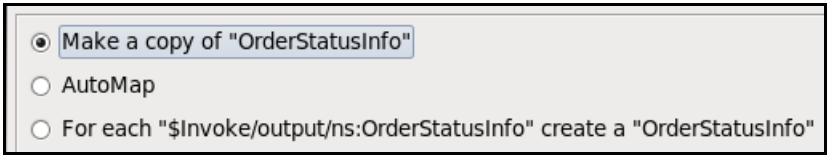
- In *Project Explorer* (left hand window), locate **OrderLookup.bwp** (process)
- Drag this process next to the **getOut** activity
  - This creates the invoke activity as shown:



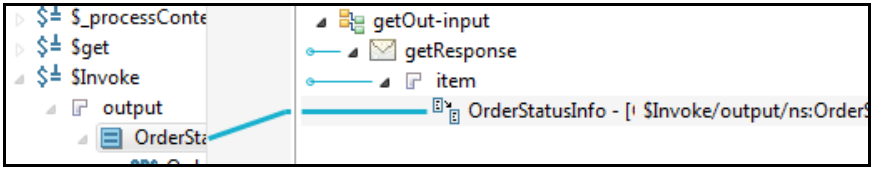
  - Connect **Invoke** to **getOut** activity
- Choose *Invoke* > *Properties* > ***Input***
  - Set OrderID = **`"TESTORDER123"`**

5. Map results returned from Invoke activity to getOut activity.

- Choose *getOut* > *Properties* > ***Input***
  - Drag from *Invoke* > *output*>***OrderStatusInfo*** to *getResponse* > *item* > ***OrderStatusInfo***
  - In the mapping wizard that pops up, choose **Make a Copy of** option



  - Click **Finish**

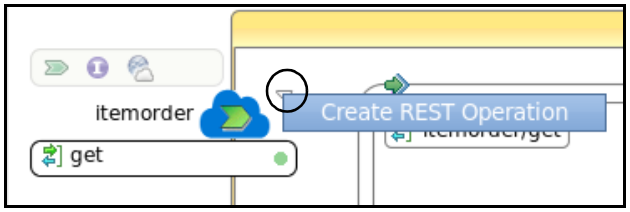


- The mapping indicates the copy of icon, **Save** your changes

Analysis: You have just created a RESTful service with a single operation: GET, which will invoke the Web Service that you created on Day 1.

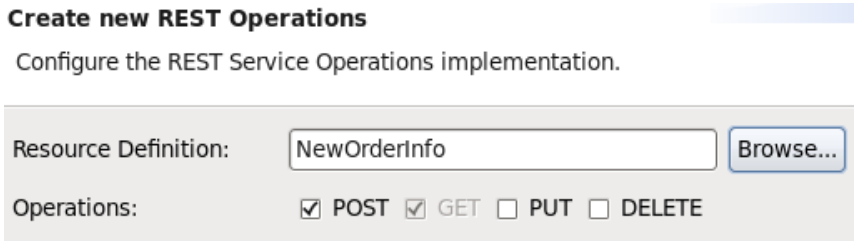
6. Add POST operation.

- Select itemorder service, click the down arrow and select Create REST Operation

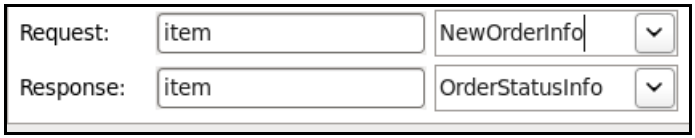


- Verify that the POST operation is selected
  - Resource Definition: Browse and select **NewOrderInfo**

ACMEShared/Schemas/ACMEOrderItemSchema.xsd

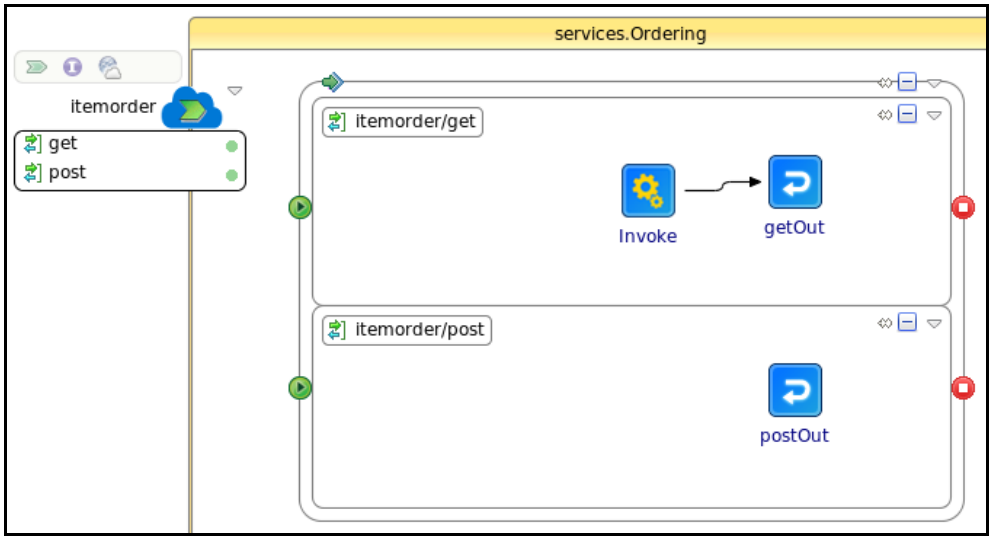


- Click **Next**
- In the next screen, the POST operation details are shown. Change as follows:
  - For Response: Click down arrow, and select **XSD element**
  - Now select *ACMEOrderItemSchema.xsd* > ***OrderStatusInfo*** and click **OK**
  - Results appear as shown below, click **Finish**

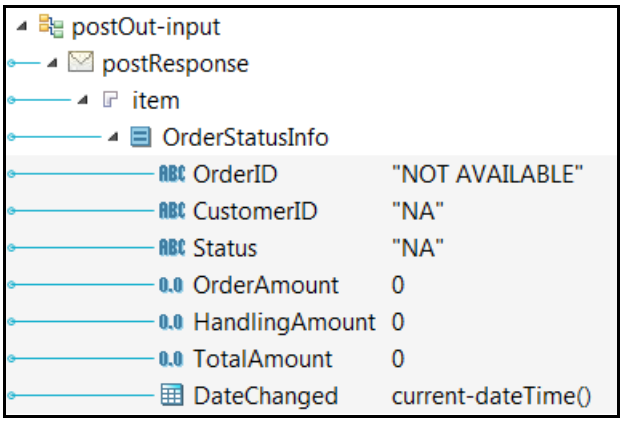


7. Verify creation of both operations, and update the POST response to send hard coded values.

- The process looks as shown below:



- Click **postOut** activity, then *Properties*
  - In the *Input* tab, set values for the fields as shown:

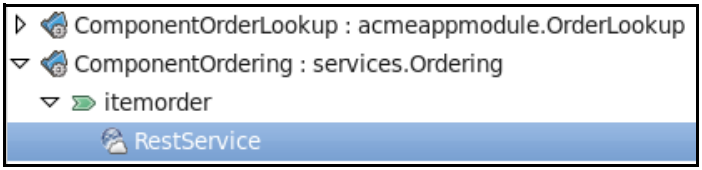


  - **Save** your changes and all problems with this activity are resolved

Note: This process is implemented in the integration class, following this one.

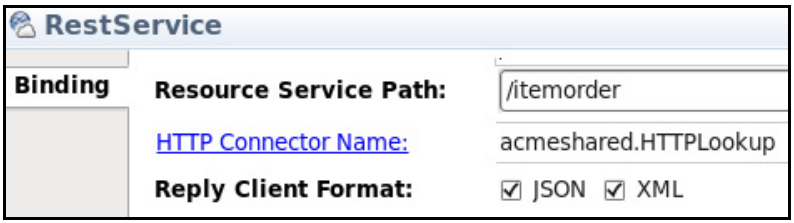
8. Locate the HTTP connector for your REST service.

- In *Project Explorer*, locate *ACMEAppModule> Module Descriptors* > ***Components***
- Double-click on Components to view Module details
- Expand services.*Ordering* > *Itemorder* > ***RestService*** as shown:



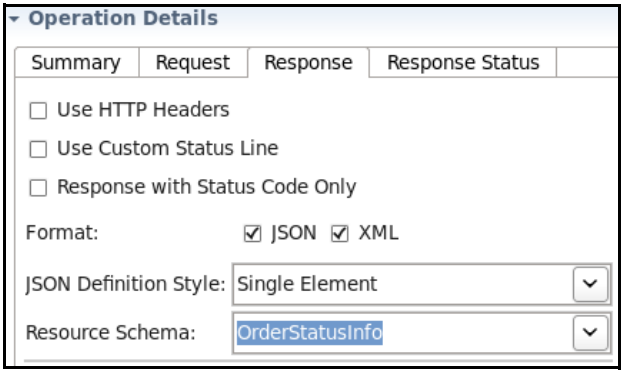
9. Update the HTTP connection setting using previously defined HTTP resource.

- Click **RestService** and then view *Properties*
- Single *Binding* tab is shown. Update it as follows:
  - HTTP Connector Name (browse): **acmeshared.HTTPLookup**
  - Reply Client Format: **JSON** and **XML** checked

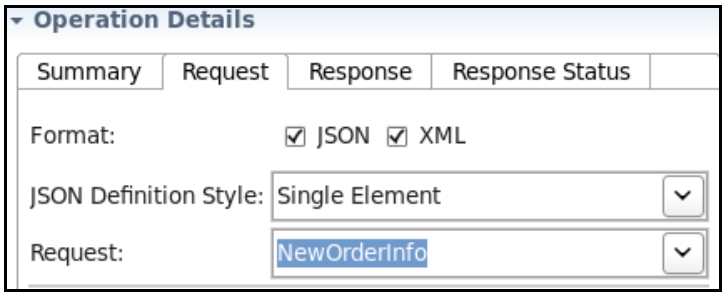


10. Set Response Type of XM for both operations

- Under Operations select **get**
  - Under Operation Details select *Response* tab
  - Format: **JSON** and **XML** checked



- Under Operations select **post**
- Under Operation Details select *Request* tab
  - Format: **JSON** and **XML** checked



- **Delete** HTTP connector created in *ACMEAppModule* > *Resources* > ***acmeappmodule*** > ItemOrder.httpConnResource
- **Save** your changes
- Keep project open for the next exercise

Lab B - Exercise 2: Using Swagger UI

Overview

In this exercise you will use Business Studio to test your RESTFul service. A Swagger interface is provided for testing POST and GET operations. In addition you will also use the browser to test your RESTful service.

Steps

1. Run your solution and verify the BW application is started successfully.

- From the menu choose *Run* > ***Run Configurations..***

  Seleccionar: acmeappmodule.OrderLookup, Timer.app y services.Ordering

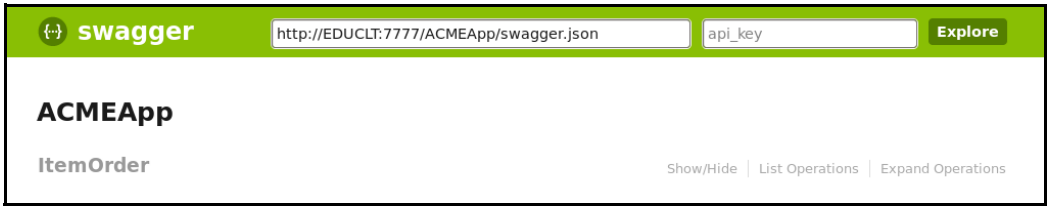
  - Select ACMEApp and Timer applications and click **Run**
  - Verify that the application is started
  - The timer fires and the web service runs and the output is displayed

2. Launch the swagger interface using a browser.

- Launch a browser using the Mozilla Firefox link on the toolbar
- Type in the following URL to access the Swagger Interface

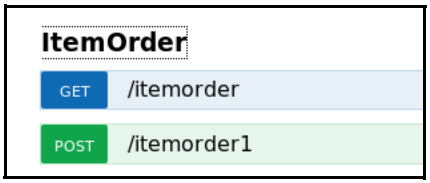
  `http://localhost:7777/ACMEApp`

  - The Swagger Interface as shown below, should appear



3. Use Swagger interface to invoke GET operation.

- Click on the **ItemOrder** (it is a hyperlink)
  - It shows two operations, POST and GET



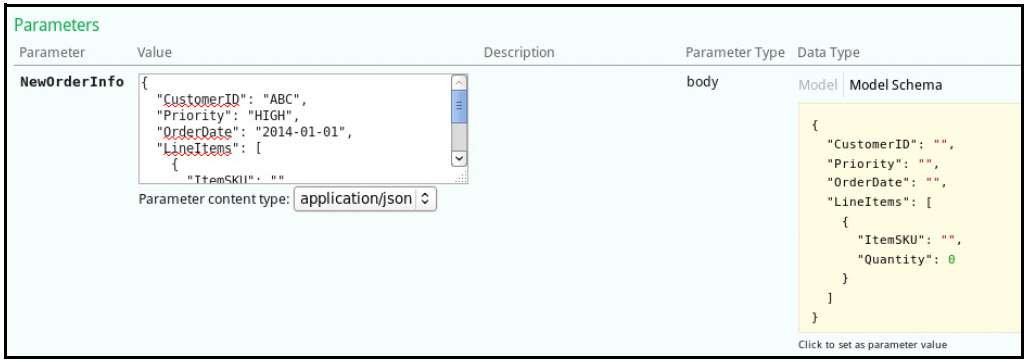
- Click on **GET**
  - Details are displayed, including Response Class
  - Click **Try It Out!** button

Note: You may get an exception in the console window, since the same HTTP port is being used for SOAP and RESTful service. You can safely ignore this error.

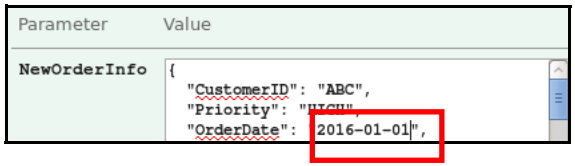
- Switch the Response Content Type (setting above Try It Out! button)
  - Click **Try It Out!** and verify that the correct content is returned

4. Test the POST operation using the Swagger UI.

- Click **GET** button to hide its details and then click **POST** button
  - Notice the *Parameters* section, where you can provide input (as JSON or XML)
  - On the right side, notice Model and Model Schema links
  - Select **Model Schema** to view sample input schema
  - Click on the schema, and it is copied to the input section



- Now provide values for **CustomerID**, **Priority** and **OrderDate** as shown



  - Click **Try it Out!**
  - No response is returned (Response code is 0 and response body is empty)
- Open the Business Studio *Console* view and the error details are displayed
  - It indicates that you should use **dateTime** format for OrderDate

5. Provide a valid date and test again.

- Provide a valid date: **`"2016-01-01T12:00:00"`**
  - **Try It Out!** again and verify response is returned

Request URL

```
http://EDUCLT:9797/itemorder
```

Response Body


```
{
  "OrderID": "NOT AVAILABLE",
  "CustomerID": "NA",
  "Status": "NA",
  "OrderAmount": 0,
  "HandlingAmount": 0,
  "TotalAmount": 0,
  "DateChanged": "2016-12-15T16:35:31.755-08:00"
}
```


6. Verify GET operation using browser as a REST client.

- In a new browser tab enter this URL:
  `http://educlt:9797/itemorder`
  - You can also copy this URL from an earlier step (see Request URL above)
- Verify that results are returned are in XML format as shown:


```
- <ns:OrderStatusInfo>
    <ns:OrderID>TESTORDER123</ns:OrderID>
    <ns:CustomerID>N/A</ns:CustomerID>
    <ns:Status>Work in Progress</ns:Status>
    <ns:OrderAmount>0.0E0</ns:OrderAmount>
    <ns:HandlingAmount>0.0E0</ns:HandlingAmount>
    <ns:TotalAmount>0.0E0</ns:TotalAmount>
    <ns:DateChanged>2016-12-15T17:42:54.218-08:00</ns:DateChanged>
  </ns:OrderStatusInfo>
```


- Enter an invalid URL and you get a 404 error: (e.g. **`http://educlt:9797/itemorder2`**)
- **Close** browser windows
- **Stop** the BusinessWorks engine

Lab B - Exercise 3: Using Parameters (Optional)

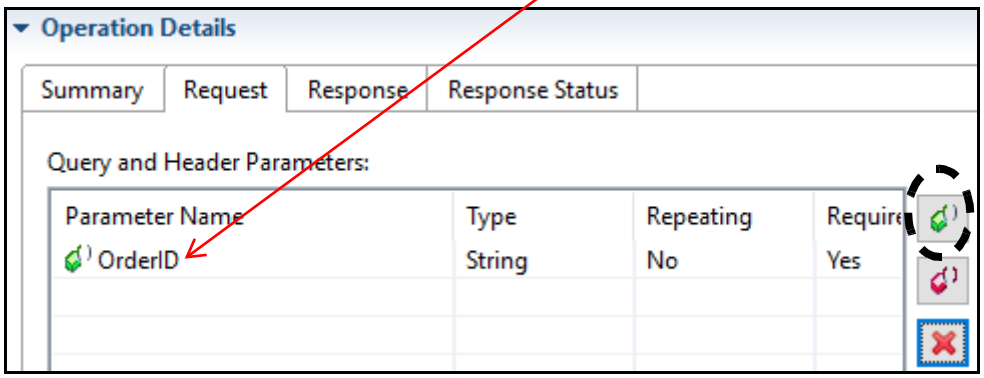
Overview

In this exercise you will update the GET operation to return values for a specific OrderID entered by the user.

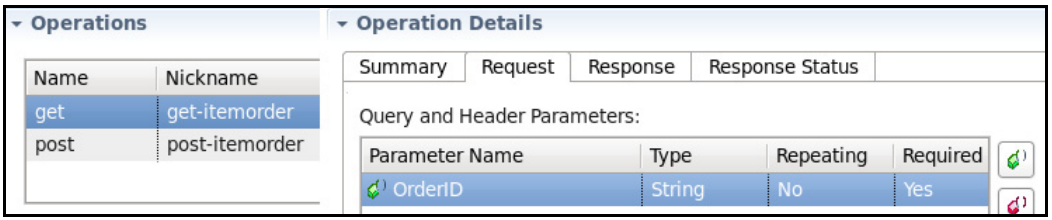
Steps

1. Add a parameter for the GET operation

- In *Project Explorer*, locate *ACMEAppModule> Module Descriptors* > ***Components***
- Double-click on **Components** and expand services.*Ordering* > *Itemorder* > ***RestService***
- In **Properties > Binding > Operations**, confirm that **get** is selected (if not, select it)
  - Now in **Operation Details**, click on **Request** tab
  - Click on **Add Query Parameter** button (circled)
    - A new parameter named **query1** is added, rename it to OrderID
    - Click on **No** under **Required** to change it to **Yes**



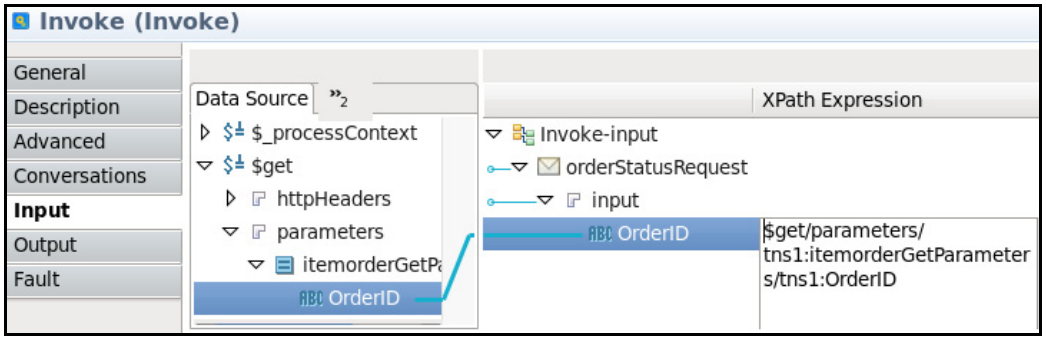
- The final result of adding the parameter is as shown below:



- **Save** your changes
- Close the **Module Properties** view

2. Update the GET operation to use this newly defined parameter

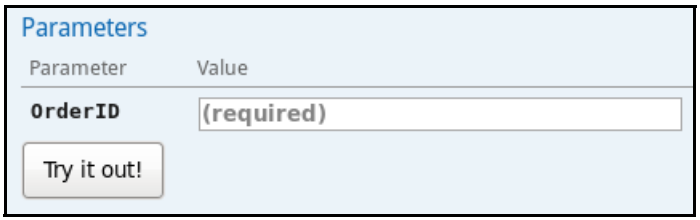
- Open the **Ordering.bwp** (if not already open)
- Locate the **Invoke** activity and select it
- Now in **Properties > Input,** map **OrderID** as shown
  - Replace the existing value (TESTORDER123)
    (Drag OrderID from the left to the *XPath Expression value for OrderID*)



- **Save** changes

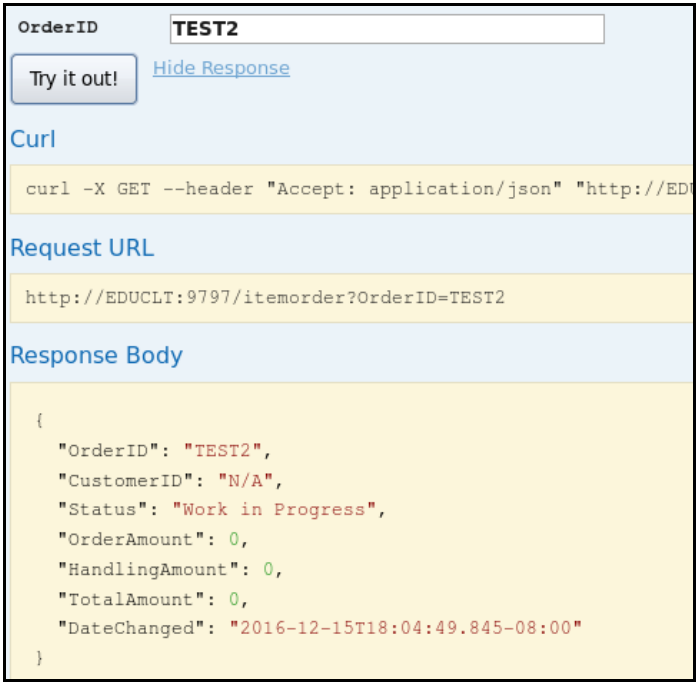
3. Run your solution and in Swagger UI verify a new parameter is visible.

- Right-click on **Ordering** process and choose ***Run BusinessWorks Applications***
  - Verify that the application is started
- In a browser, type in the following URL
  `http://localhost:7777/ACMEApp`
- Click on **<u>ItemOrder</u>**
  - Click on **GET** and you should see OrderID in the parameters section



4. Test using valid and invalid order ids.

- Enter a valid OrderID: **`TEST2`** (or `TEST1` or `TEST3`)
  - Click **Try it Out!**



  - Verify a valid response is returned as shown

5. Test using a browser as a REST client.

- In a new browser tab, enter this URL

  `http://educlt:9797/itemorder?OrderID=TEST2`

  - Verify results are returned


```
- <ns:OrderStatusInfo>
    <ns:OrderID>TEST2</ns:OrderID>
    <ns:CustomerID>N/A</ns:CustomerID>
    <ns:Status>Work in Progress</ns:Status>
    <ns:OrderAmount>0.0E0</ns:OrderAmount>
    <ns:HandlingAmount>0.0E0</ns:HandlingAmount>
    <ns:TotalAmount>0.0E0</ns:TotalAmount>
    <ns:DateChanged>2016-12-15T18:30:43.754-08:00</ns:DateChanged>
  </ns:OrderStatusInfo>
```


- Enter the following URL to see the REST service specification

  `http://educlt:9797/itemorder`

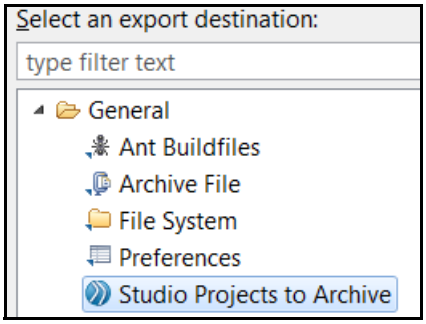

```
{"cause":null,"stackTrace":[{"methodName":"getParametersNode","fileName":"HTTPE
{"methodName":"apply","fileName":"GetMethodInflector.java","lineNumber":299,"cl
{"methodName":"apply","fileName":"GetMethodInflector.java","lineNumber":1,"clas
l,"className":"sun.reflect.GeneratedMethodAccessor21","nativeMethod":false},
{"methodName":"invoke","fileName":"DelegatingMethodAccessorImpl.java","lineNumb
{"methodName":"invoke","fileName":"Method.java","lineNumber":498,"className":"-
{"methodName":"invoke","fileName":"ResourceMethodInvocationHandlerFactory.java"
{"methodName":"run","fileName":"AbstractJavaResourceMethodDispatcher.java","lin
{"methodName":"invoke","fileName":"AbstractJavaResourceMethodDispatcher.java","
{"methodName":"doDispatch","fileName":"JavaResourceMethodDispatcherProvider.jav
```


- Enter an invalid URL and notice you get a 404 error

  `http://educlt:9797/itemord`

6. Export your solution as a ZIP file, for future reference.

- **Stop** your application
- In Business Studio menu , choose *File* > ***Export***
  - In the next window, choose *General* > ***Studio Projects to Archive***



  - Click **Next**
  - In the next window:
    - Select **ACMEApp**, **ACMEAppModule** and **ACMEShared**
    - To Archive File:
      `/home/tibco/bw6edu/dev/exports/`**`RESTful_service`**
    - Save in **ZIP** format
  - Click **Finish**

Superb! A RESTFul Service has been built and tested. Time for a short break!

Notes

Notes

Unit 3 Debugging and OSGi

This unit describes the BusinessWorks 6 development methodology.

Topics

- *Objectives,* page 74
- *The BusinessWorks Debugger,* page 75
- *Setting and Using Breakpoints,* page 76
- *Viewing Job Data,* page 77
- *BusinessWorks Debug Environment,* page 78
- *Setting up Debugging Details,* page 79
- *BusinessWorks and OSGi,* page 80
- *BusinessWorks Application Startup,* page 81
- *Accessing the OSGi Console,* page 82
- *Key OSGi Commands,* page 83
- *Viewing Bundle and Engine Details,* page 84
- *Endpoint Details,* page 85
- *Review,* page 86
- *Lab Overview,* page 87

Objectives

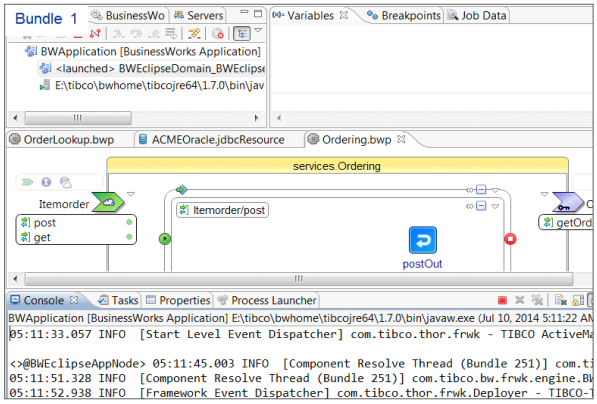
- Describe the application debugging environment
- Identify capabilities of debugger
- Explain OSGi functionality in debugger
- Explore dependency check and startup procedure
- Examine OSGi console and commands

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives

The BusinessWorks Debugger

- Opens debug perspective with these views
  - Job management
  - Process flow
    - Breakpoints
  - Console logs
  - Variables
  - Breakpoints
  - Job Data



- Process launcher: Invoke sub-process with data

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

The BusinessWorks Debugger

The debugger perspective for BusinessWorks is similar to the Java debug perspective provided by an Eclipse environment.

This perspective provides options for simple debugging, or complex analysis of your runtime environment depending on your requirements.

There are distinct views that provide details about:

1. JVM and step by step debugging

2. Job data and breakpoints

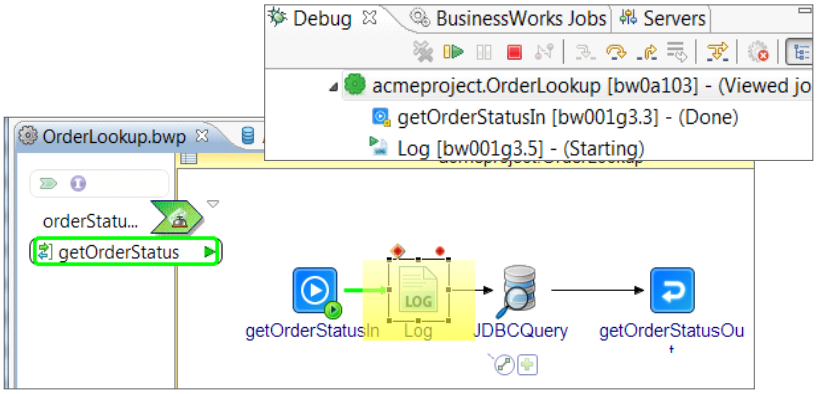
3. Process flow details

4. Console view

After a process has been completed, you can step through the process execution, and examine data for each activity in the process. You can also get visibility into the actual process flow during execution, and step into a sub-process if needed. Set multiple breakpoints, and terminate the job instance if needed.

Setting and Using Breakpoints

- Set breakpoints while engine is running
  - No restart needed
- Options: Step forward, step into and continue



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

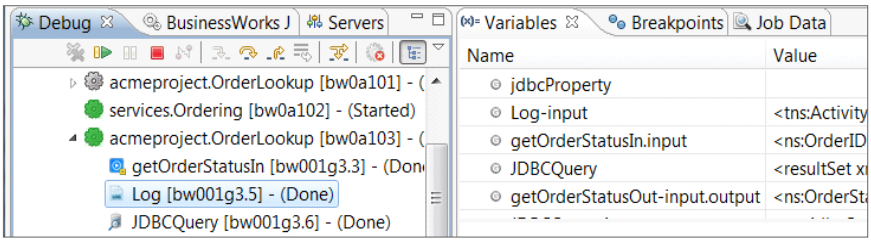
Setting and Using Breakpoints

To set a breakpoint, simply select the activity in the process, after this, right-click and set a before or after break point. You do not need to restart the engine, the breakpoint is applied when the next job is executed by the engine.

When the engine stops at a break point, you can step into or step over the break point using the menu options that are highlighted. Optionally choose to continue the process on to the next breakpoint. As you continue into the process execution, the activities in the process are highlighted, to show the execution path.

Viewing Job Data

- Select job and view job data
  - Input / Output / Error tab
    - Depends on activity
- Currently running job: view variables



TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. • © TIBCO Software Inc.

Viewing Job Data

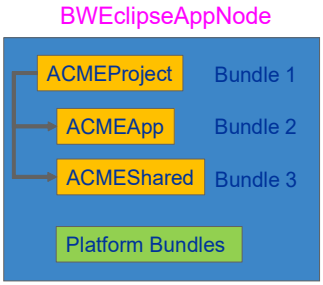
If you click on a specific activity in a process execution, you can view the data associated with this activity. Some activities produce output only (such as starter activities) and some accept an input and produce an output. Depending on the activity selected, you will see multiple tabs in the Job Data section. There will be Input and/or Output tabs and an All tab, to display all job data (up to this point).

Variables: shows the variables associated with the process being debugged including Type Names and Logical Structure.

Job Data shows available information about the running process instances.

BusinessWorks Debug Environment

- Single AppNode (JVM) started
  - Name: *BWEclipseAppNode*
    - *Full details discussed later*
- AppNode functions
  - Loads all modules in workspace
  - As OSGi bundles
  - Loads platform bundles



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

BusinessWorks Debug Environment

Here is a snapshot of the behind-the-scenes activities when you are running a debugger.

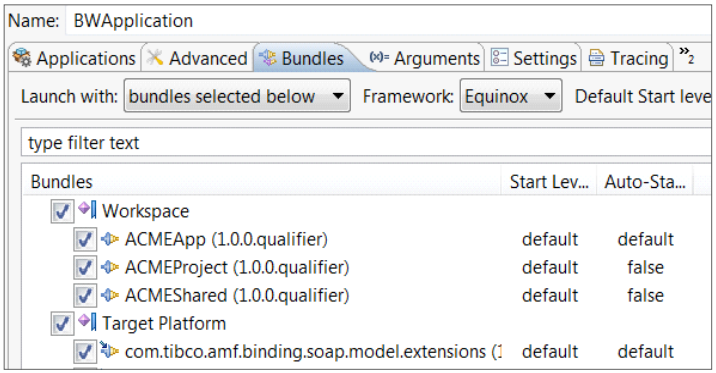
The application contains both application modules, and it is deployed in a single JVM, known as an AppNode. The name of the AppNode is generated as BWEclipseAppNode, and the JVM runs separately, outside Business Studio (JVM).

The AppNode loads your application modules as OSGi bundles into the JVM. In addition, it loads the BusinessWorks platform bundles which help execute your OSGi bundles.

AppNode and the execution environment are examined in more detail in the deployment unit.

Setting up Debugging Details

- View debug configuration
  - Select applications in workspace
  - Do NOT change target platform bundles!



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

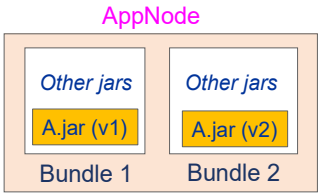
Setting up Debugging Details

The debug configuration details shows all of the application bundles (and shared bundles) in the workspace that can be selected for debugging. By default, all bundles are selected. You can uncheck the bundles that you do not want to use.

The platform bundles are also listed and shown. Do NOT modify the platform bundles (select or deselect) unless asked to do this by TIBCO support.

BusinessWorks and OSGi

- Runtime environment is OSGi-based
- OSGi Features
  - Run multiple versions in same JVM
  - Dependency validation on startup
- OSGi benefits
  - Fail fast
    - *Class not found* errors eliminated
  - Promotes modular development



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

BusinessWorks and OSGi

BusinessWorks runtime environment is OSGi based. Applications run as OSGi bundles. Benefits:

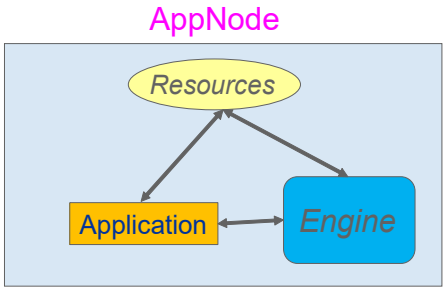
- No classNotFoundException issues. If the required/imported packages are not available then the bundle will not resolve, thus the application will not start. This is a *fail fast mechanism.*
- Promotes modular development. Developers can expose only those parts of the application that may be shared with other developers.
- Run multiple versions of a single jar file in the same JVM without any errors.

When an application starts up, it sends asynchronous messages to all the OSGi bundles it depends on, such as resources bundle and palettes bundle. If one of the OSGi bundle fails to respond, the application displays an impaired state. The JVM (Appnode) continues running when an application is in impaired. Eventually, when all OSGi bundles respond, the application switches to a started state.

It is quite common to see applications in impaired state as they are starting up.

BusinessWorks Application Startup

- Dependency validation
  - Application / engine
  - Engine / resources
  - Application / resources
- In case of failure
  - Application impaired
  - Engine continues running
- Application may be *impaired* on startup




```
FRWK-300019: BW Application [ACMEApp:1.0] is impaired.
All Application dependencies are resolved for Application
```

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

BusinessWorks Application Startup

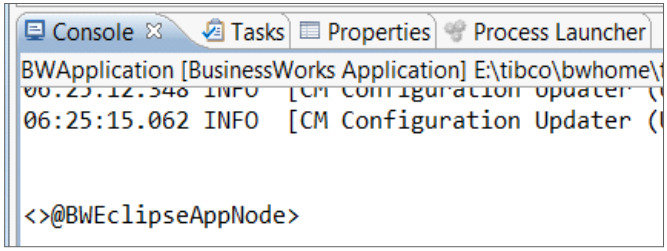
This is the typical lifecycle of an application startup.

The BusinessWorks engine itself is an OSGi bundle. An application verifies that the engine is available as it starts up. The engine then verifies that the application and all its dependencies (wiring) are available. All shared resources are evaluated and started up next.

Once the application dependencies are resolved and the shared resources are started, the application itself can be started. In the debugger you will see the application starting up with *impaired* state, and after all dependencies are resolved, the application moves to a *started* state.

Accessing the OSGi Console

- OSGi console accessed in console window
- Using the console:
  - Fix dependency issues
  - Bundle details
  - Process configuration
  - Resource details



TIBCO TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Accessing the OSGi Console

If an application does not startup, the BusinessWorks engine does not terminate. The other OSGi bundles continue to run as well. This is very beneficial, since it allows you to now use OSGi commands to find out why your application did not start up.

Even if the application does startup, the OSGi console can still be used to find out more information about the runtime environment in Business Studio debugger.

Simply hit the `<Enter>` key in the console window to access the OSGi console. The prompt for the OSGi console is usually <>@BWEclipseAppNode.

Key OSGi Commands

- General
  - help: list of commands
  - help <command>

```
<>@BWEclipseAppNode> help la

la - Print information about all
   scope: frwk
```

- Applications & resources
  - la : List Applications
  - lr: List resources
  - dsr: diagnose resources
    - –Submenu
  - Wires: Dependencies

```
<>@BWEclipseAppNode> dsr
Found following resources

ID   Name
--   ----
1    acmeshared.ACMEOracle
2    acmeshared.HTTPLookup
```

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Key OSGi Commands

Here are some useful OSGi commands to find out more about your deployed/failed application.

Use help to list the available command (100+ commands).

Also use help <command> to find out more about a specific command.

When an application is in impaired state, the `la` command lists details for all deployed applications.

The resources commands are useful to find out more information about the state of all shared resources in your applications. `lr` lists all resources. The dsr command is a diagnostic command which list all shared resources and also it allows you to drill down to get more details about a specific resource using a sub-menu.

The wires command shows all dependencies for an application.

Viewing Bundle and Engine Details

- Bundle details
  - ids: Application bundle ids
  - bundle <id>: details
- Engine details
  - le: Engine summary
  - lec: Engine Details

```
<>@BWEclipseAppNode> wires
ACMEProject[13]: shared/private =
   Wires...
      ACMEShared
   No dependents.

ACMEShared[268]: shared/private =
   No wires.
   Dependents...
      ACMEProject
```

```
<>@BWEclipseAppNode> lec
BW Process Engine Configurations:
name                : Main
debugPort           : 9090
service registration: {com.tibco.bw.
  bw.domain=BWEclipseDomain
  bw.engine.threadCount=8
  name=Main
  bw.engine.name=Main
  bw.engine.persistenceMode=memory
```

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Viewing Bundle and Engine Details

Examine the bundle details using the bundle specific commands.

The `ids` command shows ids for all deployed applications.

The `bundle <id>` command shows details for a specific application bundle.

You can examine engine details using the `le` and `lec` commands. Note that the thread count indicates the details for all applications in the engine (for the whole JVM).

Endpoint Details

- Endpoint commands
  - lendpoints: list endpoints
  - lrestdoc: Swagger access
  - lhttpep: HTTP endpoints

```
<>@BWEclipseAppNode> lendpoints
--------------------------------
[Application Name] : ACMEApp
[Endpoint Type] : REST
[Endpoint URL] : http://localhost:9797
[CLIENT FORMAT ] : xml,json
[RESOURCE PATH ] : /itemorder/{OrderID}
[HTTP METHODS] : POST,GET
------
[Endpoint Type] : SOAP
[Endpoint URL] : /SOAPServiceBinding/orderStatusPort/
------
```

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Endpoint Details

The OSGi console provides three key commands to examine the details for your REST and SOAP based services.

`lendpoints`: The lendpoints command lists all SOAP and REST endpoint details, including host name and port numbers used by the services.

`lrestdoc`: This command provides URL to access the Swagger UI.

`lhttpep`: Lists all HTTP based endpoints that are provided by the this instance.

Review

- Debugger perspective provides multiple views
- Set breakpoints without restarting debugger
- Application and dependencies are OSGi bundles
- Fail fast logic used at startup
- OSGi console provides bundle, resource details

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Run Application in debugger
  - Use breakpoints
  - Examine job data
- Use OSGi console
  - View engine details
  - Explore applications and resource details

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab C Debugging Applications

Overview

In this lab you examine the capabilities of the debugging environment in TIBCO Business Studio. You will use the OSGi console to explore details for the deployed applications.

Objectives

- *Debug a Solution, page 90*
- *Using OSGi Console,* page 96

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3

Directions

- Complete the exercises that follow

Lab C - Exercise 1: Debug a Solution

Overview

In this exercise you will launch the debugger and examine its capabilities.

Steps

1. Launch TIBCO Business Studio and open your process definition.

- Launch Business Studio use Workspace: `/home/tibco/bw6edu/workspace`
  - Browse to *ACMEAppModule* > *Processes* > *acmeappmodule* > ***OrderLookup.bwp***
  - Open the process definition (double-click)

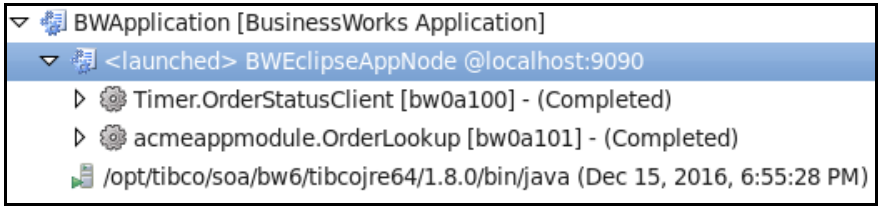
2. Launch the debugger to test your process.

Seleccionar acmeappmodule.OrderLookup y Timer.app

- Right-click on the process diagram and choose ***Debug BusinessWorks Applications*** (this launches the debugger)
  - When asked to switch to the debugger perspective, choose **Yes**

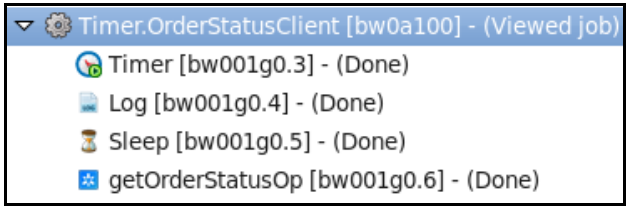
3. Examine the debug view to see process execution details.

- Locate the *Debug* view (top left corner)
  - Expand: *<launched> BWEclipseAppNode@localhost:9090*
  - Confirm that the **Timer Process** and **OrderLookup** processes are **completed**



4. View details for a process.

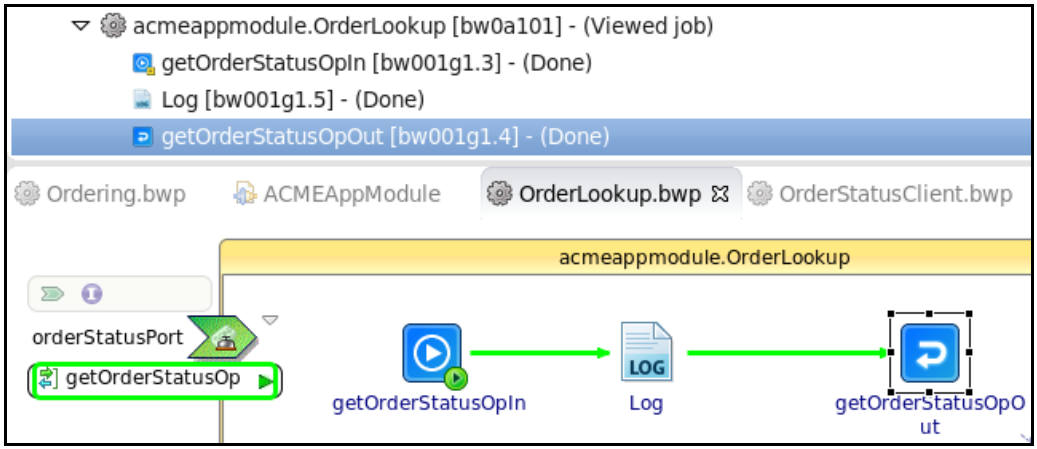
- Expand the process to view its details



  - Notice that the **Timer** and **getOrderStatusOp** were executed
  - Observe the **JobIDs** for *Timer* and *OrderLookup*
    - Observe in the graphic above the Timer JobID of **bw0a100** (your JobIDs may be different)

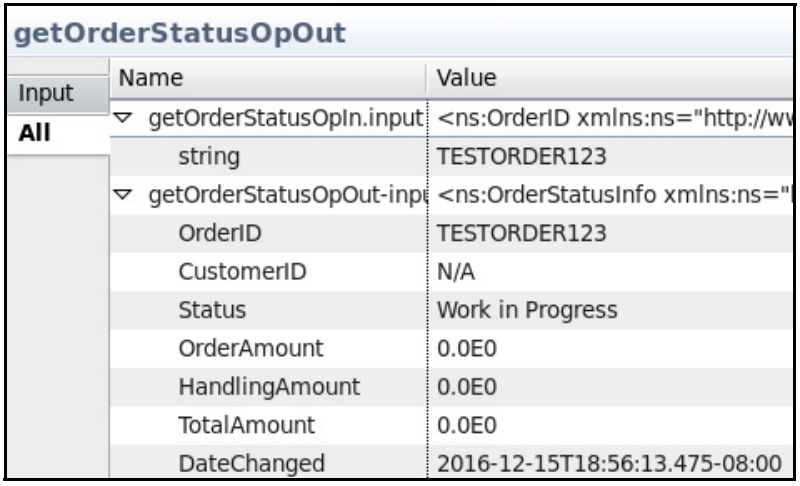
5. Examine individual activity data.

- Expand the *OrderLookup* process
  - Click on the **Log** activity
  - Notice that this activity is highlighted in the process diagram below
  - Click on other activities and verify that the process highlights the relevant activity
  - When you click on **OrderLookup**, activities are shown connected with a green arrow

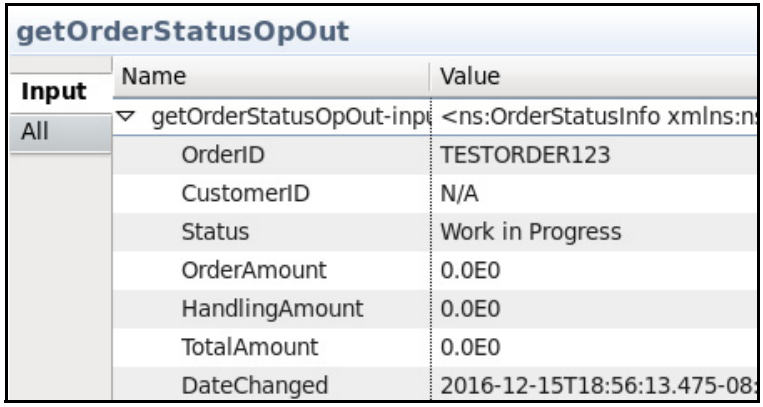


6. View input and output data for activities.

- Locate the *Job Data* view (top right hand section), and click on it
  - No job data may be displayed at the moment
  - Now click on any activity (getOrderStatusOpOut) in *Debug* view
  - Job input data is displayed by default
  - Click the *All* sub-tab to view input and output data

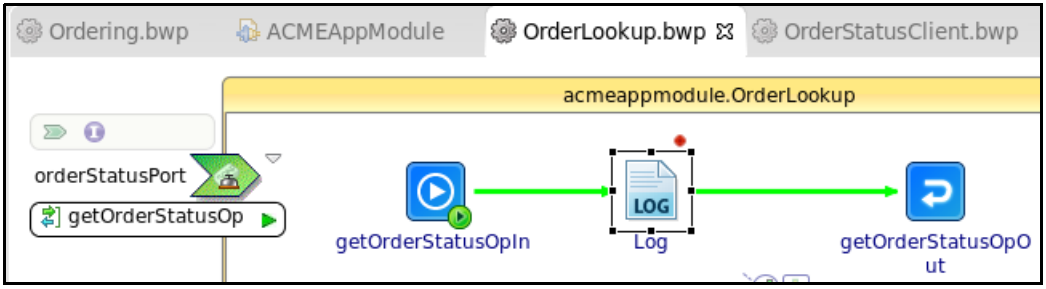


    - *Input* tab indicates that an input is provided to the activity
  - Click *Input* to see input data for this activity



7. Set a breakpoint for an activity.

- In the process view, center window, locate the **Log** activity for *OrderLookup* process
  - Right-click and choose *Breakpoint* > ***Set After***
  - Notice that a red dot appears on top of the activity

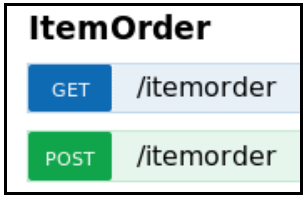


8. Open a browser page to submit a GET request.

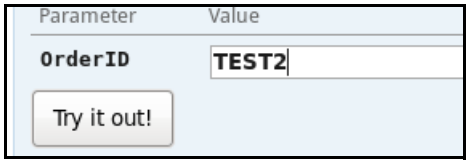
Parar el debugger y seleccionar únicamente: el proceso Ordering

- Open a browser (Firefox) and type in this URL

  `http://localhost:7777/ACMEApp`
  - Click **<u>ItemOrder</u>**, then the **GET** button

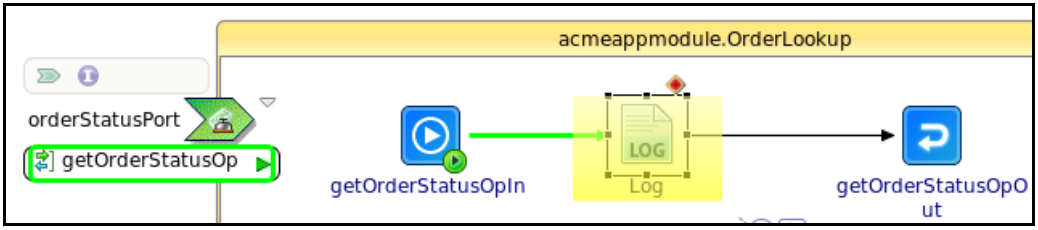


  - Enter OrderID : **`TEST2`**
  - Click **Try it Out!**



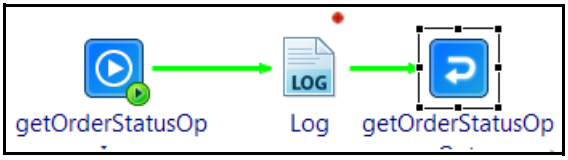
9. Step through the process and observe the data.

- In the Debugger, notice that the Log activity is highlighted



  - In the *Debug* view, the options for continuing are also highlighted

  - **Step Over** this activity by clicking on the circled button, or use the `<F6>` key
  - Step Over the next activity, Step over to execute it
  - Verify that the job data view is refreshed as you move to different activities

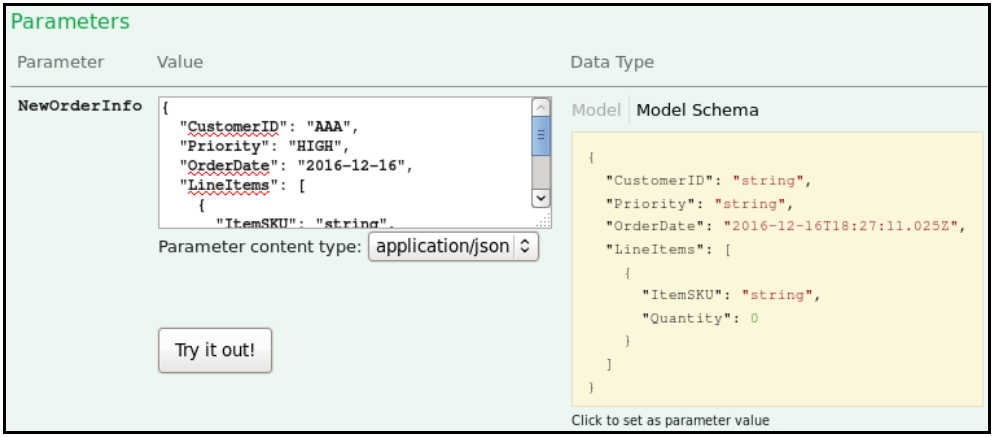


- Keep stepping over or click **resume** to until you complete the process
- Right-click on the **Log** activity and choose *Breakpoint* > ***Clear All in Process***

10. Confirm that some requests do NOT result in a job start.

- In the *Debug* view, clear all completed jobs, by clicking on the icon shown

- Now go to the browser window running the Swagger UI:
  - Click on the **POST** operation
- In *Parameters*, click on Model Schema
  - Click on the schema to load the schema definition and complete it as shown:
  - **Note:** We are providing an invalid value for Date Time



- Click **Try it out!**
- Observe that the browser window returns an empty reply (no content)
- In the *Business Studio* window, observe that **no Jobs are started**
- However, the *Console* window returns an error about date time

```
The initial value '2016-12-16' is not valid with respect

vc-datatype-valid.?: The literal '2016-12-16' is not datat
castAsGregorian(GregorianType.java:258) ~[na:na]
```

- **Stop** the debugger when done exploring
- Right-click in *Console* window and choose **Clear** to remove old messages

Lab C - Exercise 2: Using OSGi Console

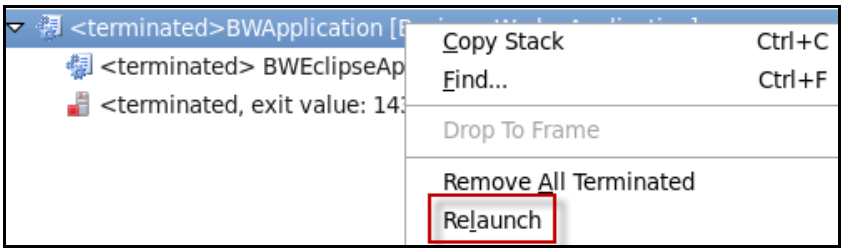
Overview

In this exercise you will use the OSGi console provided with the debugger and examine details about bundles and shared resources.

Steps

1. Start the BusinessWorks debugger.

- In the Debug perspective, right-click on *BWApplication* and choose ***Relaunch***



  - Verify that the application is started and the Received request message is seen

```
Started BW Application [ACMEApp:1.0]
021: All Application dependencies are resolved for Application
Started BW Application [Timer.app:1.0]
g - ******** Started Timer Application at: 11:01:57 AM********
Log - Received Lookup request for OrderID: TESTORDER123
```

2. Access the OSGi console in the console view.

- In the *Console* window, hit `<Enter>` and the OSGi prompt is displayed as shown:

```
BWApplication [BusinessWorks Application] /opt/tibco/bw6/tibcojre64/1.7
12:13:18.154 INFO  [bwThread:In-Memory STWorkProcessor-2]
12:13:41.200 INFO  [bwThread:In-Memory STWorkProcessor-6]

<>@BWEclipseAppNode>

<>@BWEclipseAppNode>
```

- Type in this command to see all available commands

  `help`

  - Type *help* <*command*> to get help on a specific command and its *scope*

    `help les`

3. List all running BusinessWorks Process and applications.

- The following command shows all processes:

  `lp`

```
BW Processes:
  BWProcess [name=services.Ordering,module=ACMEAppModule,state=Reso
  BWProcess [name=acmeappmodule.OrderLookup,module=ACMEAppModule,st
  BWProcess [name=Timer.OrderStatusClient,module=TimerModule,state=
```

  - List all BusinessWorks applications (scroll to see details for both applications)

    `la`

```
<>@BWEclipseAppNode> la
Applications:
-------------------------------------------------------
Name        : Timer.app:1.0
Bundle      : Timer.app_1.0.0.qualifier [95]
Bundle Id   : 95
State       : Running
                  TIBCO-BW-FRWK-300012: Started BW Appl
ConfigState : InSync
Modules     :
  (BW Application Module) TimerModule 1.0.0.qualifier:
    Bundle Id              : 320
    Bundle State           : ACTIVE
    State                  : Resolved
```

  - To view complete configuration details enter the **`lcfg`** command

4. List all details about shared resources.

- This command lists all shared resources: **`lr`**

```
<>@BWEclipseAppNode> lr
ID   Name
--   ----
1    acmeshared.HTTPLookup
2    timermodule.HttpClientResource
```

- You can see resources for your solution

5. Use **`dsr`** command to view details about a specific resource.

- Use the **`dsr`** command to display all shared resources
- A submenu is presented as shown

```
<>@BWEclipseAppNode> dsr
Found following resources

ID   Name                            Type
--   ----                            ----
1    acmeshared.HTTPLookup           {http://x
2    timermodule.HttpClientResource  {http://x

        Enter:
        ID of the Shared Resource to get more d
        'h', if you dont see the resource you a
        's', if you want to search for a Shared
        Any other key to quit:
```

- Enter **`1`** to see details for your resource
- You can continue to examine the resource further, or press `<Enter>` to exit

6. View details http and REST details.

- The following commands show details about http and REST

  `lhttpep` and **`lrestdoc`**

```
<>@BWEclipseAppNode> lhttpep
 EDUCLT:9797 (acmeshared.HTTPLookup)


<>@BWEclipseAppNode> lrestdoc
[Application Name]: ACMEApp
[Discovery Url]: http://EDUCLT:7777/ACMEApp

<>@BWEclipseAppNode>
```

- Examine more commands and see more details
- When you are done, **Stop** the debugger
- **Switch back** to *Design* perspective
- **Shut down** Business Studio

Brilliant! You can now debug a BusinessWorks solution!

Notes

Unit 4 Configuring JDBC Drivers

This unit examines the JDBC driver option and design time JDBC configuration.

Topics

- *Objectives,* page 102
- *Transitioning JDBC Activity Based Projects,* page 103
- *Packaging an Application,* page 104
- *Connecting to a Database using JDBC,* page 105
- *JDBC Driver: Setting up Design Time,* page 106
- *Design Time JDBC Configuration,* page 107
- *JDBC Driver: Run Time Changes,* page 108
- *Working with a Database,* page 109
- *Adding a JDBC Query,* page 110
- *JDBC Connections in a Process,* page 111
- *Database Schema - Lab D,* page 112
- *Interface Specification for Lab D,* page 113
- *Review,* page 114
- *Lab Overview,* page 115

Objectives

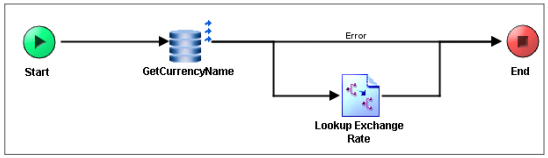
- Explain application deployment using bundles
- Examine JDBC driver options
- Discuss design time JDBC configuration
- Explore database interaction and query builder
- Describe runtime setup for JDBC drivers

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives

Transitioning JDBC Activity Based Projects

- BusinessWorks 6 must be configured twice
  - Design time:
    - Update Studio Path
      - Connect to database to set activity details
  - Run time:
    - Create OSGi Bundle
      - Connect to database, perform action



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Transitioning JDBC Activity Based Projects

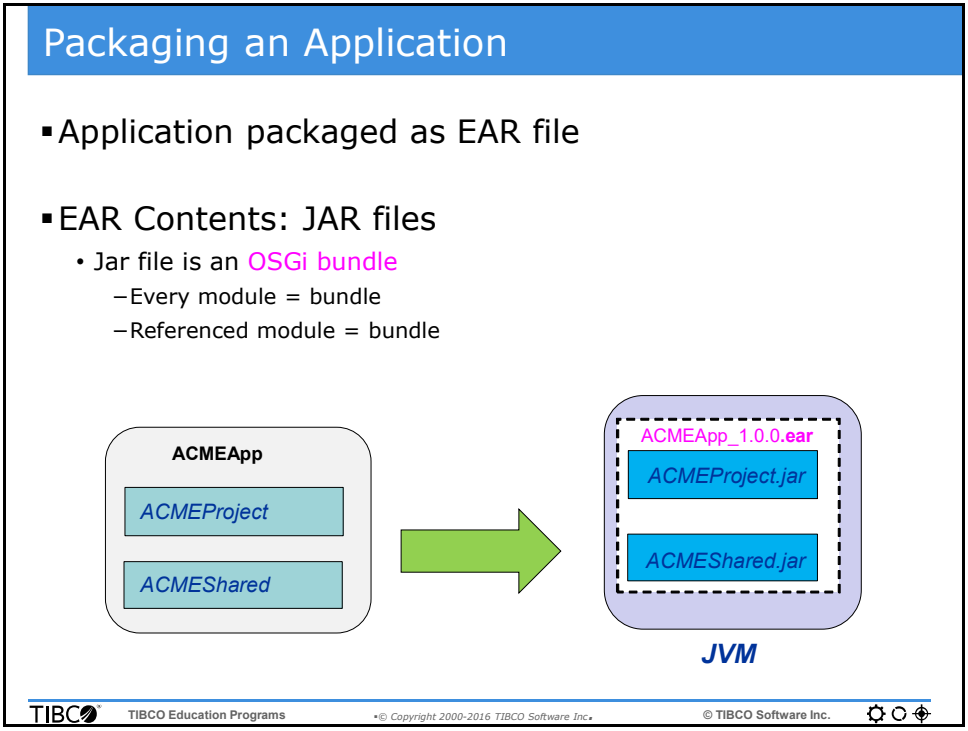
Version 5:

For design time, update the designer.tra to reference the third party JDBC JARs.

For runtime, update the bwengine.tra file with the same settings.

Version 6: When using JDBC activities in version 6, application design time and run time environments have to be updated. To update the design time environment for an application, the folder referenced by Business Studio must be populated with the JDBC driver. For the application runtime environment, an OSGi bundle must be created, and this bundle must package the JDBC driver.

Note: Beginning with version 6, TRA files are no longer updated or used in a similar manner as version 5.

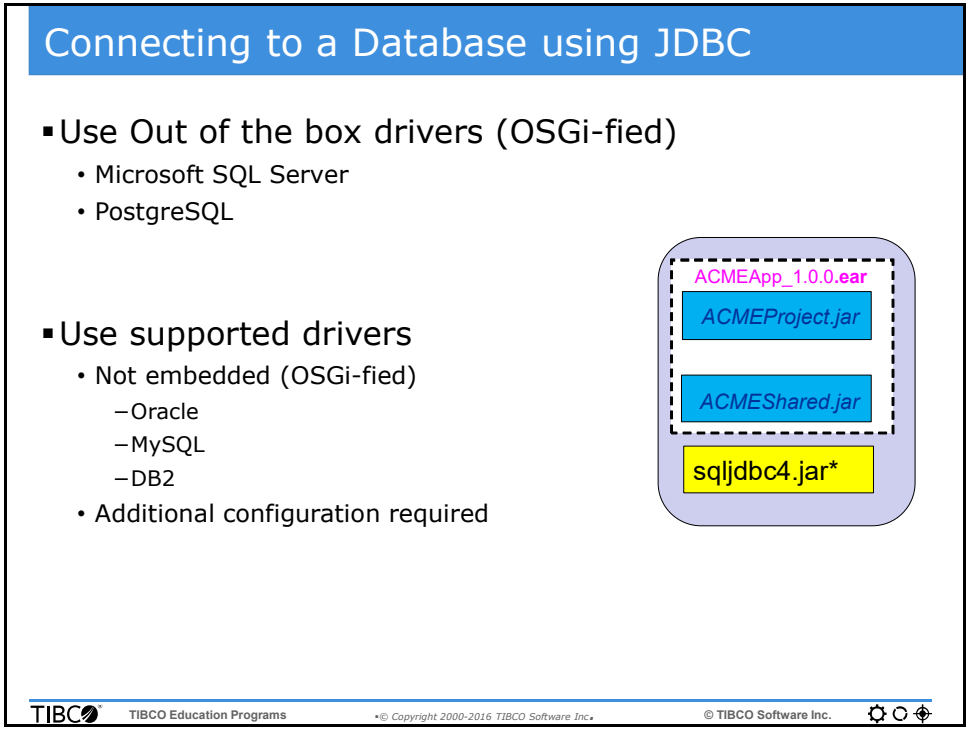


Packaging an Application

To understand how to use third-party JDBC drivers, it is important to first examine the structure of an EAR file.

An EAR file packages all modules into individual JAR files. So an EAR file can contain one application Jar file and zero or more shared module Jar files. At runtime each JAR file is deployed as an OSGi bundle.

The OSGi bundles have dependencies between them, which ensures that the application starts up after all shared modules have started up.



Connecting to a Database using JDBC

To add a third-party JDBC jar file, it must also be packaged as an OSGi bundle.

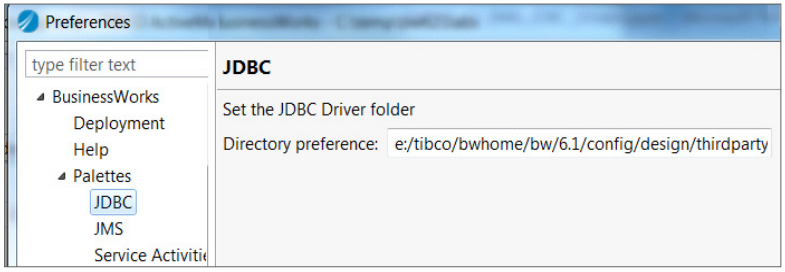
TIBCO Businessworks currently packages drivers for the following databases:

- PostgreSQL
- Microsoft SQL Server

In addition, the JDBC Palette also supports Oracle, MySQL, and DB2 databases.

JDBC Driver: Setting up Design Time

- Design time: Update Business Studio
  - To use a supported driver
    - Locate Jar folder in preferences
    - Copy file to folder
  - Business Studio ***restart not required***

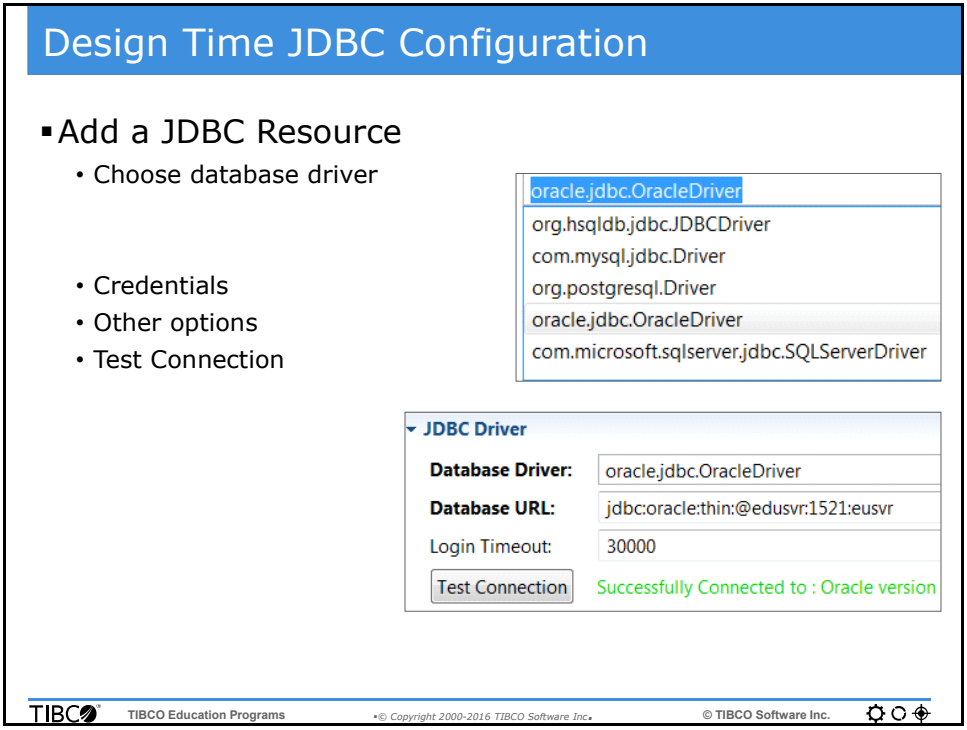


TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

JDBC Driver: Setting up Design Time

To use another location for the drivers, a preference is provided in TIBCO Business Studio which can be updated for the JDBC Palette to use drivers at Design time. To change the Directory Preference, go to Windows > Preferences > BusinessWorks > Palettes > JDBC.

To use other databases with the JDBC Palette at Design time, place the drivers at <BW_Home>/6.x/config/design/thirdparty location.



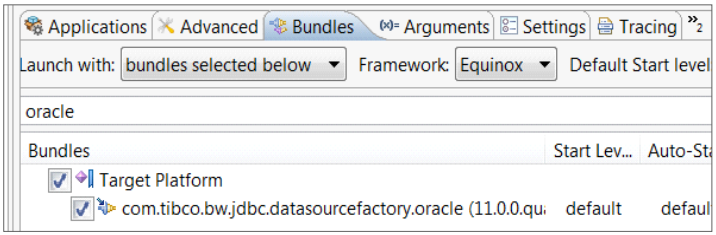
Design Time JDBC Configuration

The design time configuration provides a list of JDBC drivers supported by BusinessWorks. Remember, these JDBC drivers are supported but some of them (Oracle and SQL Server) are not installed with the product. You have to obtain these JDBC drivers and install them separately.

Each database driver has a slightly different URL, so configure based on the driver. Provide the credential and then use Test Connection to confirm that you can connect to the database. The Test Connection button is also used to test whether the JDBC driver is accessible by Business Studio.

JDBC Driver: Run Time Changes

- Run time : create OSGi-fied bundle
  - bwinstall utility
    - Example: bwinstall oracle-driver
  - bwinstall actions:
    - OSGi bundle created
    - Bundle reference added in runtime configuration



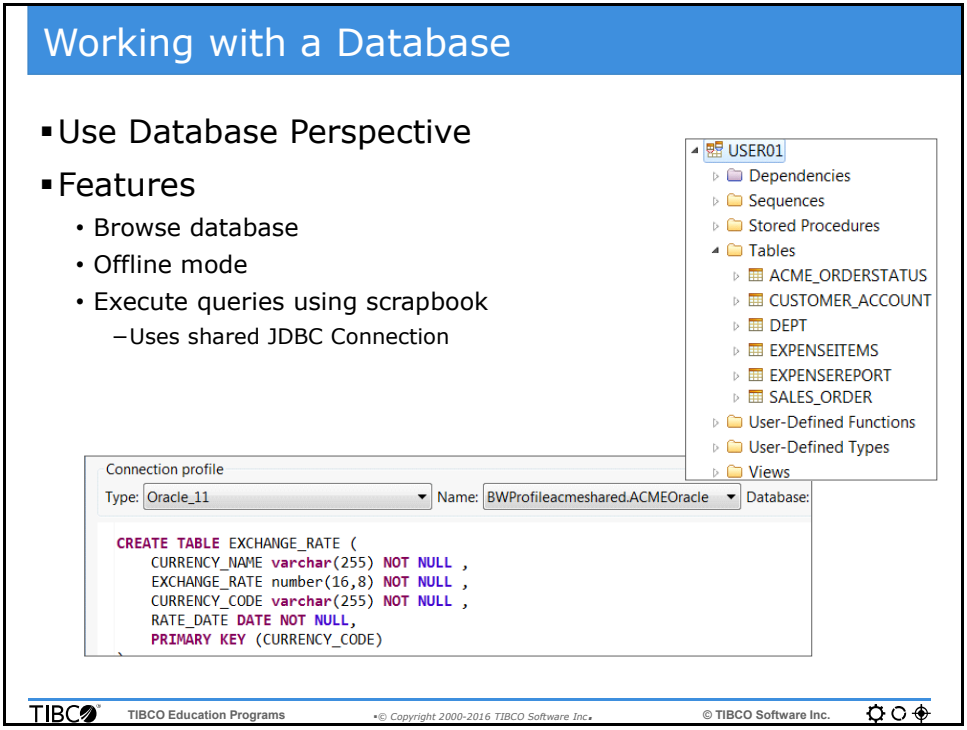
TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

JDBC Driver: Run Time Changes

To add the drivers at runtime for Oracle or MySQL, run the bwinstall command from <BW_Home>/6.x/bin. It is recommended that you run bwinstall with no arguments to access the help and instructions to add the driver to the runtime.

The script creates driver bundles and adds it to that specific Businessworks environment. Any AppNodes that are created and started in that Businessworks environment include the drivers automatically.

Since the drivers are part of the system and not part of each application, all the applications on a particular AppNode will use the same drivers. However, this is not true of AppSpaces. The drivers belong to an AppNode and not an AppSpace.



Working with a Database

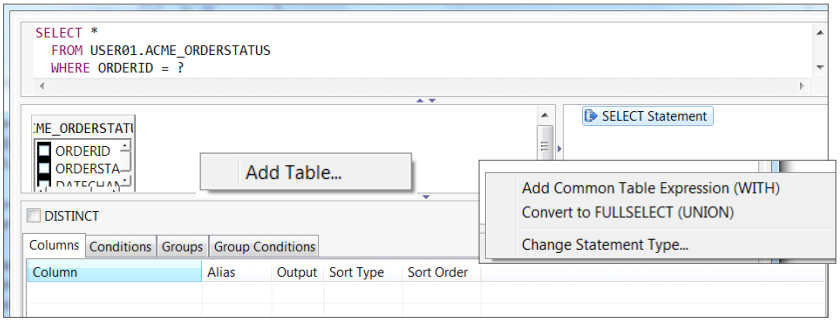
Defining a database connection provides an additional benefit - using the database perspective.

The database perspective is available as part of Eclipse and it is used to explore the database objects and to run SQL queries against the database. It uses the database connection that you defined in a BusinessWorks module.

You can also use the offline mode, which downloads the database metadata (schemas) so you can design queries when the database is not available.

Adding a JDBC Query

- Activity provides query builder
  - Choose statement type, select columns
  - Add conditions and groups
  - Execute and Test



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Adding a JDBC Query

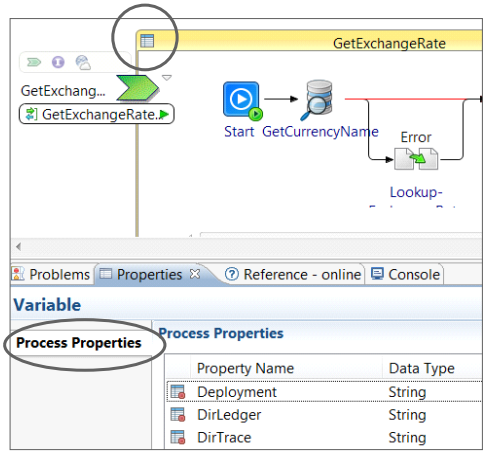
The SQL Button in JDBC query activity launches Eclipse SQL Builder. The SQL Builder is used build queries visually, and it is an excellent way for non-SQL users to define queries without worrying about the specific syntax supported by the database.

Access your database schema and objects so that you can quickly create or edit SQL statements without actually typing any SQL code. You also have the flexibility to add or modify the SQL code in the editor window.

You can test your query in the same window.

JDBC Connections in a Process

- JDBC Shared Resource used in process
  - Creates process properties
- Indication
  - Process boundary



TIBCO Education Programs ▪© Copyright 2000-2016 TIBCO Software Inc.▪ © TIBCO Software Inc.

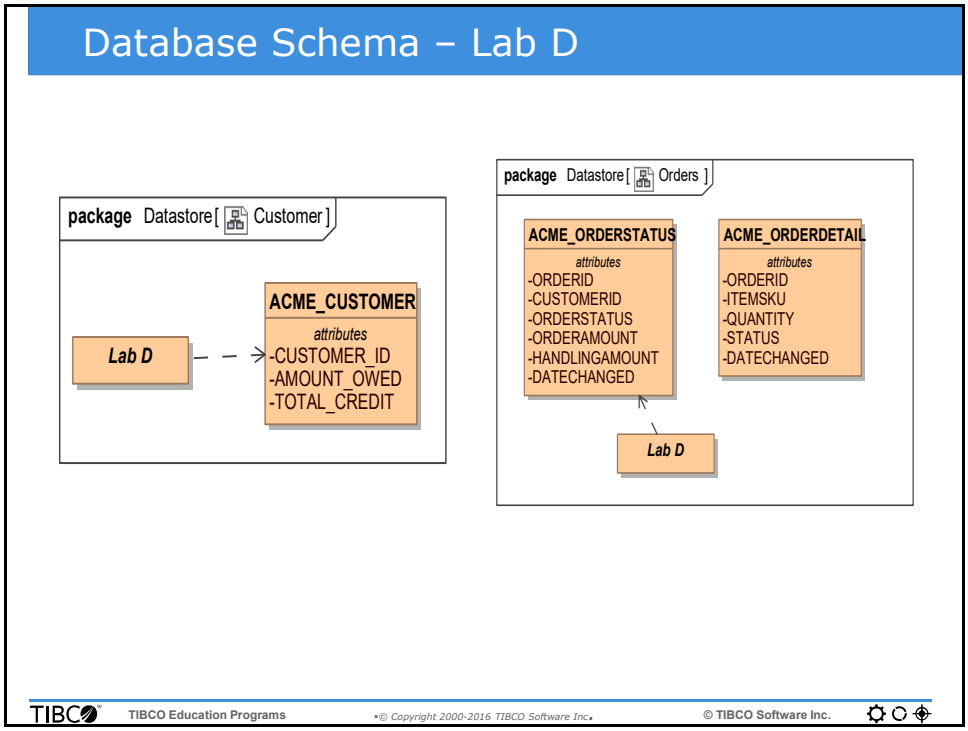
JDBC Connections in a Process

JDBC connections in version 5 are transitioned to shared resources for version 6 Applications.

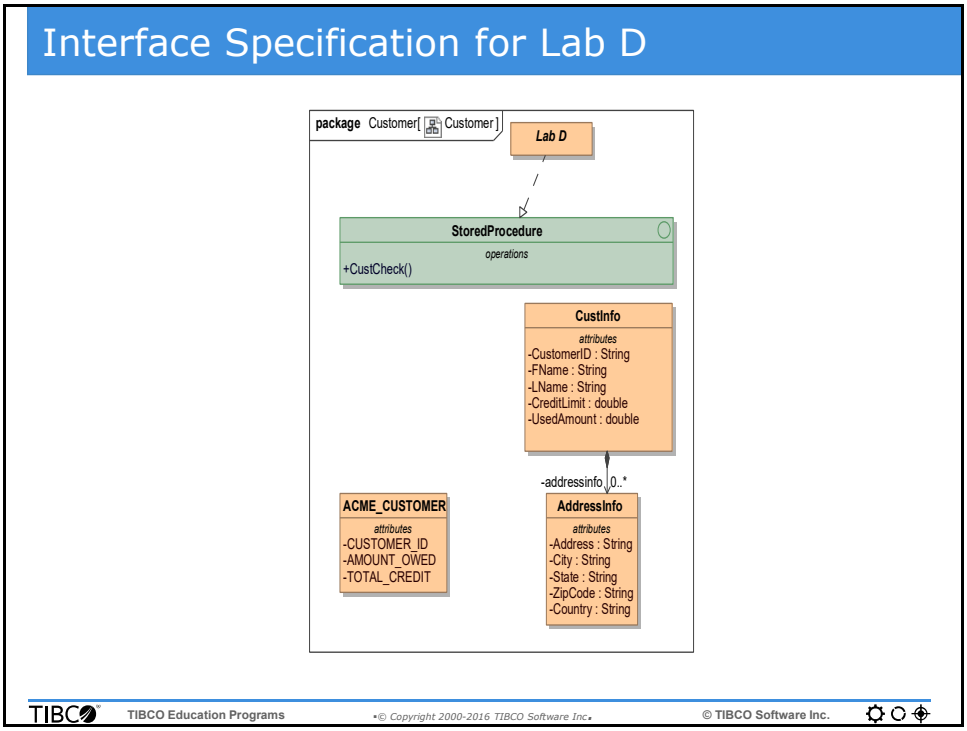
When a shared resource is used in a version 6 application, process properties and module properties referencing these process properties are automatically created. In the example above, the JDBC activity used the JDBC shared connection, and this causes the properties to be created.

These properties are created regardless of whether the version 5 activity uses global variables or not. Recall that global variables are transitioned as module properties, from the earlier module.

You can examine the effects of using a JDBC activity. Simply click on the process boundary and then examine process properties.



Database Schema - Lab D



Interface Specification for Lab D

Review

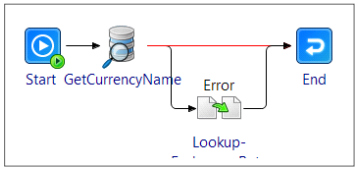
- Application runs as OSGi bundle at runtime
- JDBC drivers: embedded and embeddable
- Business Studio for JDBC design time setup
- Database perspective: browsing, interaction
- For runtime, JDBC bundle must be created

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Import JDBC Process
  - Test application
- Look up orders in a database
  - Build query for look up
- Import process that uses stored procedure
  - Test



TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Lab Overview

Notes

Lab D Database Connectivity

Overview

In this lab you will update your application to look up an Oracle database table for the status of your order. To do this, you will configure and deploy the Oracle JDBC driver as an OSGi bundle.

Objectives

- *Design Time Connectivity, page 118*
- *Running JDBC Based Applications,* page 127
- *Using Stored Procedures,* page 131

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous lab has been completed

Directions

- Complete the exercises that follow

Lab D - Exercise 1: Design Time Connectivity

Overview

In this exercise you will define a JDBC connection resource, and complete the steps required to make the JDBC driver available to your design time environment. You will create the required database tables, and update the process to select values from the database.

Steps

1. Launch TIBCO Business Studio and open your workspace.

- Launch Business Studio for workspace: `/home/tibco/bw6edu/workspace`
  - Click **OK**
  - The layout is restored as you left it in the previous exercise

2. Add a JDBC resource that uses an Oracle database.

- Browse to *ACMEShared* > *Resources* > ***acmeshared***
  - Right-click and select *New* > *JDBC Connection*
  - Resource Name: **`ACMEOracle`**
  - Click **Finish**
- Enter database details (where xx is your student id, user01, user02, etc.):
  - User name: **`userxx`**
  - Password: **`userxx`**
  - Database Driver: **oracle.jdbc.OracleDriver** (select from list)
  - Database URL: **`jdbc:oracle:thin:@edusvr:1521:edusvr`**

3. Test connection and fix any errors in configuration.

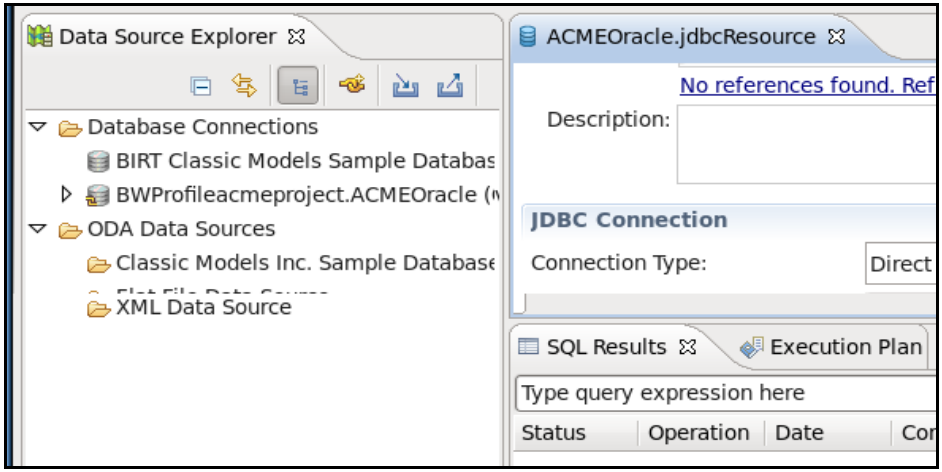
- Click **Test Connection** button
  - An error is returned next to the button: "*Invalid driver definition: JAR list not defined*"
- Launch *File Browser*
  - Browse to `/oracle/client_1` (shortcut created in file browser)
    (This link points to */opt/oracle/app/oracle/product/11.2.0/client_1*)
  - Now browse to `jdbc/lib` folder and copy **ojdbc6.jar**
- In *File Browser*, browse to
  `/opt/tibco/soa/bw6/bw/6.3/config/design/thirdparty`
  - **Paste** the **ojdbc6.jar** file in this folder (next to the *postgres and sqljdbc* jars)

Tip: Business Studio does not have to be restarted when you make this change.

- Back in Business Studio, **Test Connection** and it should work

4. Open the database perspective to view database connection.

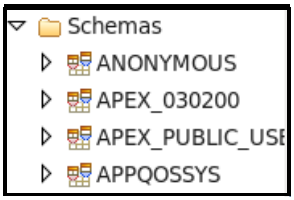
- From the top menu in Business Studio, choose *Window* > *Open Perspective* > ***Other...***
  - Now choose **Database Development** and click **OK**
- The database perspective is displayed



- Notice under database connections, a connection to **ACMEOracle** has been created

5. Review the database perspective options and close this perspective.

- Expand the ***BWProfileacmeshared.ACMEOracle > edusvr > Schemas*** folder
  - Notice the multiple user schemas defined for the database including userXX (XX= id)



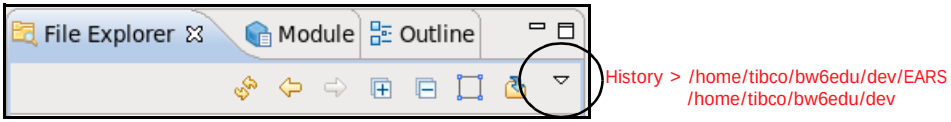
- On the bottom right hand side, notice the window where you can type queries and get results using the database connection (SQL Results Tab)
- Finally, right-click on the connection and notice the **Save Offline** and **Work Offline** modes

Tip: Save offline: downloads schemas to local files.
Work offline: Continue development when the database is not connected

- From the top menu, choose *Window > Close Perspective*
- Switch to the **Design** perspective

6. Verify File Explorer can view SQL files for class.

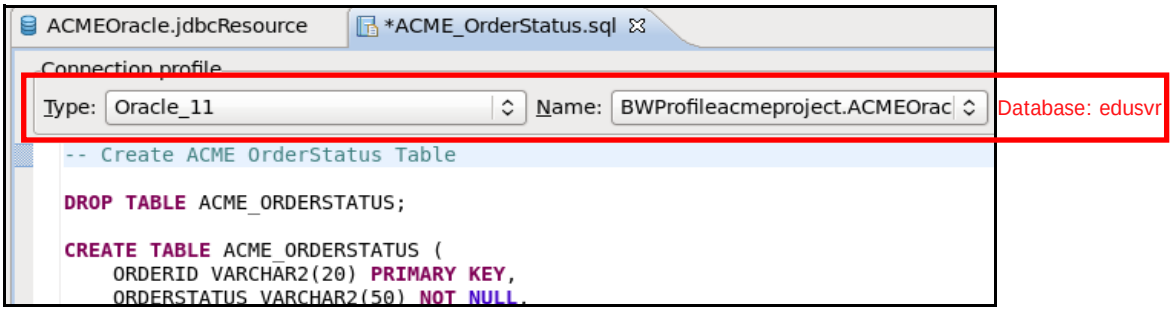
- The *File Explorer* (tab in bottom left window) shows *bw6edu/dev* folder



- Now expand the **scripts** folder, and notice the SQL Script files (.sql) are visible
- Save your project

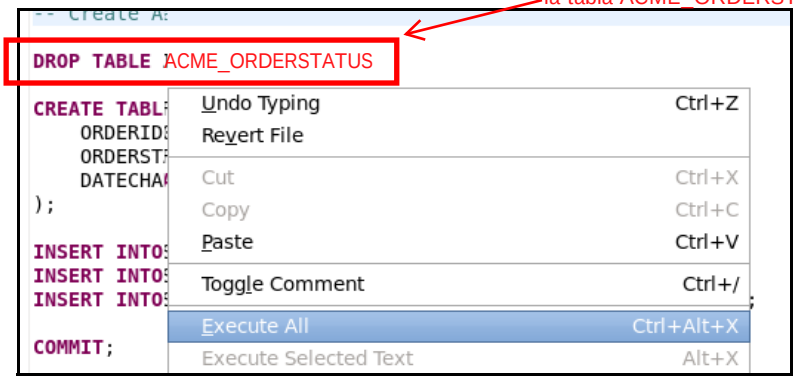
7. Execute the SQL script to create the order status table.

- In the *Design perspective*, confirm that the JDBC connection details are visible
  - If not, open the JDBC connection resource details
- In File Explorer, expand scripts folder
  - Select ACME_OrderStatus.sql
  - Drag it to the **General** section (*top half*) of the JDBC Connection
- This opens a new view (called SQL Scrap Book)
  - In this view, locate Name field, and set it to BWProfileacmeshared.ACMEOracle



- You are now connected to your Oracle database
- Right click anywhere on the script itself and choose **Execute All**
  - For "Error during execution, continue?" click **Yes**

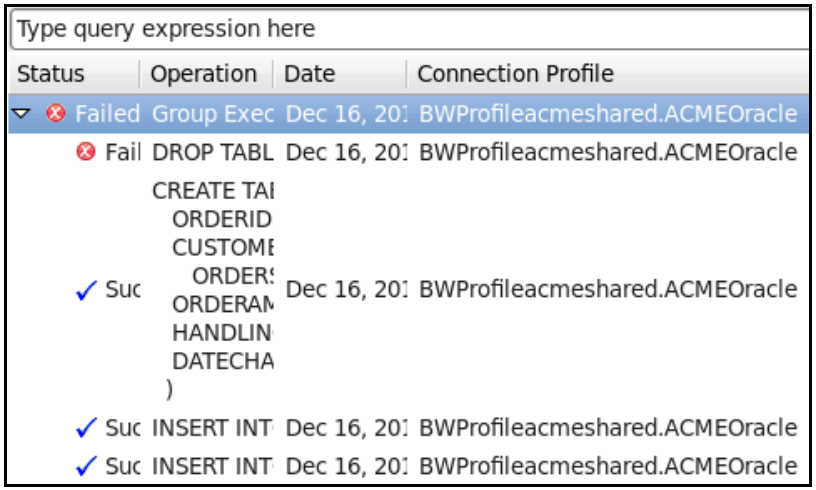
El error se refiere al primer comando. Se hace drop a la tabla ACME_ORDERSTATUS que aún no existe.



Tip: This executes the script and defines the ACME_OrderStatus table and inserts three rows into the table.

8. Verify the results of your script execution.

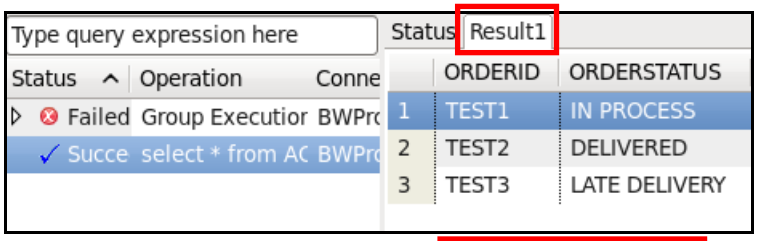
- Below the script window, you should see the SQL Results window
  - It will show a status of **Failed**, since the DROP TABLE command fails
  - View the details by expanding the view as shown:



- Replace the SQL Script with the following script

```
SELECT * FROM ACME_ORDERSTATUS
```

  - Right-click and Execute all
  - Results window indicates three pre-defined orders as shown:

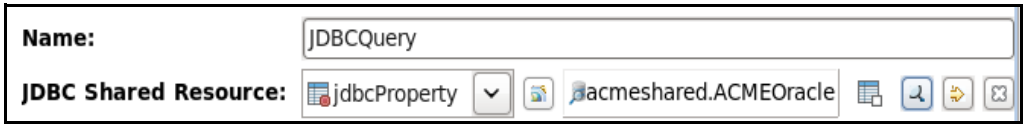


  - Close the *SQL Scrap book* view (window) - **do not save changes**
  - Also close the *SQL Results* view

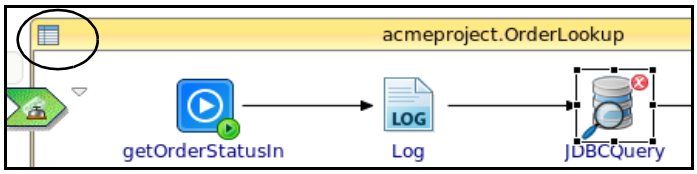
Tip: There are many features to the database design perspective, which are not covered here. You can read about them in the Eclipse documentation.

9. Add a JDBC activity to your process and set it to use your JDBC connection.

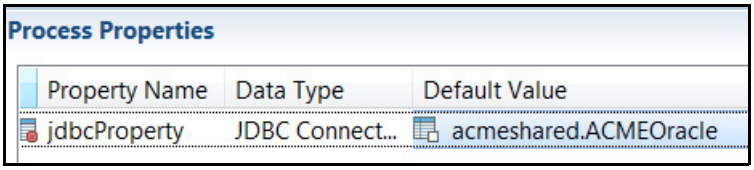
- Close the **JDBC connection** view
- Open *ACMEAppModule> Processes > acmeappmodule >* ***OrderLookup.bwp*** (double-click)
- From Palettes, choose *JDBC* ***> JDBC Query activity***
  - Drop the activity between the **log** and **getOrderStatusOutOp activity**
  - This connects the activity to **log** and **getOrderStatusOutOp**
- Select the JDBCQuery activity
  - In **Properties**, notice setting for JDBC Shared Resource
    - *jdbcProperty* is predefined, and it is a process property
    - For **value**, browse (using magnifying glass)
    - Choose *acmeshared.ACMEOracle*



  - **Save** your changes
- Notice the process definition window, a new icon has appeared as shown:



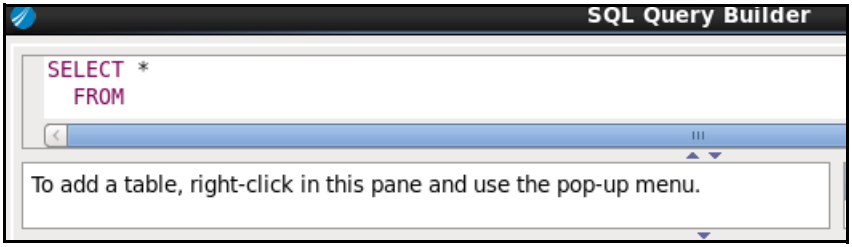
  - Click on this Process Properties icon, notice a new process property *jdbcProperty* has been created
  - It maps to your JDBC connection



10. Build a SQL query to fetch records from the order status table.

- Select *JDBCQuery* > *Properties* > ***Statement*** tab (partial view shown)
  - Locate and click the **SQ** button, to launch SQL Query Builder

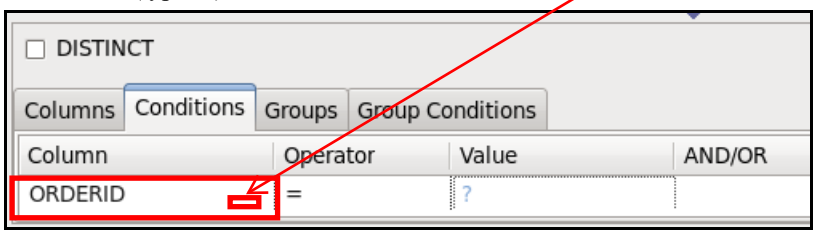
- Construct the select statement:
  - Delete the word *TABLENAME*



  - Right-click in the box shown, and choose **Add Table**
  - Browse and select **USERXX.ACME_ORDERSTATUS**
  - The statement changes to the following:

```
SELECT * FROM USERXX.ACME_ORDERSTATUS
```

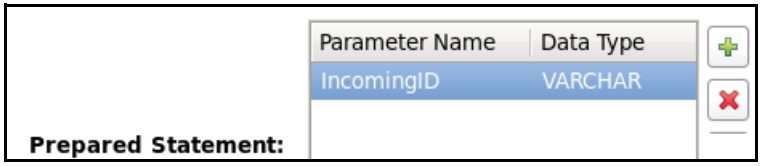
- Click on the conditions tab
  - Column: **ORDERID** (click beneath column and select from drop-down list)
  - Operator: = (select)
  - Value = **?** (type in)



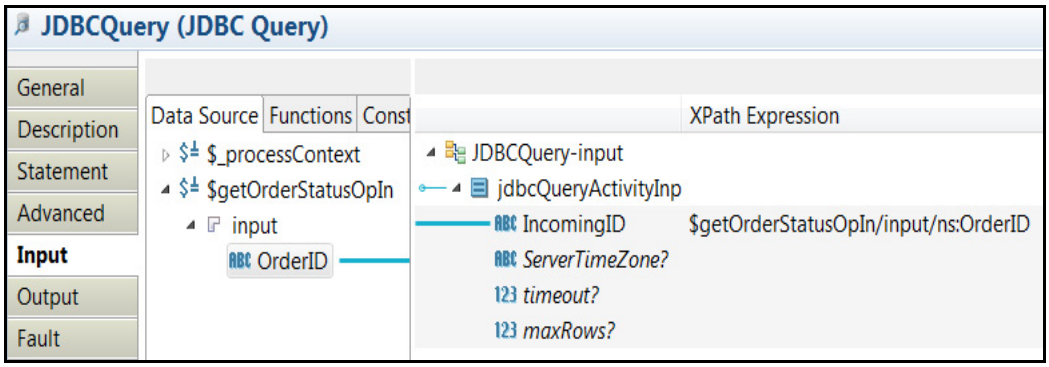
  - The "**?**" parameter will be substituted by an incoming OrderID
  - Click **OK** to close SQL Query builder

11. Add a parameter which will map to the incoming order id.

- Still in the JDBC Query activity, add a parameter
  - In Prepared Statement, click +
  - A new Parameter named "parameter" is added
  - Change its name to **IncomingID**



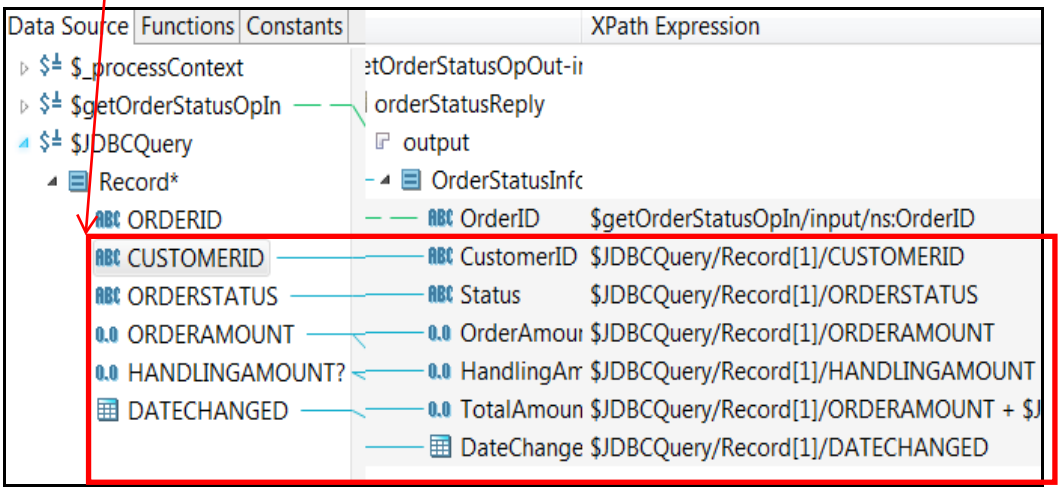
  - Click **Fetch**
  - **Save** your changes
- Map the parameter to the incoming OrderID
  - In Properties, click **Input** tab
  - Map `getOrderStatusOpIn > input > OrderID` to `IncomingID` as shown



  - **Save** your changes
  - There should be no errors (problems) with your process

12. Update the service response to use database query values.

- Choose the **getOrderStatusOpOut** activity
- In *Properties* > ***Input***
  - Map values from `JDBCQuery > Record`
  - TotalAmount = **OrderAmount + HandlingAmount**
- The database query may return more than one record. So you see a <<filter>> element
  - Replace <<filter>> with 1 for both mappings



Hacer los mapeos y seleccionar la opción en la ventana pop-up "Replace existing expression"

- **Save** your changes
- Keep project open for the next exercise

Lab D - Exercise 2: Running JDBC Based Applications

Overview

In this exercise you will use Business Studio to test your application module. You will create an OSGi bundle for Oracle Driver so that it can be used by the BusinessWorks application at run time.

Steps

1. Run your solution and review the runtime error returned by the engine.

- Right-click on OrderLookup process and choose **Run BusinessWorks Applications**
  - The process runs but the application is **impaired**

```
Started by BusinessStudio.
0002: BW Engine [Main] started successfully.
 Started OSGi Framework of AppNode [BWEclipseAppNode] in AppSpa
18: Deploying BW Application [ACMEApp:1.0].
21: All Application dependencies are resolved for Application [
19: BW Application [ACMEApp:1.0] is impaired.
0554cf3da)] c.t.b.s.r.d.ReferenceDependency - TIBCO-BW-SR-FRWK-
source/httpconnector}HttpConnectorConfiguration.b25d1829-d530-4
```

  - This indicates that application will not process requests
  - The BusinessWorks engine continues running
- Scroll to the right, to locate the full error message which is as shown:

```
TIBCO-BW-SR-FRWK-503000: Unable to start SharedResource
[acmeproject.ACMEOracle] from Module
[ACMEProject:1.0.0.qualifier], DeploymentUnit
[ACMEApp:1.0]. <Reason>: TIBCO-BW-SR-JDBC-500003: The
database driver [oracle.jdbc.OracleDriver] is not found.
Ensure that DataSourceFactory bundle providing this driver
is available in the environment.
```

- **Stop** the BusinessWorks engine
- **Shut down** Business Studio

2. Copy the JDBC Jar file to a specific location, to create a bundle.

- Launch *File Browser*
  - Browse to `/oracle/client_1` (shortcut created in file browser)
  - Now browse to `jdbc/`**`lib`** folder and copy **ojdbc6.jar**
- In File Browser, browse to `/opt/tibco/soa/bw6/bw/6.3/`**`config`**
  - Now go to `drivers/shells/jdbc.oracle.runtime/runtime/`**`plugins`**
  - Keep going to **com.tibco.bw.jdbc.datasourcefactory.oracle/lib**
  - **Paste** the **ojdbc6.jar** file here

3. Create the bundle using the bwinstall command.

- In a Terminal window, change directory to `/opt/tibco/soa/bw6/bw/6.3/`**`bin`**
- Type this command to view options for **bwinstall**

  `bwinstall help`
- Run the command to create an OSGi bundle for the Oracle driver

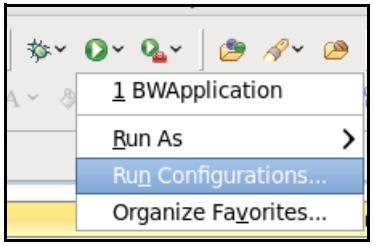
  `bwinstall oracle-driver`
- Review the output displayed in the console:

```
     [copy] Copying 1 file to /opt/tibco/bw6/eclipse-
.eclipse.equinox.p2.touchpoint.eclipse/plugins/com.ti
     [copy] Copying 1 file to /opt/tibco/bw6/eclipse-
.eclipse.equinox.p2.touchpoint.eclipse/plugins/com.ti
     [copy] Copying 1 file to /opt/tibco/bw6/eclipse-
.eclipse.equinox.p2.touchpoint.eclipse/plugins/com.ti
     [copy] Copying 1 file to /opt/tibco/bw6/eclipse-
.eclipse.equinox.p2.touchpoint.eclipse/plugins/com.ti
install-oracle-driver:
BUILD SUCCESSFUL
```

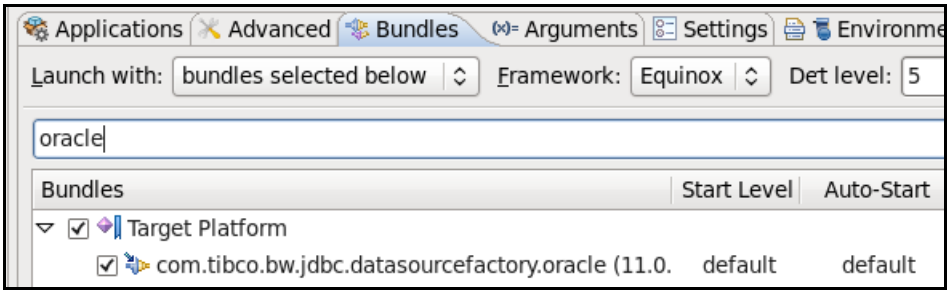
- If you see the copying messages and BUILD SUCCESSFUL message, it indicates that:
  - Bundle was built successfully
  - All bundle files were copied to the correct folders
  - The bundle is now a part of **product features**
    - It will be available to any application *run on this machine*

4. Verify that the bundle is visible in Business Studio.

- Launch TIBCO Business Studio and load *workspace*
- In the Run menu, choose *Run Configurations...*



- In the left menu, click *BusinessWorks Application* > ***BWApplication***
  - Now choose **Bundles** tab in right hand window
  - In *type filter text* enter (type) **oracle** and verify that the oracle bundle is available and selected



5. Run the application and observe another runtime error.

- Run the application and it fails, scroll through the output and locate the cause of the error

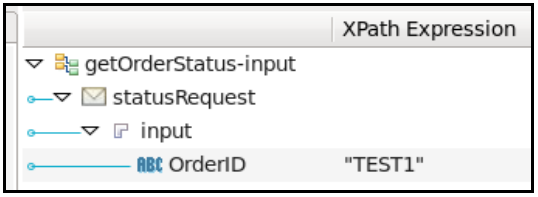
```
CMEProject.Log - Received request for OrderID; TESTORDER123
re - TIBCO-BW-CORE-500050: The BW process [acmeproject.OrderLookup] instance faulted,

 Activity input data validation error. The input data for activity [getOrderStatusOut]
StatusInfo)  {http://genxdm.org/typed/local-types}type-0

}OrderID)  {http://www.w3.org/2001/XMLSchema}string
}OrderStatus)  {http://www.w3.org/2001/XMLSchema}string
}DateChanged)  {http://www.w3.org/2001/XMLSchema}dateTime
```

6. Remove the cause of the error and run the application again.

- Stop the BusinessWorks engine
- The error happens because the tester process submits an OrderID: TESTORDER123
  - This OrderID does not exist in ACME_OrderStatus, it has TEST1, TEST2, and TEST3
  - Open the ~~TimerModule~~ TimerModule>Processes>Timer>OrderStatusClient process
  - In **getOrderStatusOp**, set **OrderID** to `TEST1 (or TEST2 or TEST3)`



- **Save** your changes and **Restart** the BusinessWorks engine
  - Confirm that the process runs without any error
  - Stop the engine when done

```
TIBCO-THOR-FRWK-400020: The contituent [TimerModule{<none>, [bw-appmodule,osgi-bundle
HOR-FRWK-300019: BW Application [Timer.app:1.0] is impaired.
HOR-FRWK-300006: Started BW Application [Timer.app:1.0]
g.TimerModule.Log - ******* Started Timer Application at: 2:26:51 PM **********
g.ACMEAppModule.Log -  Received Lookup Request for OrderID: TEST1
```

7. Export your solution built so far as ACMEJDBC1.zip.

- Choose *File* > *Export* and then *General* > *Studio Projects to* ***Archive***
  - In the next window, choose all
    - To Archive File: `/home/tibco/bw6edu/dev/exports/`**`ACMEJDBC1.zip`**
    - Save in ZIP format
- Click **Finish**

  Well Done! You have looked up a database table, and resolved a couple of runtime errors!

Lab D - Exercise 3: Using Stored Procedures

Overview

In this exercise import a process using a JDBC stored procedure activity. Complete project and test.

Steps

1. Create required database tables using provided script.

- Launch SQLPlus using the following command in a new Terminal window:
  `sqlplus userxx/userxx@edusvr`
  Here, replace XX with the unique ID assigned by the instructor
  - The SQL Plus prompt is displayed
  - Execute the following command:
    `@/home/tibco/bw6edu/dev/scripts/ACME_Customer.sql`
  - Execute the following command:
    `@/home/tibco/bw6edu/dev/scripts/ACME_Procedure.sql`
- The command executes successfully
- Exit SQLPlus prompt

2. Import the JDBC project into your BW6 workspace.

- In Business Studio open Workspace `/home/tibco/bw6edu/workspace`
- From the top menu, choose *File* > ***Import...***
  - Choose *General* > *Existing Studio Projects into Workspace,* click next
  - Choose **Select Archive File,** and browse and select `/home/tibco/bw6edu/dev/others/ACMECustomerChkv1.zip`
  - Confirm both projects are selected and click **Finish**

3. Examine processes and other artifacts included in the imported project.

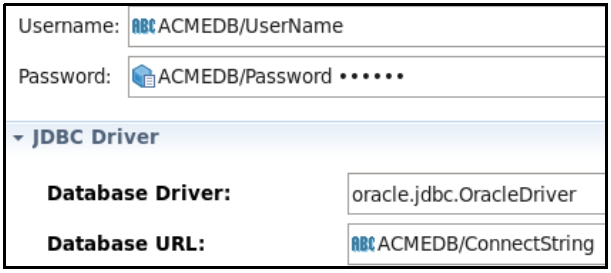
- Verify following resources get created
  - In *Processes> acmecustomerchk* folder:
    - CheckCustCredit.bwp
    - CheckCustCreditService.bwp

4. Update JDBC settings and test connection to the database.

- Navigate to *ACMECustomerCheck* > *Modules Descriptors* > ***Module Properties***
  - Under *User* Tab verify variables and database connection are displayed as shown
    - **Update** values for *UserName* and *Password*, with your assigned **UserID**

Groups / Properties	Type	default
ACMEDB		
ConnectString	String	jdbc:oracle:thin:@edusvr:1521:edusvr
Password	Password	••••••
UserName	String	userxx

- In **Resources** > *acmecustomerchk* folder, open **JDBCOrcl.jdbcResource**
  - Verify the three module properties are set as shown:

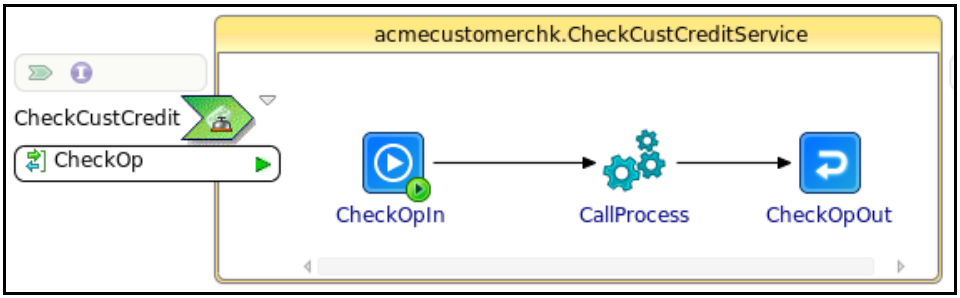


- **Test Connection** and it should work

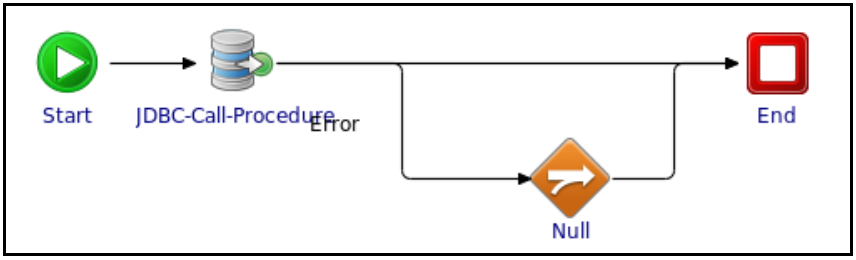
Guardar cambios

5. Examine the CustomerCheck service.

- Open *CheckCustCreditService* process and see it details:
  - This is the *Service*



  - This process uses a call process activity (non-WSDL)
- Open the process *CheckCustCredit*, rearrange all activities to see them properly



- Select the *JDBC-Call-Procedure activity* > *Properties* > ***Procedure*** tab
  - Examine value for Schema: ACMEDB/Username
- **Save** Changes
- There should be no errors in your application module

6. Verify the BusinessWorks application includes the application module.

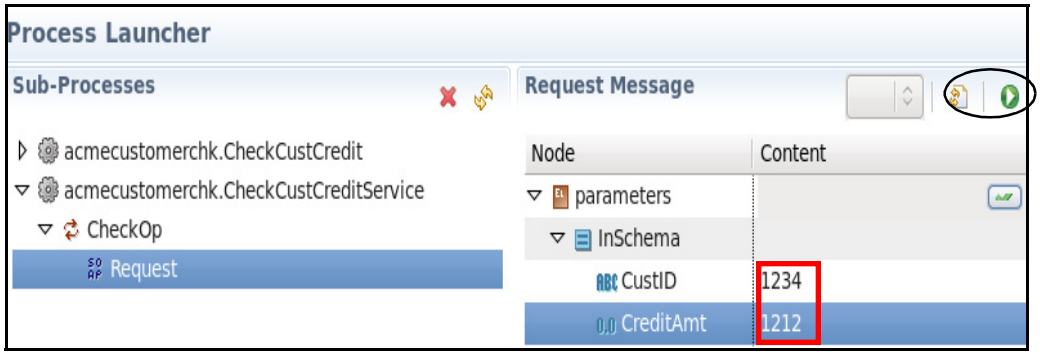
- In the application **ACMECustomerChk.application**
  - *Includes* tab: verify *ACMECustomerChk (1.0.0.qualifier)* module is included,
  - If not, use **Add** button to add it, **Save** changes, and **close** the application details view

7. Run your solution and verify that it starts up successfully.

- Choose **Debug Configurations** and select ***ACMECustomerCheck*** Seleccionar solo esta aplicación
- Confirm that the debug perspective is open
  (if it does not, it means you are in Run mode. *Stop* and *restart* in debug mode)
- Verify that the BW Application has been started

8. Using process launcher, test the process.

- In the debug perspective, locate the **Process Launcher** view
  - Expand *CheckCustCreditService* > *CheckOp (Double-click)* > ***Request***
    - Enter a **CustID** and **CreditAmt** as shown:

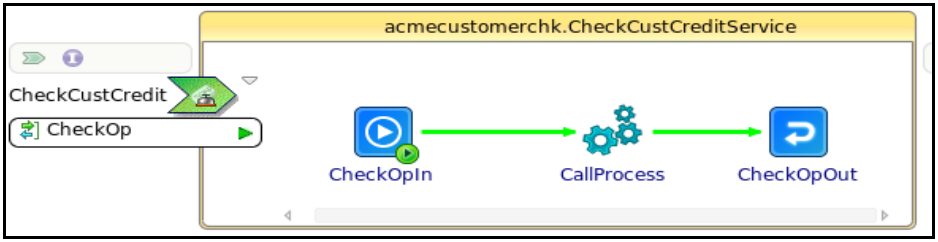


- Click on the Green **Run** button to run the sub-process
- **Note**: If the Green Arrow is grayed out, try these options:
  - Choose the Request field (right side)
  - Locate the Debug window (top left) > Highlight the launched line
    - The Start button in process launcher should now be visible

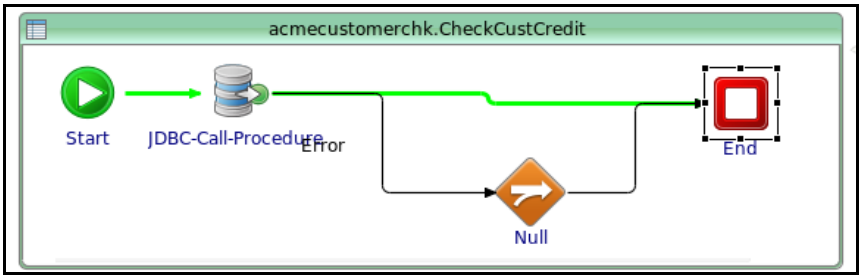
Lo que a mí me funcionó fue dar click en el tab BusinessWorks Jobs y regresar al tab Debug.
My workaround was clicking outside the BW Jobs tab and then click the Debug tab again.

9. Confirm that the sub process has executed successfully.

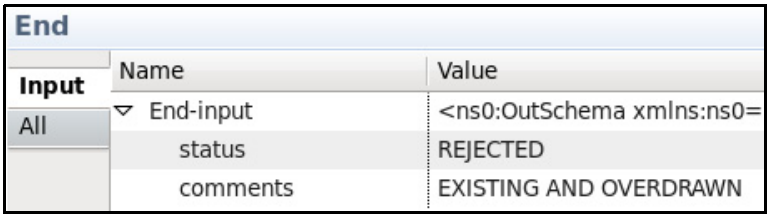
- Confirm that the sub-process has executed, viewing the transitions turning green



- Verify that the sub process takes the success path



- Verify that the process runs and returns a value of GOOD for credit check status
  - Step through the process and examine Job Data for various activities
- Re-run the process with a different CustID and CreditAmt (5858 and 11200)
  - Output for this data is shown (in End activity) :



End

Input | All

Name	Value
End-input	<ns0:OutSchema xmlns:ns0=
status	REJECTED
comments	EXISTING AND OVERDRAWN

- **Stop** the debugger when done testing and **Shutdown** TIBCO Business Studio

Note: If you get an Oracle runtime error:
- Confirm that Oracle database is accessible (Test Connection in JDBC resource)

Notes

Unit 5 Working with Properties

This unit describes properties and their usage. It also examines the types and usage of variables.

Topics

- *Objectives,* page 138
- *Variables and Properties,* page 139
- *Understanding Scope of Properties,* page 140
- *Accessing Process Properties,* page 141
- *Defining Process Properties,* page 142
- *Process Properties vs. Process Variables,* page 143
- *Process Property Features,* page 144
- *From Process Properties to Module Properties,* page 145
- *Module Property Details,* page 146
- *Types of Module Properties,* page 147
- *Application Properties,* page 148
- *Property Layers and Scope,* page 149
- *Application Configuration Profiles,* page 150
- *Variables,* page 151
- *Review,* page 152
- *Lab Overview,* page 153

Objectives

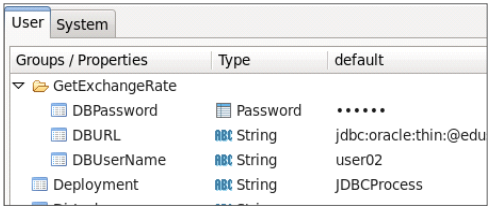
- Discuss module properties
- Differentiate between various property types
- Compare process properties and variables
- Explain usage of application profiles
- Examine types and usage of variables

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives

Variables and Properties

- Variables:
  - Process variables
  - Shared variables
    - ***Module*** and Job shared variables
- Properties:
  - Module Properties
    - Applied across ***module***
  - Application Properties
  - Process Properties



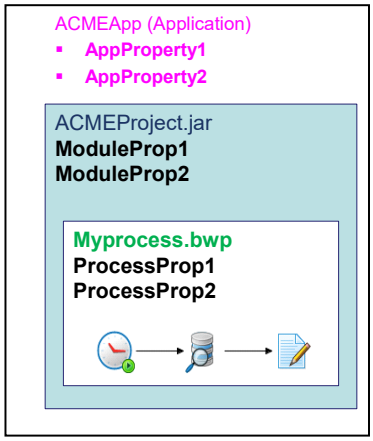
- Scope: ***Application > Module > Process properties***

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Variables and Properties

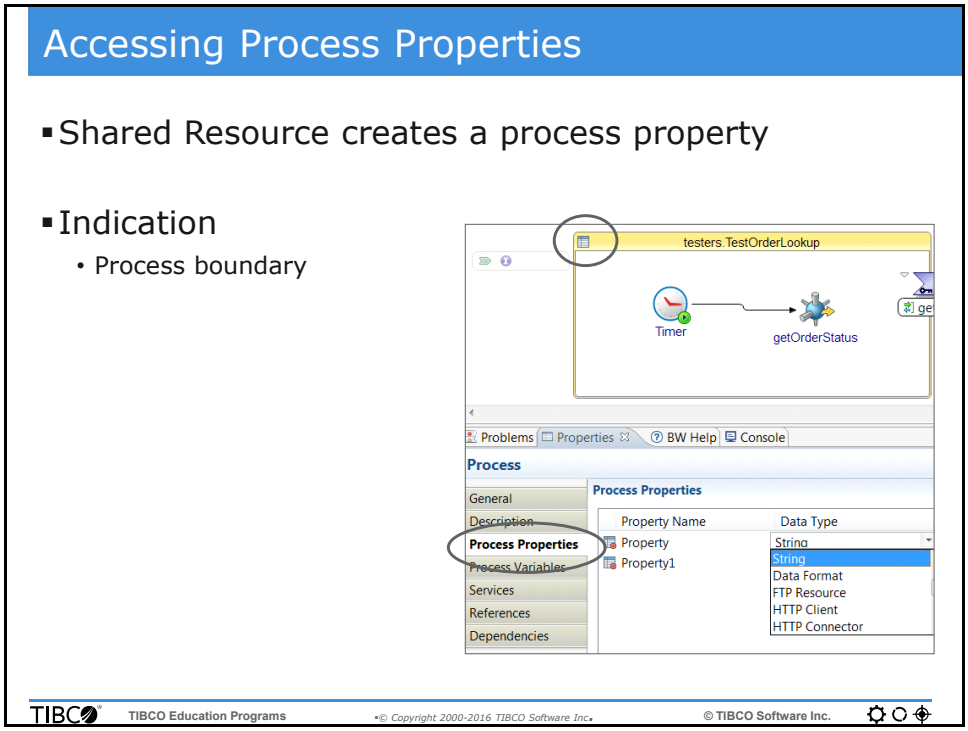
Understanding Scope of Properties

- Application level
  - For all modules
- Module level
  - For all processes
- Process level
  - Specific process



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Understanding Scope of Properties

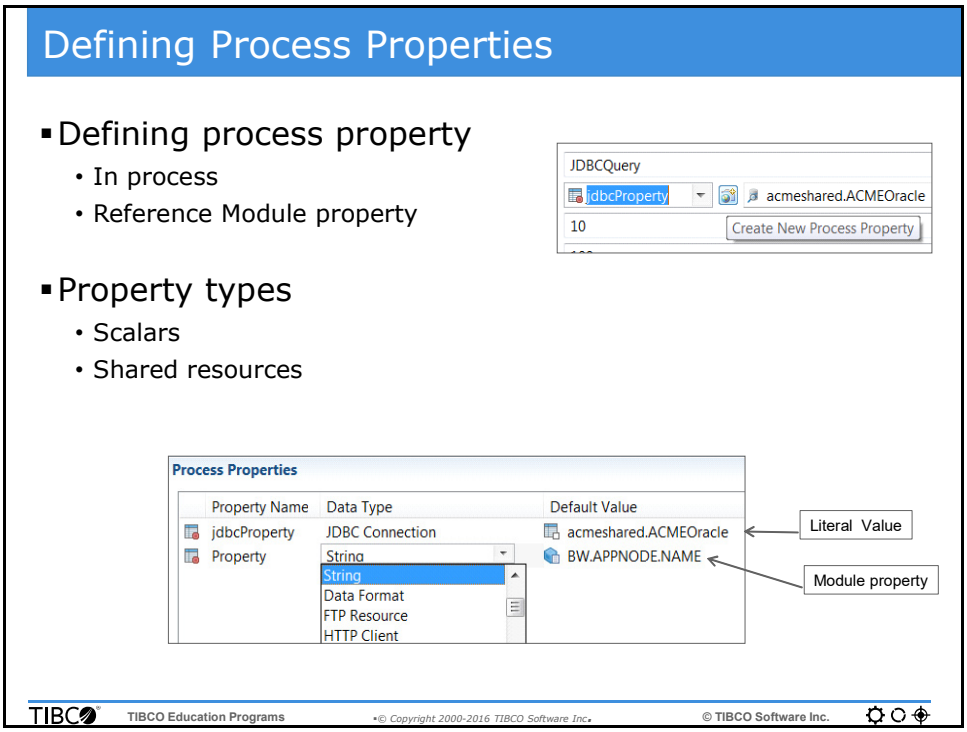


Accessing Process Properties

Process properties are visible within a module. The allowed values for a process property are literal, shared resource reference or a module property reference.

By default, a process does not have any pre-defined process properties.

When a process property is defined by the user, the process boundary changes and a new icon appears on the top left corner as shown above. You can click on the icon to access the properties, or in Properties tab, choose Process Properties section.



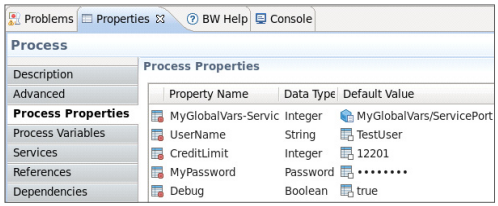
Defining Process Properties

Properties are used to define configuration. Depending on where and how they are defined and qualified, properties can be classified into application properties, module properties, process properties, and activity configuration properties.

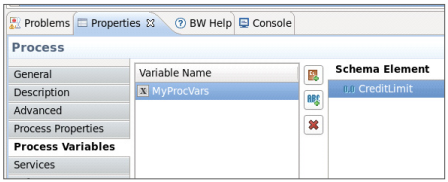
Properties defined in the inner layer can reference a property defined at the parent layer. For example, a process property can reference a module property instead of providing a literal value. Private properties are not visible to the encapsulating layers.

Process Properties vs. Process Variables

- Properties:
  - Built in editor
    - Value at creation
  - Stored with process
  - Recommended



- Variables:
  - XML Schema editor
  - Separate XSD
  - Values: Assign activity



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Process Properties vs. Process Variables

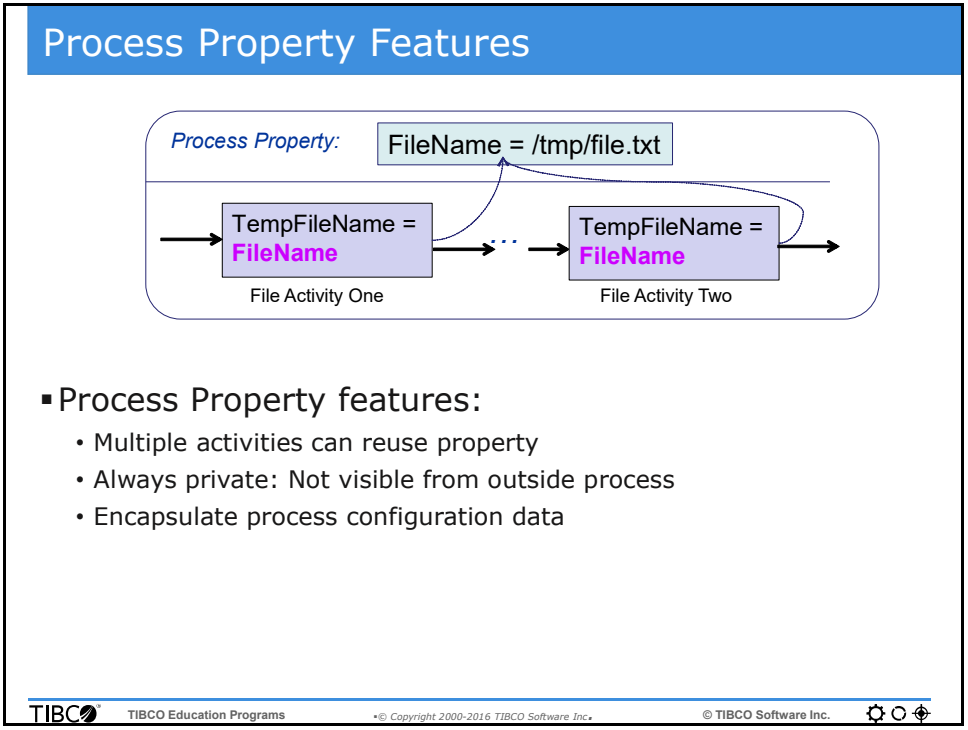
Process variables are also available in version 6. In fact, process variables from version 5 projects are transitioned as process variables in version 6.

Both process variables and process properties are defined at the process level as shown. However, process properties provide significant benefits over the classic process variables.

Process properties can be directly assigned values at creation time, instead of using the assign activity. Process properties use a built-in simple editor, whereas process variables use an XML editor, since these are stored in separate XSD files.

Module properties can also be injected into process properties, but this functionality is not available for process variables.

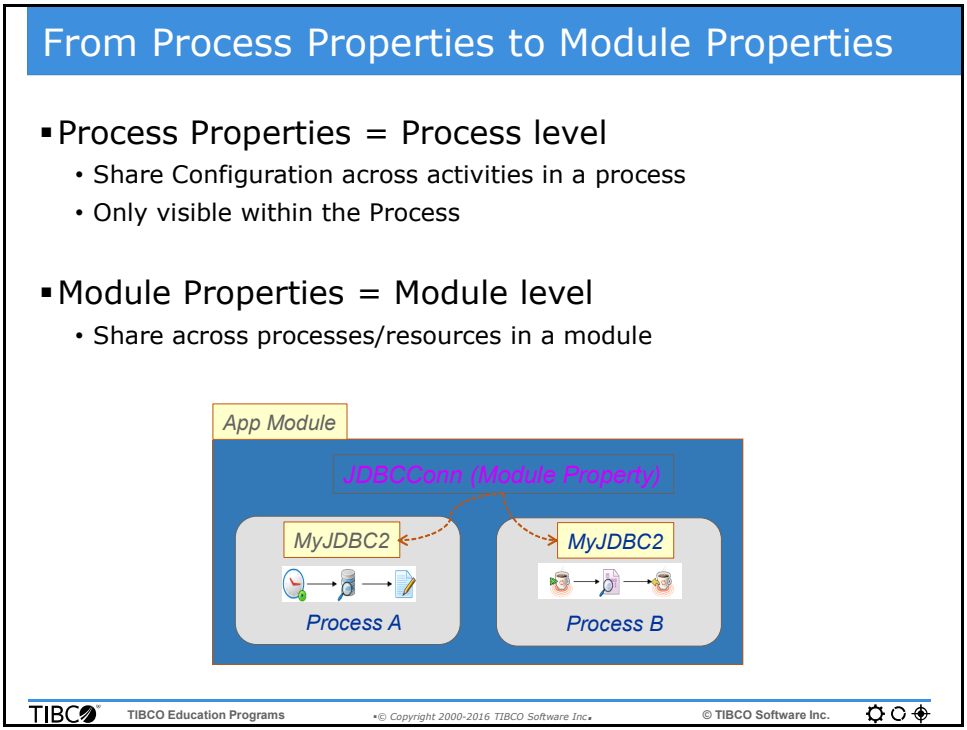
After you transition your version 5 project, consider converting process variables to process properties.



Process Property Features

A process property is defined for the whole process. As a result, any activity in the process can access the property to use its value.

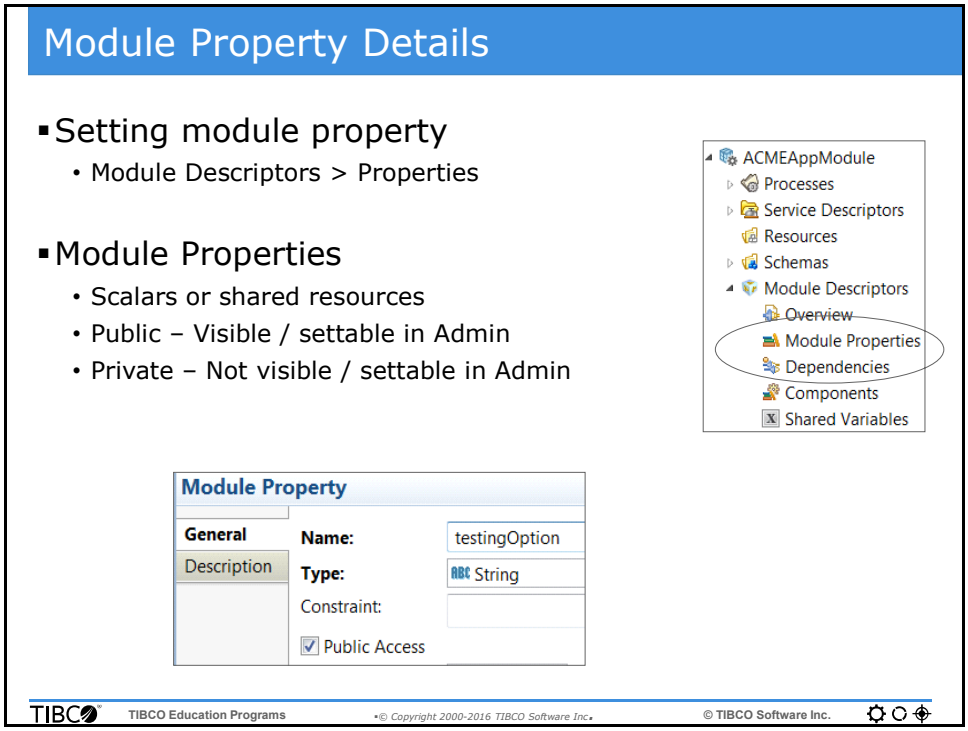
The process property is not visible outside the process, for example a sub process called by the process. This allows you to encapsulate data that is meant specifically for the process.



From Process Properties to Module Properties

Properties follow the layered configuration model where configuration is pushed from top to the bottom.

A module property defines a value that can be applied to all processes and resources in the module. A process property can use this value or override it with a value that is specific to a process.

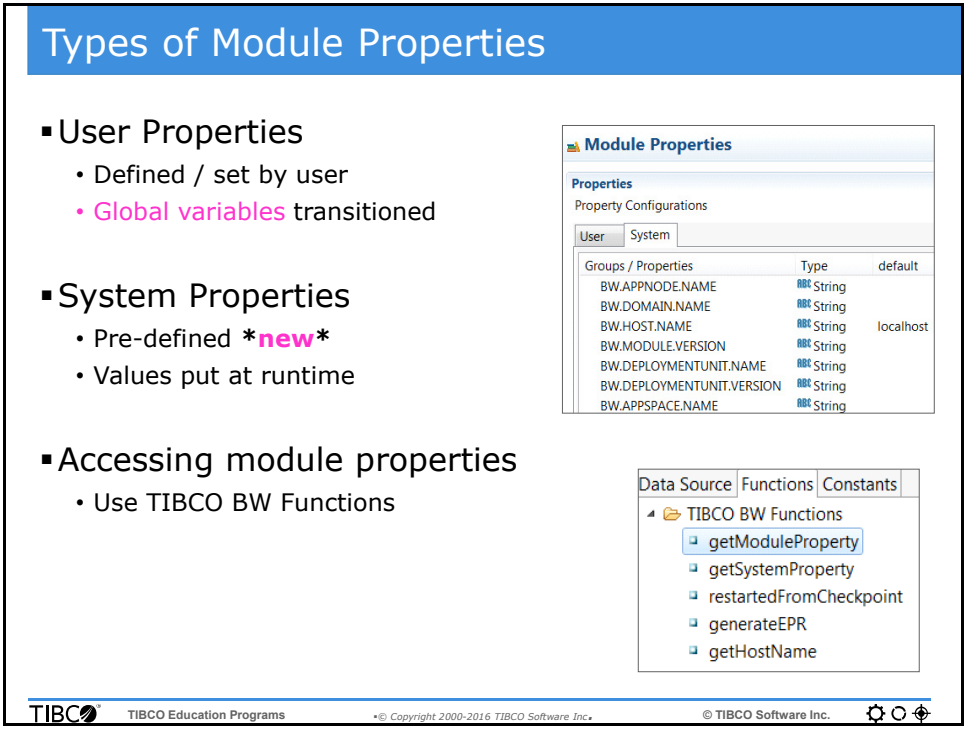


Module Property Details

Module properties are visible within the module. To access a process property, reference a module property from a process property and then reference the process property from the activity.

Visibility: Module properties are local to a module only. Thus, a module property defined in a shared module will not be visible in an application module, even if the application module references the shared module.

Module properties are for application modules and shared modules. The application has application properties, and these are examined later.



Types of Module Properties

This is a summary of the types of module properties available when developing BusinessWorks applications.

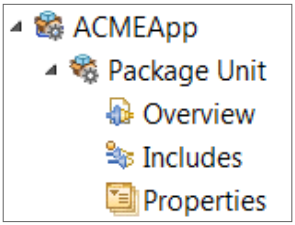
User properties are defined by the user at the module level (application module or shared module).

System properties are pre-defined properties which should not be deleted or used. These properties are used by the runtime environment to manage the application bundle.

You can access a module property (user or system) in a process activity. Simply use the appropriate TIBCO BusinessWorks function in your XPath mapping.

Application Properties

- Application properties
  - *New* for 6
  - Scope: all modules in application
- Module Properties
  - Reconfigure at Application level
  - Override during deployment
  - Syntax: **%%name%%**



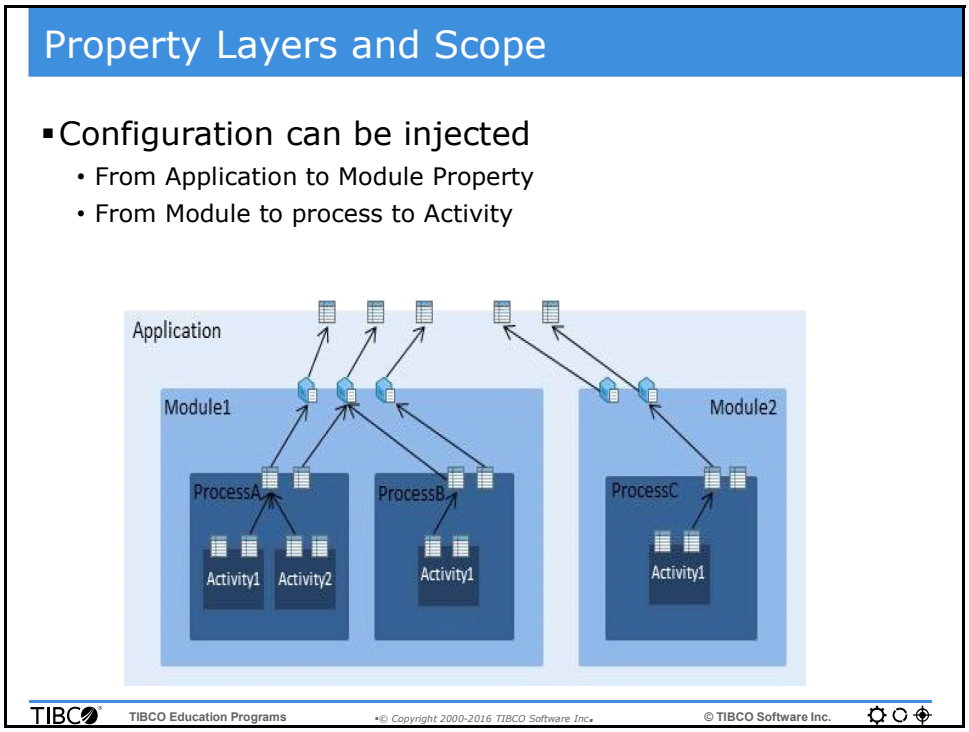
Properties	[default]
Application	
AppProperty1	
moduleProp3	122
moduleProp1	ABC
MyTestAppforVars.mod	
moduleProp3	%%moduleProp3%%
moduleProp2	••••••••
moduleProp1	%%moduleProp1%%

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Application Properties

Application properties displays all the module properties in the application. These properties are visible in TIBCO Enterprise Administrator. You cannot add new properties at application level.

Application properties overrides module properties enabling you to use different values for the same module.



Property Layers and Scope

Properties defined in the inner layer can reference a property defined at the parent layer. For example, a process property can reference a module property instead of providing a literal value. Public properties are visible to the encapsulating layers.

Choosing the right level ensures an easier to maintain list of properties in your application and keeps the number of properties at the application level to a minimum.

Application Configuration Profiles

- Profile: ***cool* feature**
  - Set of values for all defined properties
- Application can have multiple configuration profiles
  - One ***default*** profile
  - Created from all module properties
- Default profile can be changed
  - At design or deployment

Application	Module		Dev (default)	QA	Production
		HTTP Port	80	9123	8080
		JMS Queue	Test	Queue1	POBox

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Application Configuration Profiles

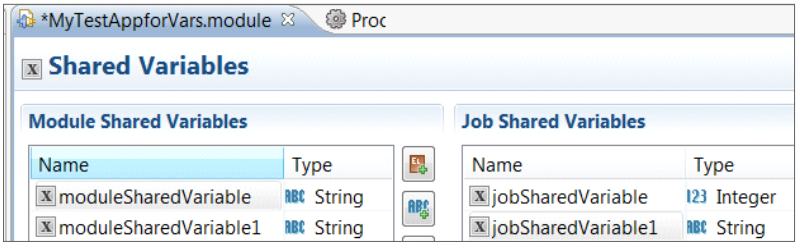
An application is created with profiles using TIBCO Business Studio. You can define multiple profiles when creating an application in TIBCO Business Studio.

A profile is the configuration of the variables an application uses. When an application is deployed with different variables, different profiles are available for each deployment. For example, you can create a Windows profile for an application that runs on a Windows machine and another for the same application running on a UNIX machine.

Each profile has a set of defined properties and values. The values use the appropriate operating system syntax to point to the files in the file system. The files are created and maintained outside of TIBCO Business Studio.

Variables

- Defined at various levels – determines scope
  - Process
  - Job
  - Module
- Values can be set / changed at runtime
  - getShared / setShared functions



TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. • © TIBCO Software Inc.

Variables

A process variable saves the state at the process level and a scope variable saves the state within the scope.

Variables defined within a scope are visible only within the scope. If the scope variable name is the same as a process variable name, then the scope variable takes precedence over the process variable within the scope.

Shared variables are used to save the state. There are two types of shared variables:

- Module shared variable saves the state at a module level.
- Job shared variable saves the state for the duration of a job.

Review

- Types of Properties: Process, Module, Application
- Global variables == module *user* properties
- Use process properties instead of variables
- Application properties: Highest level of setting
- Application profile: Set of values

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Define module properties
  - Define profile
  - Update tester process
- Run solution
  - Change default profile
  - Run tester process

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab E Properties and Variables

Overview

In this lab you will update your application to use module properties. You will define a NewOrder server and update the tester process to invoke the right service based on a module property. You will also examine profiles.

Objectives

- *Working with Module Properties, page 156*
- *Using Module Properties,* page 160
- *Types of Properties,* page 172

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous lab has been completed

Directions

- Complete the exercises that follow

Lab E - Exercise 1: Working with Module Properties

Overview

In this exercise you will define module properties using the different options provided by Business Studio.

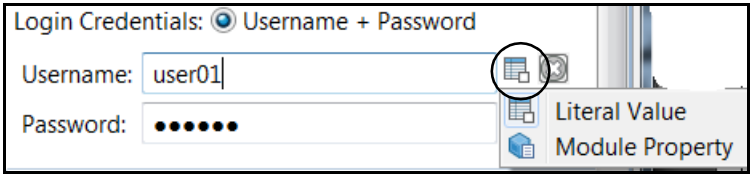
Steps

1. Launch TIBCO Business Studio and open your workspace.

- Launch Business Studio for workspace: `/home/tibco/bw6edu/workspace`

2. Update the JDBC resource to use a module property for the user name.

- Browse and open *ACMEShared* > *Resources* > *acmeshared* > ***ACMEOracle.jdbcResource***
  - Locate the **Username** field
  - Click on the icon circled below
    - Choose **Module Property**

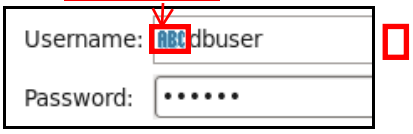


  - The value of Username is blanked out
  - Type in the following value (where XX is your assigned user ID) and click "Create a Module Property" icon

    `dbuser userxx`

  - Select (Press and hold the **<Enter> key** for a couple of seconds)

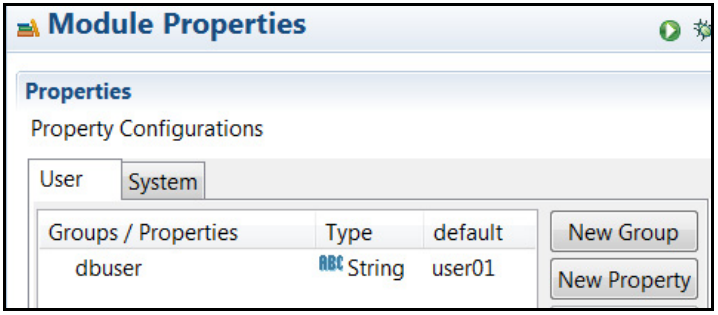
Alternativamente, seleccionar la lista de valores y elegir dbuser



  - **Save** your changes

3. Examine the result of your update actions.

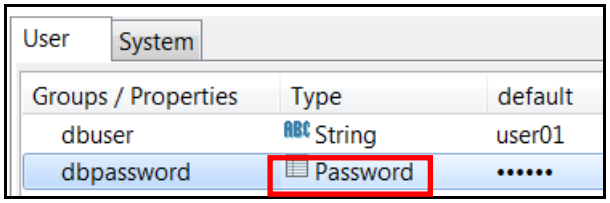
- In Project Explorer, browse and open *ACMEShared* > *Module Descriptors* > ***Module Properties***
  - Notice that a new Module Property has been defined as shown below:
    - Name: dbuser, set the default value to **userXX**



    - Stay in this view

4. Add a new property for database password.

- Click on **New Property** button and select the **newProperty**
- Select the *Properties* > *General* tab
  - Name: **`dbpassword`**
  - Type **`Password`** dificil sacar la lista
  - **Save** the Changes
  - Default (set value in Module Properties *User* tab): **`userXX`** (hit <enter>)
  - Verify your configuration in the *Module Properties* > *User* tab

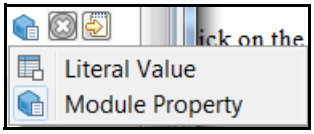


Tip: Passwords are obfuscated for security.

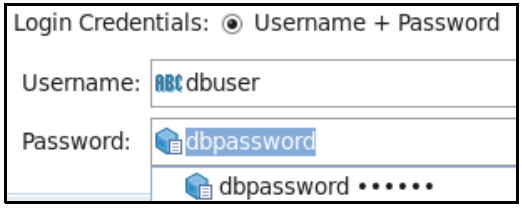
- **Save** your changes

5. Update the JDBC resource to use the password property.

- Locate and open the *ACMEOracle.jdbcResource* view
  - Locate the Password field
  - Click on the option to select Module Property



- Now select the module property from the drop down list as shown



- **Save** your change
- Click **Test Connection** and verify that it works as before

6. Define a module property for the http port used by the HTTPLookup resource.

- Open *Resources* > *acmeshared* > ***HTTPLookup.httpConnResource***
  - Update **Port** to use a **Module Property**
  - Type in the following value: **httpOrdering 9797** (Press and hold the **<Enter> key**)
- **Save** your changes
- Verify that the module property has been created
- Go to *ACMEShared* > *Module Descriptors* > ***Module Properties***

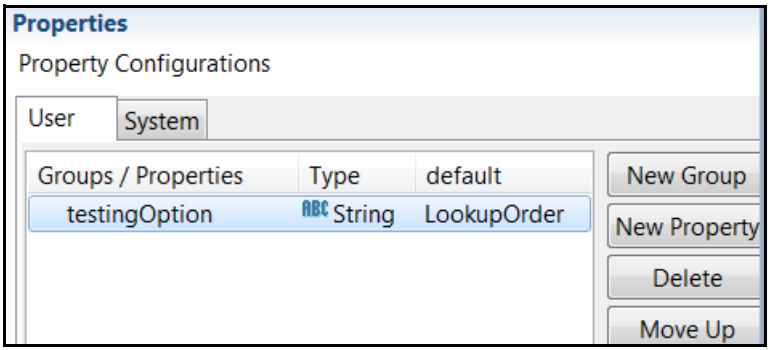
Alternativamente, seleccionar la lista de valores y elegir `httpOrdering`

Groups / Properties	Type	default
dbuser	String	user05
dbpassword	Password	••••••
httpOrdering	Integer	9797

- **Save** your changes and **close** this properties view

7. Add new module property in TimerModule to manage the tester process.

- In *Project Explorer*
  - Open: *TimerModule* > *Module Descriptors* > ***Module Properties***
- Confirm that the **User** tab is selected
  - Click the **New Property** button
  - Define a new property
    - Name: **`testingOption`**
    - Type: **String**
    - default: **LookupOrder**



- **Save** your changes
- **Close** the properties view
- Keep project open for the next exercise

Lab E - Exercise 2: Using Module Properties

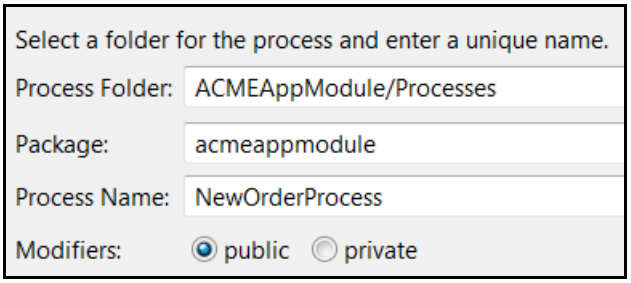
Overview

In this exercise you will define an empty process for new order processing. You will also update the tester process to test new order or order lookup, based on a module property.

Steps

1. Define new process to implement new order processing.

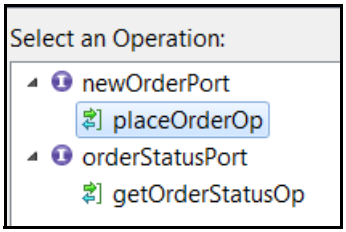
- Browse to *ACMEAppModule* > *Processes* > ***acmeappmodule***
  - Right-click and add a new **BusinessWorks Process**
  - Name: **`NewOrderProcess`**
  - Patterms: *Empty Process*



  - Click **Finish**
- A new empty process is created and opened
- **Save** your changes

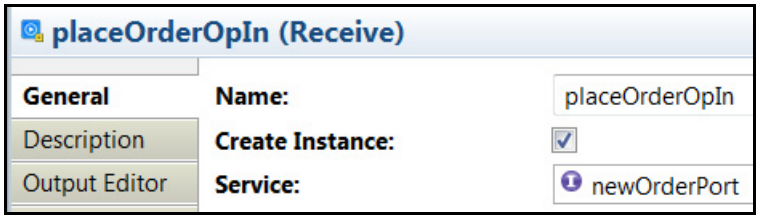
2. Update this process to implement the NewOrder operation.

- From *ACMEShared* > ***Service Descriptors***
- Drag **ACMEOrderWSDL.wsdl** and drop it on the process
  - Choose **Implement Operation,** and then choose **placeOrderOp**
  - Click **OK**



3. Update the process so that it is instantiated when a request is received.

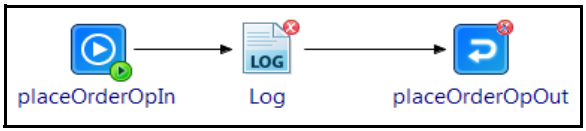
- Locate the **placeOrderOpIn** activity and select it
  - Examine the Properties for this activity (click *Properties* tab)
  - Create Instance: **Check**



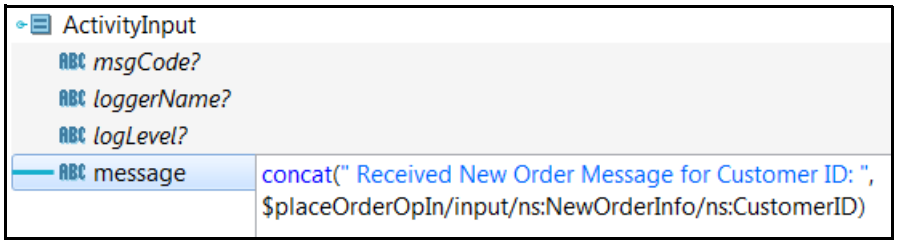
  - **Save** your changes
- Click *Output Editor* tab
  - Verify that the NewOrderInfo schema is presented as output

4. Add a log activity to indicate that a new order request was received.

- Add a **Log** activity between *placeOrderOpIn* and *placeOrderOpOut*



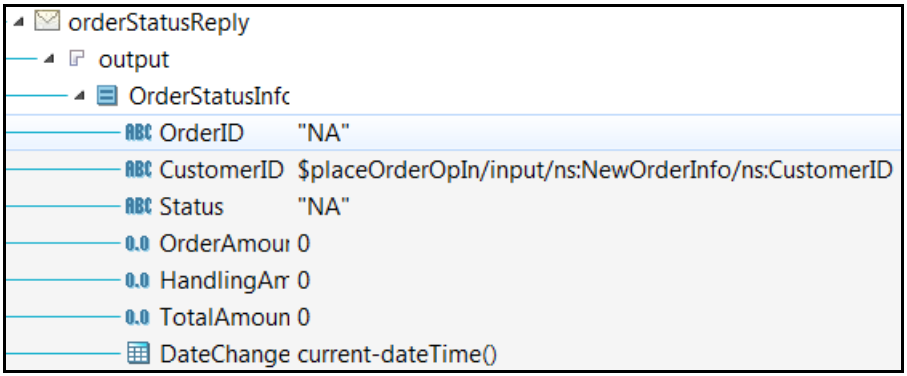
- In *Input* tab, add a message as shown below



- **Save** changes

5. Provide hard coded values for output returned by this process.

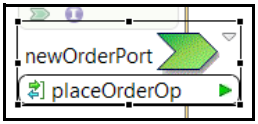
- Update the input to **placeOrderOpOut** as shown below:



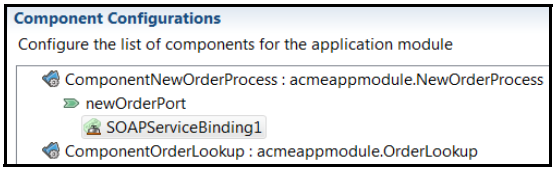
- **Save** changes and all process related errors should be resolved

6. Add a SOAP binding.

- In your process diagram, click on the **newOrderPort** to select it as shown



- Right-click and pick *Components* > *ComponentNewOrderProcess* > ***Create SOAP Binding***
- A new binding is created for this port and displayed



- Click on **SOAPServiceBinding1**
- In *Properties* > ***Binding***, expand **Transport Configuration**
  - Transport Type: **HTTP** (select from list)
  - For the HTTP Connector name, click the browse button and click **Create Shared Resource**
  - Accept the default name and **Finish**
  - **Save** changes
- Confirm that there are no problems (errors or warnings) in your project

7. Generate the concrete WSDL for the NewOrder Service.

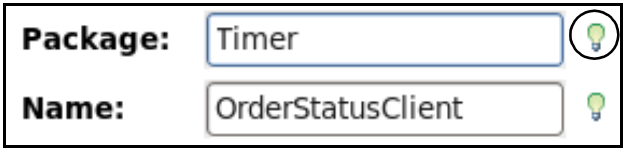
- Click the Generate Concrete WSDL link (above Transport Type)
  - Location: `/TimerModule/Service Descriptors`
  - File Name: **ACMENewOrder.wsdl**, click **Next** Ajustar el nombre si es diferente.
  - Embed Abstract WSDL and Embed Schema **checked**
- Click Advanced and then update Namespace URI: **`http://www.tibco.com/tibedu/ACMENewOrder`**
- Change the Port to **`9797`** and click **Finish**
- The newly generated WSDL file is opened. Examine it and close the view

8. Modify the SOAP binding to use the HTTP port defined earlier.

- In *Project Explorer,* open *ACMEAppModule* > *Module Descriptors* > ***Components***
  - Navigate to *ComponentNewOrderProcess* > ***newOrderPort***
  - Click on **SOAPServiceBinding1**
- In *Properties* **>** ***Binding***, expand **Transport Configuration**
  - Transport Type: **HTTP** (select from list)
  - For the HTTP Connector name, click the browse button and select **HTTPLookup**
  - **Save** changes
- Confirm that there are no problems (errors or warnings) in your project

9. Update the timer module process to reflect that it can test one of two services.

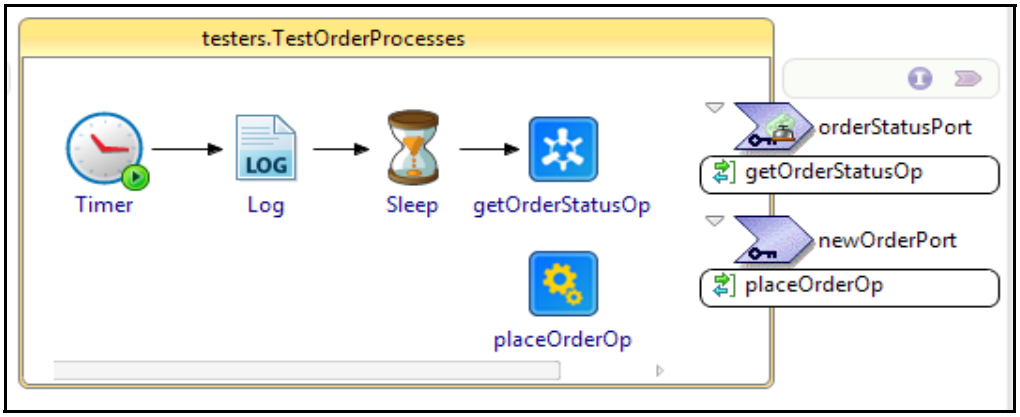
- Browse to *TimerModule* > *Processes* > ***Timer***
  - Double-click the **OrderStatusClient.bwp** to open it
  - Click on the process border (yellow outline) to view Process properties



  - Now, click on the yellow bulb icon next to Package (circled)
  - Rename package to **`testers`**, this may close the process definition view
  - Open process again and click on yellow bulb icon next to **Name**
  - Rename the process to **`TestOrderProcesses`**
- Click **OK** and **Save your changes**

10. Add an activity in the tester process to invoke NewOrderProcess.

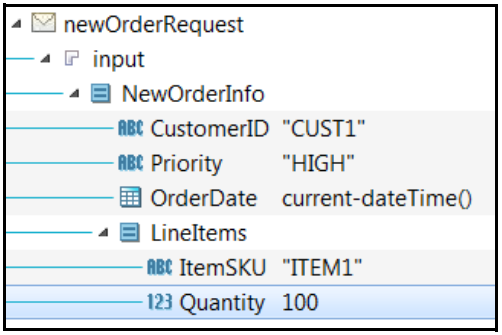
- Confirm that the **TestOrderProcesses** is open
- From Project Explorer, drag **ACMENewOrder.wsdl** onto **TestOrderProcesses**
  - Choose *Invoke Operation* , and a reference is added as shown



  - **Save** changes

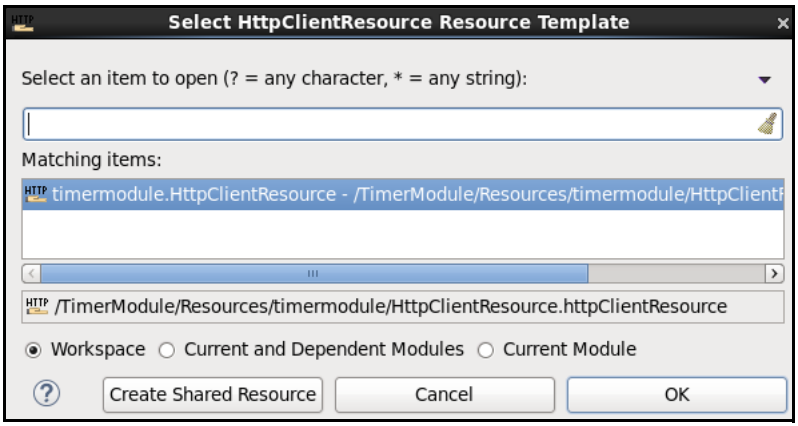
11. Configure the input to placeOrderOp activity.

- Click on **placeOrderOp** activity, and select *Properties* > ***Input***
  - Provide values to the fields as shown below:



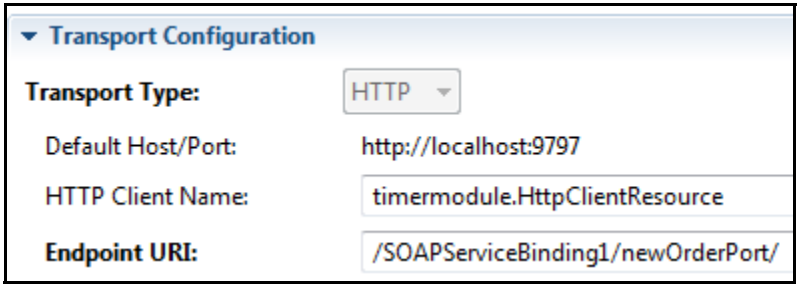
12. Update the reference to make it a binding reference.

- Click on the reference chevron for **newOrderPort**
- In *Properties* > ***General***, change Reference Type to *Binding Reference*
- A new *Binding* tab appears
- In *Binding* Tab:
  - Add a new SOAP Binding Click en OK, se cierra la ventana.
  - For HTTP Client name, select the resource created earlier and click **OK**



En la propiedad HTTP Client Name del Binding hacer click en el botoncito browse (3 puntitos) y despliega esta ventana.

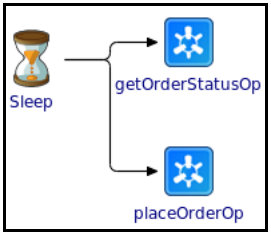
  - Verify your Transport Configuration



  - **Save** changes and close this view

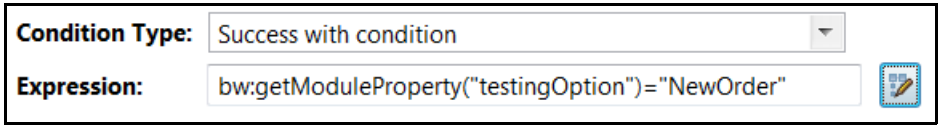
13. Update the Tester process to invoke the placeOrderOp operation of the newOrderPort interface.

- Connect the **Sleep** activity to **placeOrderOp** activity



14. Make a conditional transition between Sleep and placeOrderOp.

- Click on the transition between Sleep and **placeOrderOp**
- In *Properties* > ***General*** tab
  - Label: **`CallNewOrderOp`**
  - Condition Type = **Success with condition**
  - Expression (Click the button to open the **XPath Expression** editor)
  - **`bw:getModuleProperty("testingOption")="NewOrder"`**



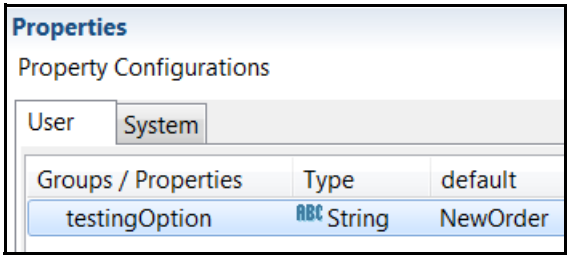
- Verify that there are no mapping errors and **save**

15. Update the transition to getOrderStatusOp as the default transition.

- Click on the transition between Sleep and getOrderStatusOp
  - Label: **`CallOrderStatusOp`**
  - Condition Type: **Success with no matching condition**
- **Save** your changes, and confirm that there are no problems

16. Update the module property to run New Order Process.

- In *Project Explorer*
  - Open (double-click) *TimerModule* > *Module Descriptors* > ***Module Properties***
  - Change the value of testingOption to **NewOrder**



  - **Save** your changes

Tip: You have updated the tester process to invoke the New Order process or Order Status process, based on the value set for the module property **testingOption.**

17. Launch the BusinessWorks tester and confirm that New Order process is executed.

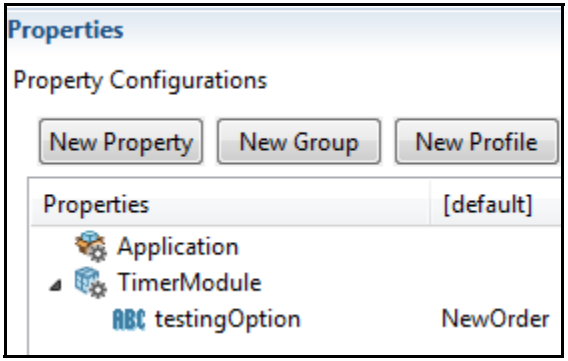
- Create a new configuration named **RunACME** which includes **ACMEApp** and **Timer.app**
  - Run BusinessWorks applications using *RunACME*
  - Confirm that New Order Process is executed by looking at the log message in console

```
dca-8870-c8a9054cc64b)] com.tibco.thor.frwk.Application - T
dca-8870-c8a9054cc64b)] com.tibco.thor.frwk.Application - T
dca-8870-c8a9054cc64b)] com.tibco.thor.frwk.Application - T
els/sharedresource/httpconnector}HttpConnectorConfiguration
-  Received New Order Message for Customer ID: CUST1
```

- The output: "Received New Order Message " indicates that New Order process was called
- **Stop** the engine when you are done

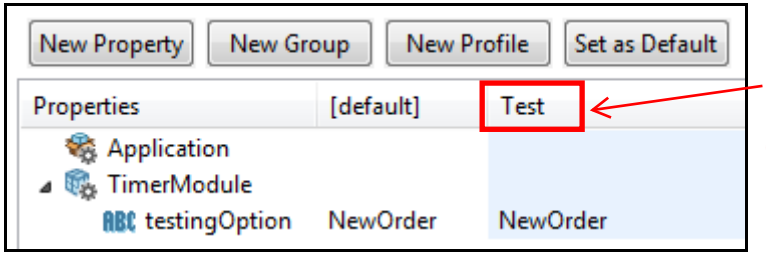
18. Locate the properties section for the Timer application.

- In Project Explorer, browse to *Timer.app* > *Package Unit* > ***Properties***
  - Notice that the properties for all application modules are visible here
  - All properties have values under **default** profile



19. Create a new profile named Test and define new values for the profile.

- Click on the **New Profile** button, and name the Profile as **`Test`**



Debido a la resolución de la pantalla, esta columna podría no verse. Moverse hacia la derecha para encontrarla.

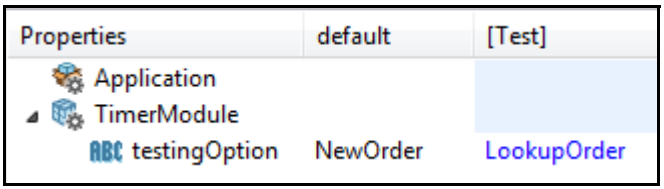
- Notice that the value for the property is copied over to the new profile

Note: You may need to scroll to the right to see the new property

- **Save** your changes

20. Update the property value in the new profile and set it as default profile.

- Choose **testingOption,** change its value in Test profile to **`LookupOrder`**
- Click on the **Test** column and then click **Set as Default**
  - Click **Yes** to set it as default
- Test profile is now noted as **[Test]**



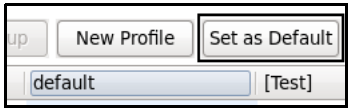
- Save the changes and keep this view open

21. Run the BusinessWorks application and confirm that Lookup Process is executed.

- Select *Run* > ***Run Configurations..*** Me funcionó deteniendo la ejecución, ajustando el nuevo default y volver a correr la configuración RunACME. Cuando hice cambios al profile, la ejecución no tomaba el cambio.
  - Run BusinessWorks applications using *RunACME*
  - The BusinessWorks engine starts and indicates that Lookup order has been called

```
 Started by BusinessStudio, ignoring .enabled settings.
 Started by BusinessStudio, ignoring .enabled settings.
dule.Log - ******* Started Timer Application at: 10:11:50 PM
Module.Log -  Received Lookup Request for OrderID: TEST1
```

- In the Timer.app Properties view, switch default profile back to default
  - Click on **default** and click **Set as Default**



  - **Save** your changes
  - In console window verify that the application is stopped and redeployed
  - Now verify that the **New Order** process is executed

Tip: To switch profiles at development time, you must set the default profile.

  - **Stop** the engine

- Update the property value in the new profile and set it back as **Test** profile
  - **Exit** Business Studio

Lab E - Exercise 3: Types of Properties

Overview

In this exercise import a SOAP Service that uses variables and process variables. Make the required changes to resolve any design time errors.

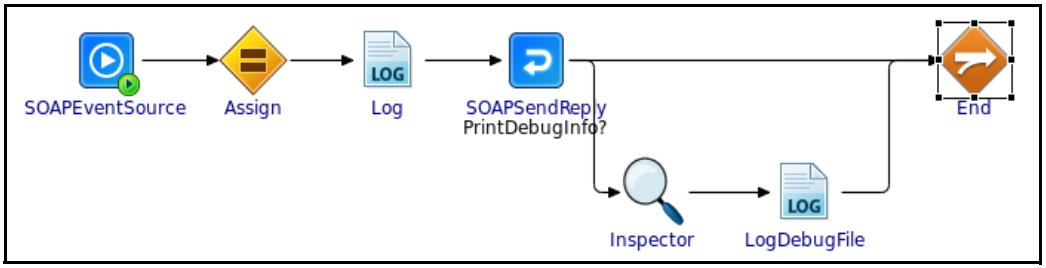
Steps

1. Import the ACMEShipping project into your workspace.

- Open Business Studio with the Workspace `/home/tibco/bw6edu/workspace`
- From the top menu, choose *File* > ***Import...***
  - Choose *General* > *Existing Studio Projects into Workspace,* click **Next**
  - Choose **Select Archive File,** and browse and select `/home/tibco/bw6edu/dev/others/ACMEShippingv1.zip`
- Confirm both projects are selected and click **Finish**

2. Examine the process in the ACMEShipping project.

- Examine *Processes* > *acmeshippingv1* > > ***ACMEShipping***

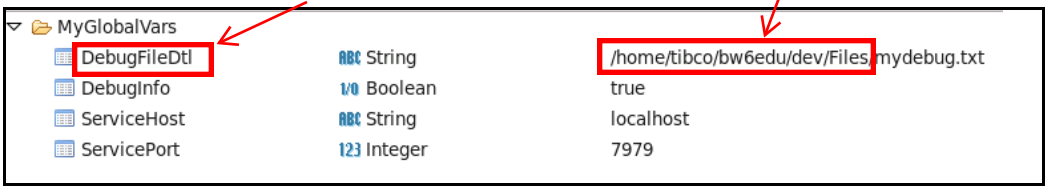


  - Examine the configuration, input and output parameters of all activities
  - Notice the Inspector activity outputs data to file when **DebugInfo**= **true**
  - Also, confirm that there is a process variable named **SpecialIDSeries** (Assign activity)

3. Examine module properties.

- Examine module properties by browsing to ***Module Descriptors > Module Properties***
  - Confirm that Module properties are present, with their values
  - **Update** DebugFileDtl as shown below

Validar que estos directorios existan, si no, crearlos.

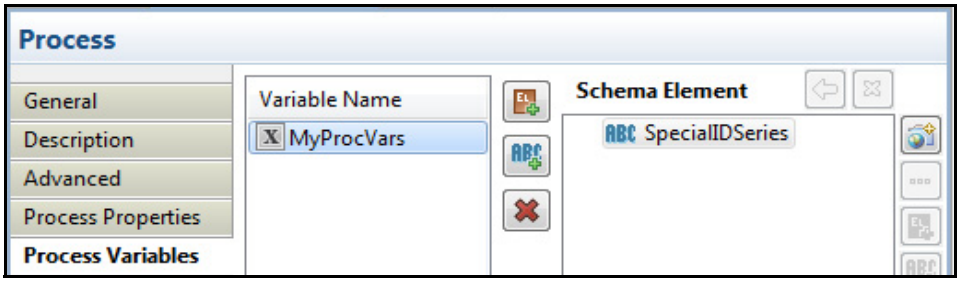


- Browse to and open *ACMEShippingv1/Resources/**HTTPConn.httpConnResource***
  - Verify that the settings look as shown below

4. Examine the details for the process variable.

ACMEShippingv1 > Processes > acmeshippingv1 > ACMEShipping

- Click anywhere on the blank space in the process
  - Now in *Properties*, click on **Process Variables**
  - Confirm that *MyProcVars* > ***SpecialIDSeries*** is available

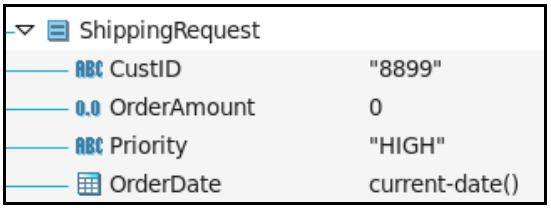


5. Define a calling process to test the service

- Right click on Service chevron in the process
  - Select **BusinessWorks Process > Create Calling Process**
  - Enter Process Name: **ShippingTester**
  - Click **Finish**
- **Save** the changes

6. Complete the calling process design by adding a Timer, and providing data

- Add a Timer activity
  - Connect *Timer* to the *DoShippingOp*
- Complete the data input to be provided to **DoShippingOp**



- One error remains in the reference definition
  - Hover over the Reference chevron to see error details
- Fix the error:
  - Click anywhere on the process diagram for ShippingTester (not on an activity)
  - Now in Properties > Advanced, Set Activation to **Single AppNode**

  - **Save** changes and verify there are no more errors

7. Verify the BusinessWorks application details and run it.

- Select the application **ACMEShippingv1.application**
  - In *Package Unit* > ***Includes*** verify ACMEShippingv1(1.0.0.qualifier) module is included
  - If not, add it, save changes
  - Close the application details window
- In *Run* > ***Debug Configurations***, define a new BusinessWorks Application configuration
  - Name: **ACMEShipping**
  - Applications:ACMEShippingv1.application (includes both processes)
- Now click **Debug,** and verify that application is started successfully
  - Verify that the SOAP Service is also executed

```
Started BW Application [ACMEShippingv1.application:1.0]
.Log - Received Shipping  Request for CustiID: 8899. JobId [bw0a101], ProcessInstanceId [bw0a101],
 Detailed Debug  Info written to /home/tibco/bw6edu/dev/Files/mydebug.txt. JobId [bw0a101], Proces
```

8. Examine the debug output generated by the engine.

- Observe the console window
- Stop the debugger when done testing
  - Shut down TIBCO Business Studio

Notes

Unit 6 Using Java Message Service

This unit introduces messaging concepts and discusses the features of TIBCO Enterprise Message Service.

Topics

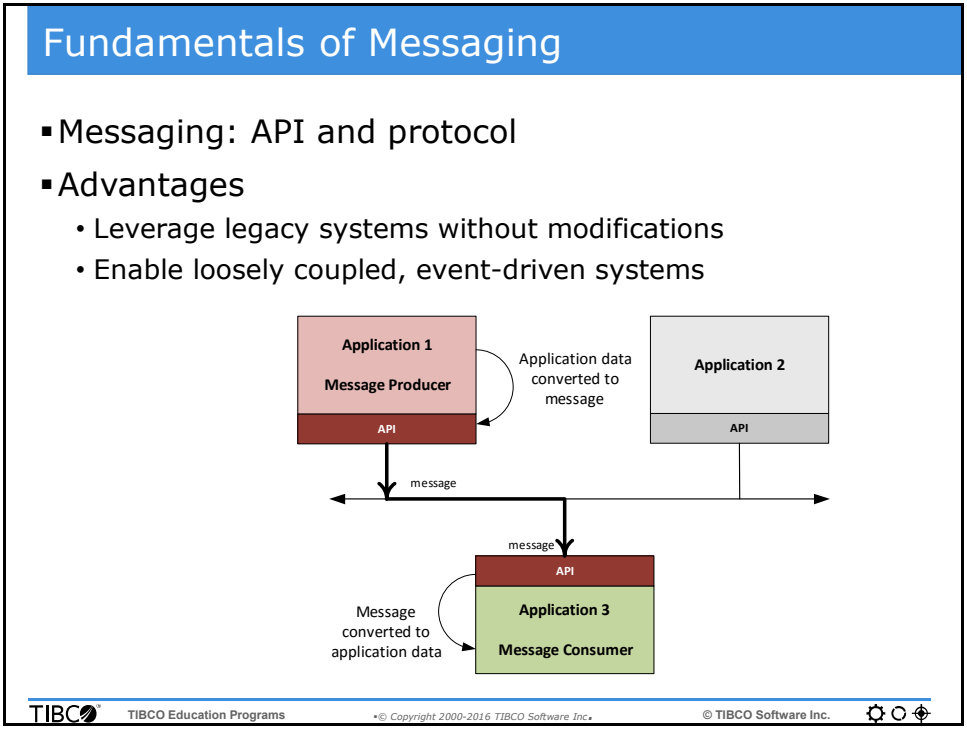
- *Objectives,* page 178
- *Fundamentals of Messaging,* page 179
- *Messaging Features,* page 180
- *Message Processing Choices,* page 181
- *Java Message Service (JMS),* page 182
- *TIBCO Enterprise Message Service,* page 183
- *Publish-Subscribe Messaging,* page 184
- *Point-to-Point Messaging,* page 185
- *JMS Message Format,* page 186
- *Client-Service Delivery Modes,* page 187
- *TIBCO EMS Access Control,* page 188
- *Administering TIBCO EMS,* page 189
- *JMS In BusinessWorks,* page 190
- *Java Projects in Business Studio,* page 191
- *Review,* page 192
- *Lab Overview,* page 193

Objectives

- Knowledge
  - Articulate the role of messaging in integration
  - Explain TIBCO Enterprise Message Service
    - Features and advantages
  - Examine messaging in ActiveMatrix BusinessWorks
- Skills
  - Configure Enterprise Message Service
  - Communicate using JMS messaging

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Objectives



Fundamentals of Messaging

Messaging involves the electronic exchange of data (e.g. events, requests, and replies) among computer applications. Two components form a messaging layer - an API which produces and consumes messages, and a transport that sends the messages between these API clients.

Messaging allows programs to share common message-handling code, to isolate resources and interdependencies, and to easily handle an increase in message volume. Messaging also makes it easier for programs to communicate across different programming environments (languages, compilers, and operating systems) since the only requirement common among environments is the common messaging format and protocol.

Message orientated middleware (MOM) provides a layer between the application and the physical underlying network.

Messaging Features

- Standards-based
  - Interoperability among vendors
  - XML, JMS etc.
- Enables:
  - Decoupled components
  - Time independence
  - Location transparency
  - Scalability
- Qualities of service (QoS)
  - Persistent
  - Non-persistent
  - Transactional

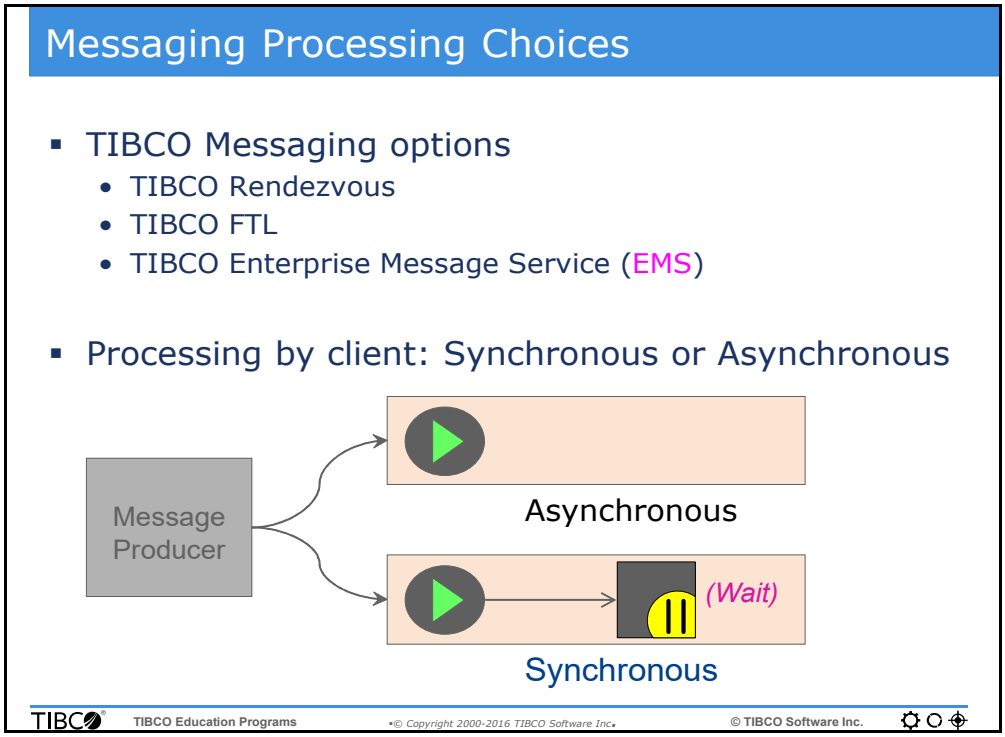
TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Messaging Features

Self-describing messages mean that producers and consumers do not have to have intimate knowledge of the message structure in order to be able to use it. Addressing a message to a subject, queue, or topic decouple the producers and consumers.

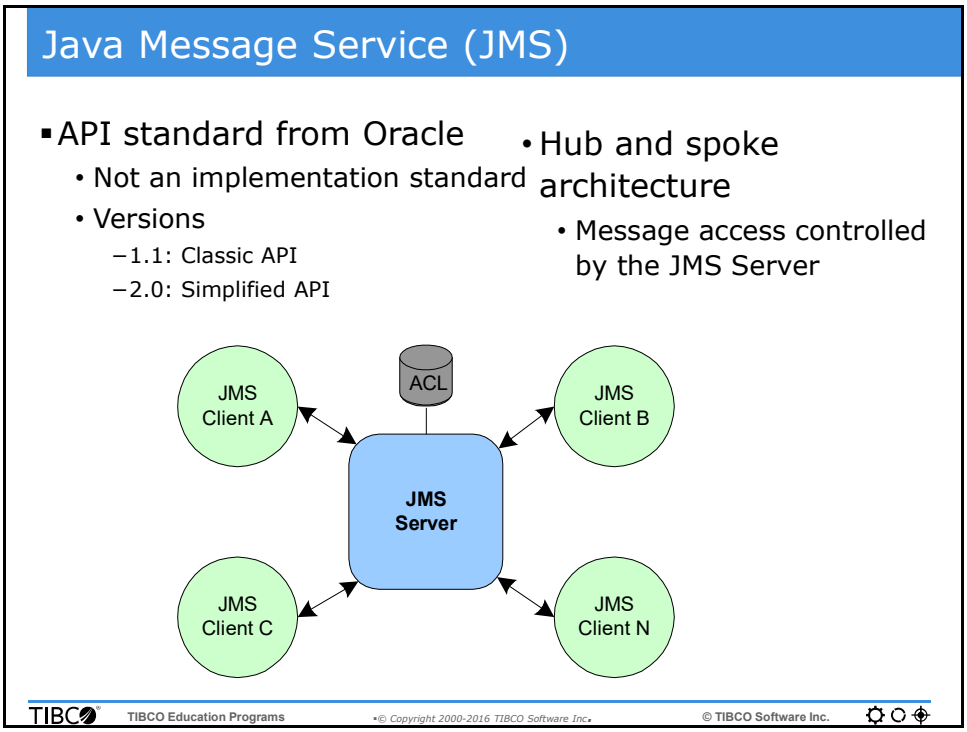
QoS can be categorized as:

- **At least once service (Guaranteed)**: consumer need not be available at the time the message is being produced. Message is kept in persistent storage.
- **At most once service (Reliable)**: consumer must be available at the time the message is being produced. Transport mechanism provides for message reliability in real-time.
- **Exactly once service (Transactional)**: Exactly once delivery and rollback mechanism if a transaction can not be committed.



Message Processing Choices

- **Asynchronous** processing requires that application code register a callback method before proceeding with its normal processing. Thereafter, when a message arrives, the registered callback receives control to process the message. Asynchronous processing offers a developer the opportunity to create elaborate multi-threaded dispatching designs.
- **Synchronous** processing requires that application code explicitly call a method to process an incoming message. An explicit call is typically a blocking call which suspends processing until a message becomes available. If no message is available, the period for which the message processing call blocks can be varied by a parameter value. Synchronous processing is typically used in a server whose sole purpose is to wait for and process incoming request messages, and to send back replies to the requestor.



Java Message Service (JMS)

JMS is a standard API for building centrally-managed messaging applications. The standard specifies the API, it does not define how to implement services within the API. The standard does not address standards above the API level, such as specifying the wire format. Interoperability between vendor JMS implementations is not guaranteed.

JMS is a client/server ("hub-and-spoke") architecture: JMS requires a central server component where all produced messages are sent and where all consumers get their messages from. Even though the JMS architecture is hub and spoke, JMS clients communicate in peer-to-peer fashion: producers and consumers don’t know about each other, they only communicate with the server.

TIBCO Enterprise Message Service

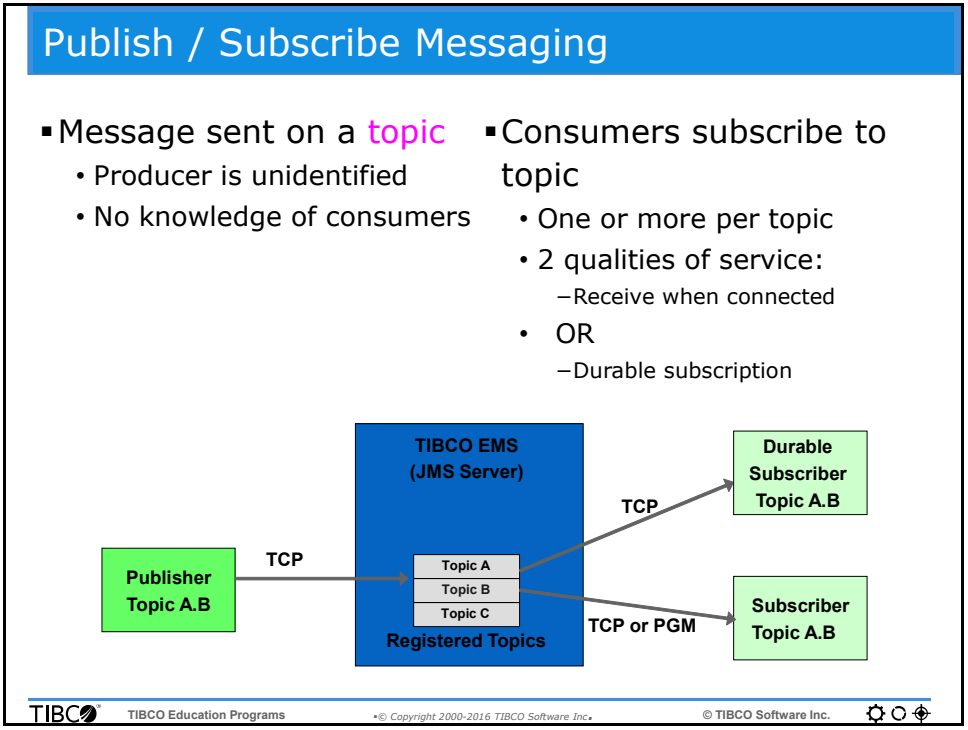
- TIBCO EMS Server 8.x
  - C-based server process for optimum performance
  - Fully compliant with the JMS 1.1 and 2.0 specification
- Features
  - Java and .NET Admin API's
  - Centralized administrator for multiple EMS Servers
  - Bi-directional connectivity with TIBCO Rendezvous
  - Client APIs: C ,Java, C#, COBOL
  - Failover
  - Routing
  - SSL support

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

TIBCO Enterprise Message Service

TIBCO Enterprise Message Service (EMS) is TIBCO's JMS implementation. It provides the required JMS server component and APIs for building JMS clients.

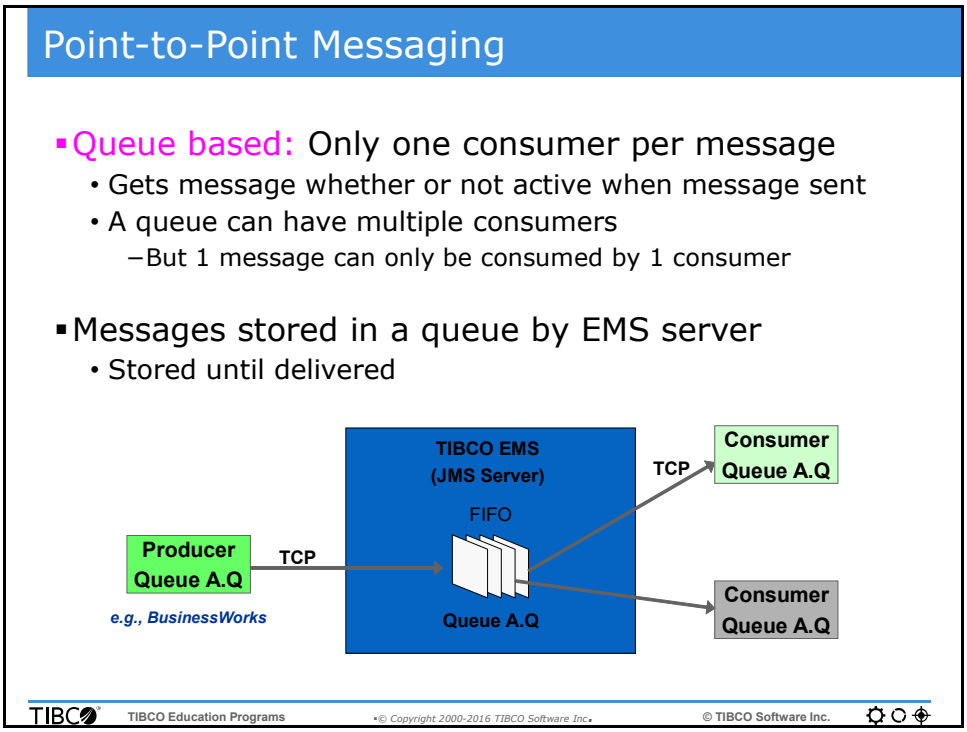
TIBCO EMS provides additional, vendor-specific features, such as bi-directional connectivity with TIBCO Rendezvous, message compression and fault-tolerance.



Publish-Subscribe Messaging

Features of JMS topic-based publish / subscribe include:

- Producers address messages to a topic; a single producer can publish to many consumers.
- Consumers subscribe to topics and receive messages on those topics.
- For standard publish / subscribe messaging, consumers can only receive messages when they are active.
- Topic messages with off-line durable subscribers are retained for guaranteed delivery. They are retained in RAM and if PERSISTENT they are also copied to the file system for recovery from JMS server failures.

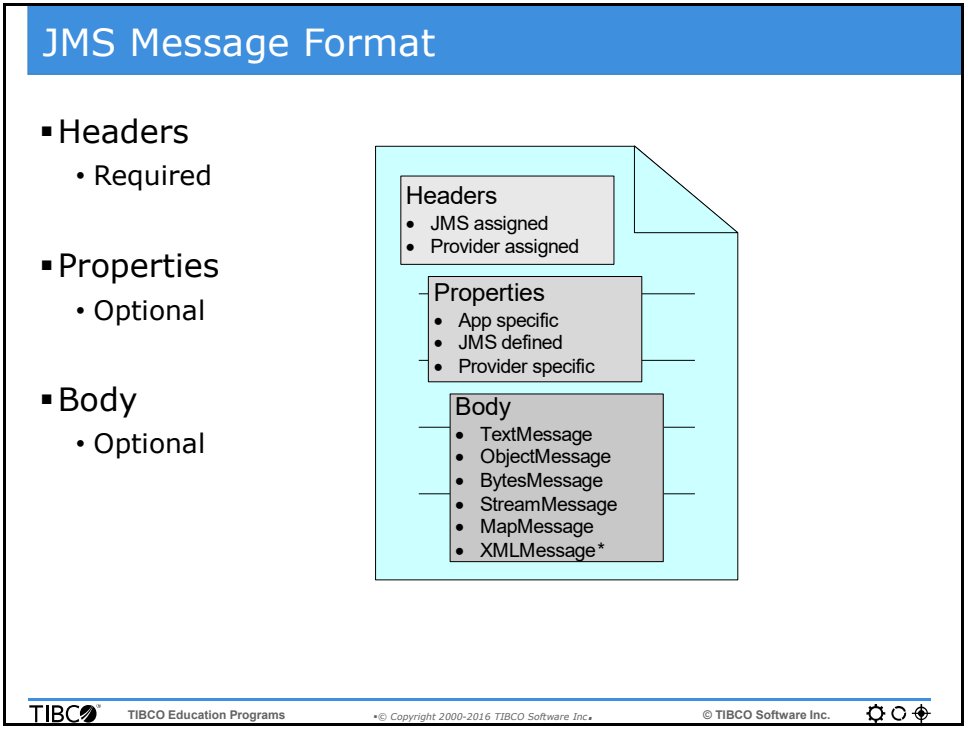


Point-to-Point Messaging

Features of JMS queue-based point-to-point include:

- Messages are point-to-point (one-to-one)
- Only one producer and one consumer per message
- Queue consumer can get the message whether or not it is active at the time the message is sent
- Messages wait in a queue for consumption

Note: Queues can have multiple consumers but each message is only consumed by one of them.

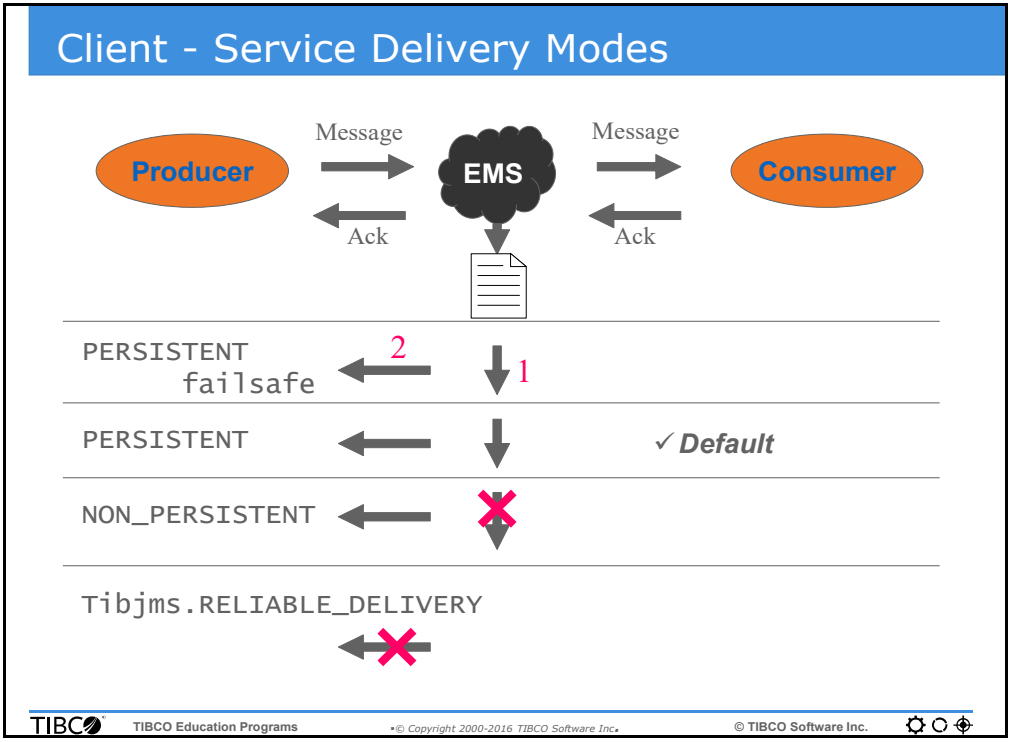


JMS Message Format

The JMS message format is derived from a basic *Message* type that includes *Headers*, which are required for message consumption, and *Properties*, which are user-defined and vendor-specific. The *Body* is optional. The JMS specification proves these *Body* types:

- **TextMessage** – Java language string (e.g. XML)
- **ObjectMessage** – Serializable Java object
- **BytesMessage** – Stream of interpreted bytes
- **StreamMessage** – Stream of primitive Java types
- **MapMessage** – Set of name/value pairs

* The JMS *Message* type is extensible so that vendors can define additional *Body* message types. For example, most vendors, including TIBCO, have an **XMLMessage** body, which is TextMessage.



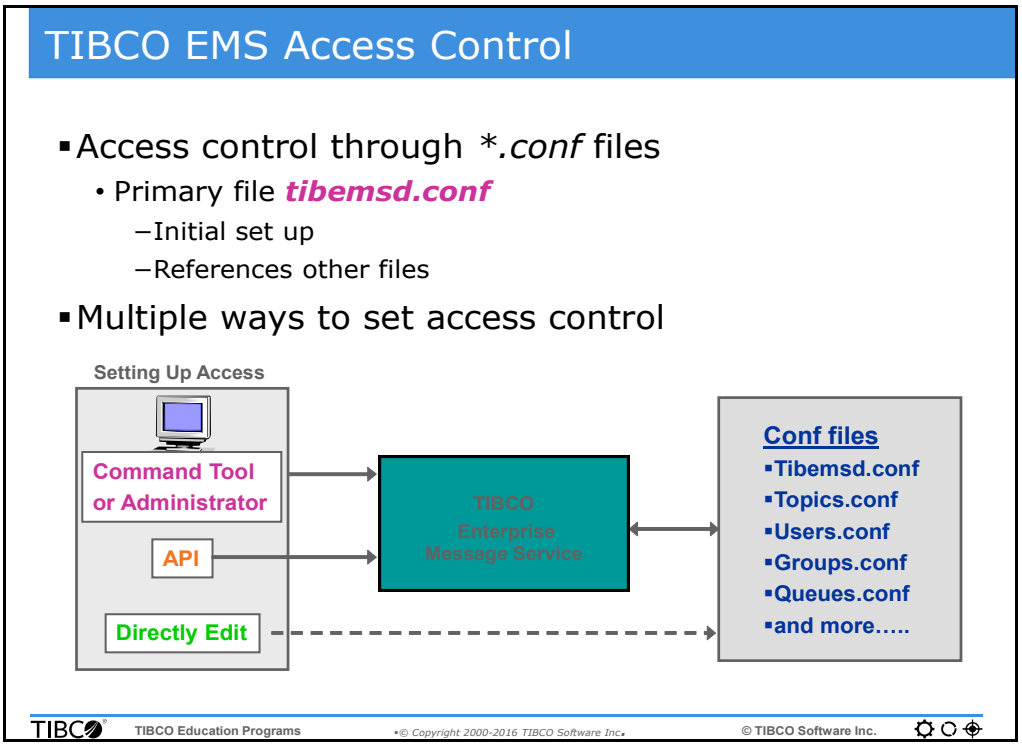
Client-Service Delivery Modes

PERSISTENT failsafe: Message is 1st written to the external store using unbuffered I/O, and after the write completes, the message is acknowledged. There's increased reliability in the case of a JMS failure, but at the cost of low throughput, since acknowledgements are delayed by disk write.

PERSISTENT: (default) Message is written to the external store using ordinary buffered I/O, followed by an acknowledgement to the producing client. Higher throughput than PERSISTENT failsafe, but decreased reliability: if JMS fails, buffered messages not yet been written to disk at the time of the failure will be lost.

NON_PERSISTENT: Message is acknowledged, but not written to an external store. Higher throughput than PERSISTENT delivery mode since disk I/O is eliminated. Further decrease in reliability because a JMS failure destroys all undelivered messages sent with this mode.

Tibjms.RELIABLE_DELIVERY: Completely avoids acknowledgements, resulting in potentially increased throughput, but without the possibility of reporting service trouble to the producing client, via an acknowledgement. Delivery mode is set by the producer's `setDeliveryMode()` method.

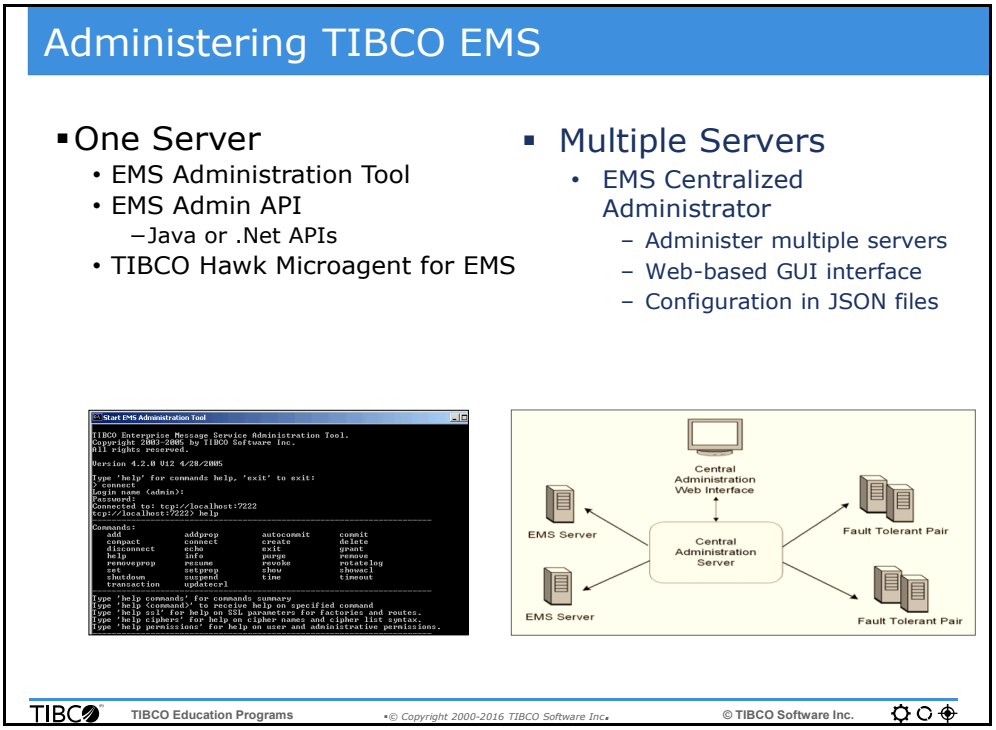


TIBCO EMS Access Control

The ACL (Access Control List) is a security feature for authenticating users and authorizing actions they can perform. The ACL is a local file on the server or can be provided by Directory Services such as LDAP.

Configuring the ACL in TIBCO EMS for message access control can be done four ways:

- Using the Administration tools (either the command line utility that ships with TIBCO EMS or the TIBCO Administrator UI if using JMS with BusinessWorks)
- Using the API
- Or by directly editing the *.conf files



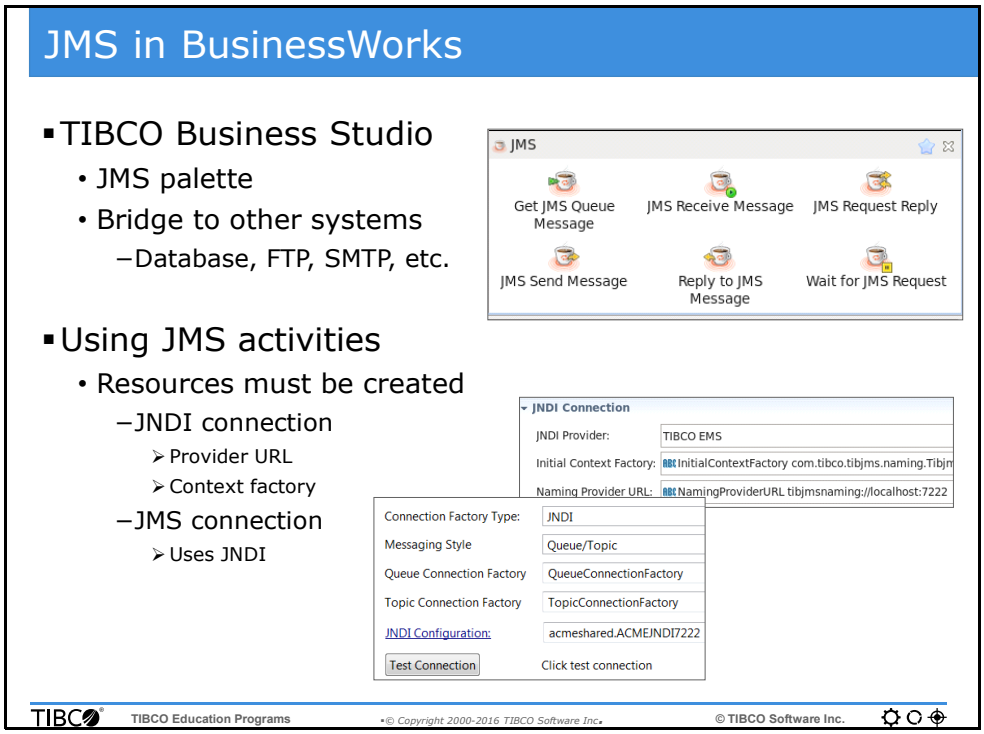
Administering TIBCO EMS

The TIBCO EMS Administration Tool is an easy to use interface for configuring the EMS Server. Using this tool you can:

- Connect to the EMS Server
- Create users and groups and define message access control
- Create message topics and queues
- Shutdown the EMS Server

Alternatively, you can use the web-based TIBCO Administrator GUI for administering the TIBCO EMS Server.

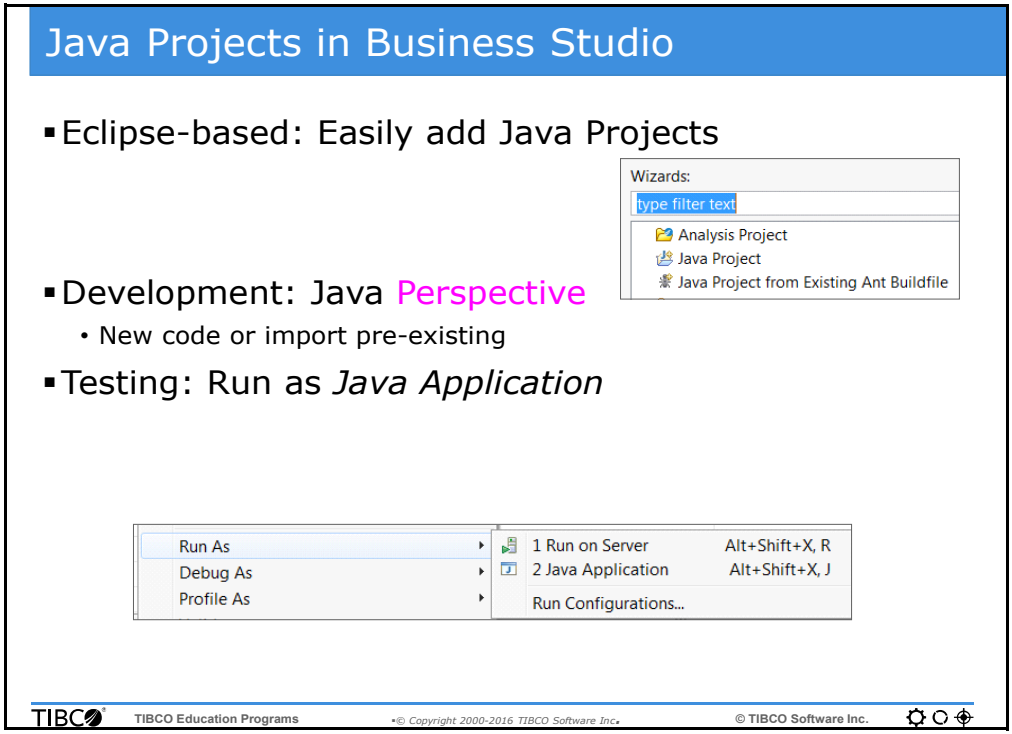
In addition, TIBCO EMS provides Java and .Net APIs for custom administration. And, TIBCO EMS contains an embedded TIBCO Hawk Microagent that can be used to manage and monitor EMS.



JMS In BusinessWorks

The default installation of BusinessWorks 6 includes an JMS palette, which can be used for EMS messaging. The EMS connection is defined as a shared resource, and it provides a connection to the EMS Server.

The Send and Receive message activities are wrappers for the JMS API and provide the functionality without any coding. Simply fill in the properties and start producing or consuming messages.



Java Projects in Business Studio

When using JMS messaging in BusinessWorks, you may frequently find the need to run a Java program which produces or consumes JMS messages.

In such an event, you may update your workspace and add a Java project directly to it. Defining a Java project in the same workspace allows you to store the code along with the other BusinessWorks artifacts.

Since Business Studio is an eclipse-based tool, it provides out-of-the box Java compatibility and common features needed for Java development. When you define a Java project, Business Studio switches to the Java perspective for this reason.

Review

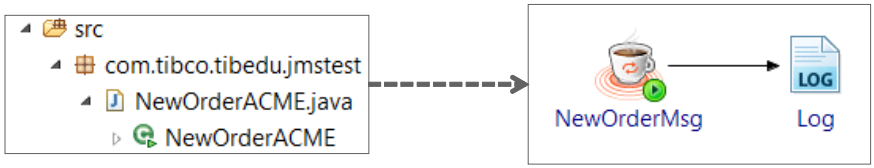
- BusinessWorks supports specification-based messaging
- TIBCO EMS provides JMS messaging
- Administration: Admin Tool or EMSCA
- JMS activities use JNDI, JMS connection
- Java projects can be created in Business Studio

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Develop JMS Test process
- Add Java project to send JMS message
- Test Java project and JMS process



TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Lab Overview

Notes

Lab F JMS Messaging

Overview

In this lab, you will import a project that has processes that send and receive JMS queue messages. You will also build a process that receives a JMS message and returns a response.

Objectives

- *Use JMS Solution, page 196*
- *Build JMS Client,* page 199

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous lab has been completed

Directions

- Complete the exercises that follow

Lab F - Exercise 1: Use JMS Solution

Overview

In this exercise, import a BusinessWorks project which has processes to publish and subscribe to JMS topic messages.

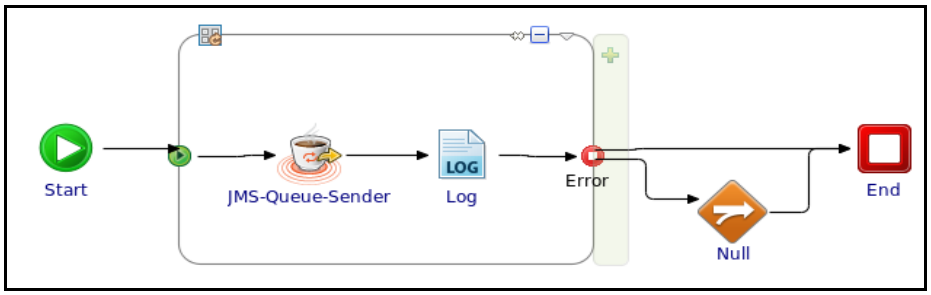
Steps

1. Launch TIBCO Business Studio and import a JMS based project.

- Use shortcut *Applications* > *TIBCO* > *BW6* > ***Business Studio***
  - Workspace: **/home/tibco/bw6edu/workspace**
- From the top menu, choose *File* **> *Import...***
  - Choose *General* > *Existing Studio Projects into Workspace,* click next
  - Choose **Select Archive File,** and browse and select `/home/tibco/bw6edu/dev/others/JMSSendv1.zip`
  - Confirm both projects are selected and click **Finish**

2. Examine the SendOrderRequests process and JMS connection details.

- In Project Explorer, double click on *JMSSend* > *Processes* > ***SendOrderRequests.bwp***
  - Move activities around to make the process more readable
- In Project Explorer, double-click on **SendOrderRequests.bwp** to open it



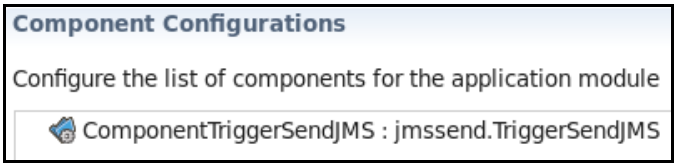
- Notice that this is a sub-process (indicated by green *process boundary*)

3. Examine other artifacts in the project.

- Open the *TriggerSendJMS* process and examine its details
- Browse to **Resources** folder and confirm that:
  - JMS Connection resource is defined
  - JNDI config resources has also been defined (used by JMS connection).

4. Examine process starters for this application module.

- In Project Explorer, browse to *JMSSend* > ***Module Descriptors***
  - Double click **Components,** one starter process is added as shown



5. Examine the application generated by the migration.

- In Project Explorer again, browse to **JMSSend.application**
  - Expand the folder and browse to *Package Unit* > ***Includes***
  - Confirm that the **JMSSend** application module is in the includes list

6. Define a new debug configuration to run the JMS Application only.

- In *Run* > *Debug Configurations..*
- Update the new configuration as follows
  - Name: **`JMSOnly`**
  - Applications: Select **JMSSend** (unselect all other applications)
- **Apply** changes and click **Debug,** confirm that only the JMS application is started
  - Observe that the application is impaired because the EMS server is not running
- **Stop** the AppNode

7. Resolve error with JMS application by starting EMS Server.

- Follow this path: *Applications* > *TIBCO* > *EMS* > ***TIBCO EMS Server - Start***
  - A new window opens and it indicates that the EMS Server is active
- **Minimize** this window.

8. Switch to design perspective and set breakpoint.

- On the top right corner, choose the **Design** perspective
- Open **SendOrderRequests**.bwp
  - Right-click on log activity
  - Choose *BreakPoint* ***> Set Before***
- **S**witch back to **Debug** perspective
- Start the debugger with JMSOnly configuration

9. Complete a job using debugger.

- The jobs wait at the log breakpoint
- Select BusinessWorks Jobs tab and select SendOrderRequests
  - Click **Resume** button (in Debug window, the button with a Green arrow)
- Step through or simply complete the jobs
- **Stop** the debugger when done and **Remove** the breakpoint
- Keep Business Studio open for the next exercise

Tip: In this exercise, the a dynamic queue was created. We will define a static queue to store messages in the next exercise

Lab F - Exercise 2: Build JMS Client

Overview

In this exercise you will build a process that receives a JMS message and returns a response.

Steps

1. Define the destination queue using EMS Administration Tool.

- Launch EMS Administration tool
  (*Applications* > *TIBCO* > *EMS* > ***TIBCO EMS Administration Tool***)
  - Connect to the EMS Server

    `connect (hit enter twice)`
  - Create new Queue

    `create queue ACME.NewOrder`
- Type **quit** to exit the tool

2. Rename the TimerModule and Application to ACMETester, and add a JNDI Configuration.

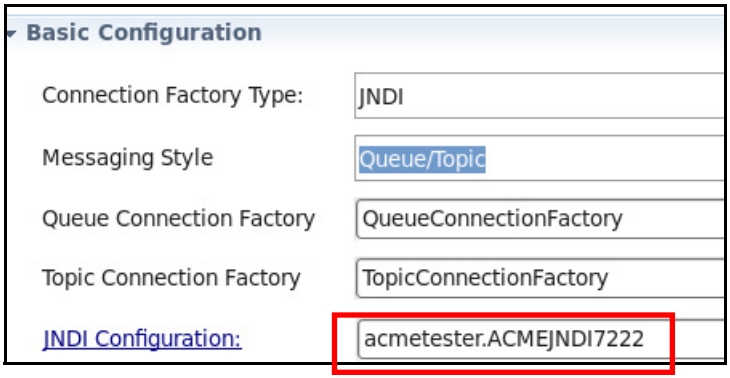
- In Project explorer, right- click on TimerModule and choose **Rename...** to **ACMETester**
- Similarly, rename Timer.app to **ACMETester.app**
- Browse and open *ACMETester* > *Resources*
  - Right-click and add **New > JNDI Configuration**
    - Resource Name: **ACMEJNDI7222**
  - A new package named *acmetester* is being defined here
  - Click **Next**, and verify default settings in the next window

JMS Provider	TIBCO EMS
Initial Context Factory	com.tibco.tibjms.naming.TibjmsInitialContextFactory
Naming URL	tibjmsnaming://localhost:7222

  - Click **Finish**
  - **Close** the JNDI configuration window which is opened

3. Add a JMS connection that uses the JNDI resource you just created.

- Still in *ACMETester> Resources > acmetester*
  - Right-click and choose *New* ***> JMS Connection***
    - Resource Folder: **ACMETester/Resources**
    - Resource Name **ACMEJMS**
    - Click **Finish**
- The JMS configuration view is opened, update it as follows:
  - Messaging Style: **Queue/Topic**
    (Queue/Topic Connection Factory fields appear with default values)
  - JNDI Configuration: **ACMEJNDI7222** (browse to select)



- **Test Connection** and verify success
- **Save** changes
- **Close** the JMS Connection Resource view

4. Define a new process to test the new order service.

- In Project Explorer, browse to *ACMETester* > *Processes* > ***testers***
  - Right-click and select *New* **> *BusinessWorks Process***
  - Name: **`JMSTest`**

Process Folder:	ACMETester/Processes
Package:	testers
Process Name:	JMSTest

  - Accept other defaults and click **Finish**
- **Save** your changes

5. Add a JMS Queue Receiver activity to the process and configure it.

- Right-click in the process and choose *Add Activity* > *JMS* > ***JMS Receive Message***
- In *Properties* > ***General*** tab, update as shown (for other fields, keep their default values):

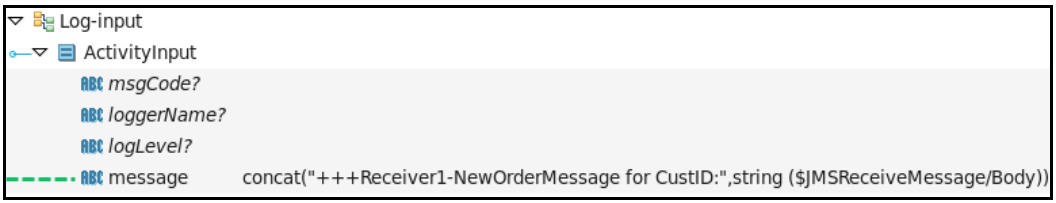
Name:	JMSReceiveMessage
Messaging Style:	Queue
JMS Connection:	JmsConnection acmetester.ACMEJMS
Destination:	ACME.NewOrder

Tipo: Literal Value
Escribir el valor: ACME.NewOrder

6. Add a log activity to record the incoming request.

- Select **JMS Receive Message** activity and click on the + icon that appears below
- Drag the + icon next to **JMS Receive Message** activity
  - In the new menu, choose *General Activities* > ***Log***
  - This adds the activity and connects it as shown

- Configure the **Log** activity, in the input tab, update the message field as shown:

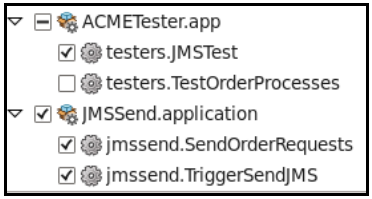


- Verify there are no errors and **Save** changes

7. Run JMSSend and ACMETester applications.

- Define a new debug configuration named ACMETest
  - Select these applications: **ACMETester** and **JMSSend**

  Note: Select only the items shown below



  - **Apply** and **Debug**

Quitar Breakpoints antes de ejecutar el paso 8

8. Send JMS Queue messages using process launcher.

- After the timer fires and the new order messages are shown
- Observe output in the **console** tab

```
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
Log.JMSSend.Log -  Sent Request for  CustID:78787. JobId [bw0a100],
THOR-FRWK-300006: Started BW Application [ACMETester.app:1.0]
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787871
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787872
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787873
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787874
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787875
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787876
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787877
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787878
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:787879
Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID:7878710
```

- When you are done exploring
  - **Stop** the debugger
  - **Shut down** Business Studio

Manda los mesajes combinados y al final estos mensajes de error. IGNORAR

```
[Fatal Error] :1:1: Content is not allowed in prolog.
[Fatal Error] :1:1: Content is not allowed in prolog.
[Fatal Error] :1:1: Content is not allowed in prolog.
[Fatal Error] :1:1: Content is not allowed in prolog.
[Fatal Error] :1:1: Content is not allowed in prolog.
[Fatal Error] :1:1: Content is not allowed in prolog.
[Fatal Error] :1:1: Content is not allowed in prolog.
[Fatal Error] :1:1: Content is not allowed in prolog.
22:05:41.989 ERROR [bwEngThread:In-Memory Process Worker-8] com.tibco.bw.core - TIBCO-BW-CORE-500050: The BW process [testers.TestOrderProcesses] instance faulted, JobId [bw0a102], ProcessInstanceId [bw0a102], ParentProcessInstanceId [-], Module [ACMETester:1.0.0.qualifier], Application [ACMETester.app:1.0].
<CausedBy> TIBCO-BW-CORE-500052: Invoke activity [placeOrderOp] fault.
<CausedBy> com.tibco.bw.core.runtime.api.exception.BWServiceRuntimeException: TIBCO-BW-BINDING-SOAP-555030: Unable to send a request to the service provider for Operation [{http://www.tibco.com/tibedu/ACMENewOrder}placeOrderOp], SoapReferenceBinding [SOAPReferenceBinding], Application [ACMETester.app:1.0] due to exception [com.tibco.bw.sharedresource.http.api.HTTPException]. Message Received: [java.net.ConnectException: Connection refused]
<CausedBy> com.tibco.bw.sharedresource.http.api.HTTPException: java.net.ConnectException: Connection refused
<CausedBy> java.net.ConnectException: Connection refused
```

Notes

Unit 7 Adding Java Projects

This unit describes the java development capability of Business Studio.

Topics

- *Objectives,* page 206
- *Business Studio: Additional Features,* page 207
- *Creating Other Projects in Business Studio,* page 208
- *Creating a Java Project,* page 209
- *Java Perspective,* page 210
- *Adding External JAR Files,* page 211
- *Running Java Projects,* page 212
- *BusinessWorks Projects in Java Perspective,* page 213
- *Review,* page 214
- *Lab Overview,* page 215

Objectives

- Knowledge
  - Examine Java development capabilities of Business Studio
  - Compare Java and BusinessWorks views
  - Explain compile and run options for Java
- Skills
  - Build Java Project

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives

Business Studio: Additional Features

- Business Studio: Eclipse-based tool includes:
  - Java Development Tools
  - Spring IDE
  - CVS/Subversion Support
  - Plug-in Development

Feature Name	Feature Id
Eclipse CVS Client	org.eclipse.cvs
Eclipse FTP and WebDAV Support	org.eclipse.team.extras
Eclipse Java Development Tools	org.eclipse.jdt
Eclipse Platform	org.eclipse.platform
Eclipse Project SDK	org.eclipse.sdk
Eclipse RCP	org.eclipse.rcp
Help System Base	org.eclipse.help
PDE	org.eclipse.pde

- Other features
  - XML / XPath 2 / XSL tools

Feature Name	Feature Id
Eclipse CVS Client	org.eclipse.cvs
Eclipse FTP and WebDAV Support	org.eclipse.team.extras
Eclipse Java Development Tools	org.eclipse.jdt
Eclipse Platform	org.eclipse.platform
Eclipse Project SDK	org.eclipse.sdk
Eclipse RCP	org.eclipse.rcp
Help System Base	org.eclipse.help
PDE	org.eclipse.pde

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Business Studio: Additional Features

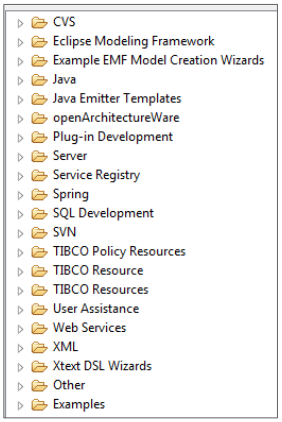
Business Studio is based on Eclipse. Eclipse itself is a fully functional Java IDE (Integrated Development Environment).

In addition to BusinessWorks projects, it can be used to define different kinds of other projects, such as Java Project, Spring Project, Java project from Ant buildfile and others.

In also supports multiple version control systems, such as CVS and Subversion, so developers can check-in, check-out and manage their code.

Creating Other Projects in Business Studio

- Design perspective
  - Choose New > Other…
- Multiple project options for Java:
  - Java project
  - Project from existing Ant buildfile
  - Plug-in project



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Creating Other Projects in Business Studio

By default, TIBCO Business Studio has short cuts to create BusinessWorks applications and application modules.

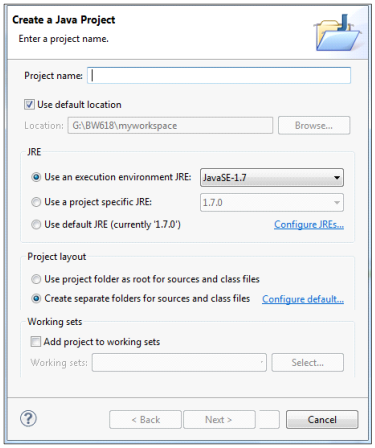
The other projects are however not disabled, and are available in Business Studio. Simply choose New > Other when defining a new project and you can see the various project types supported by Business Studio.

There are multiple options to create Java projects. You can create plain Java projects which can be used for standalone Java applications and applets. You can also define a project from an existing Ant buildfile, which defines all the project objects, and you simply have to fill in the business logic (code).

Finally, you can define a plug-in project. A plug-in project is defined using the plug-in development environment and is used to define plug-ins which can integrate with the Eclipse platform and/or other plug-ins. BusinessWorks components are defined as plug-ins for the Eclipse platform.

Creating a Java Project

- Choose New > Other.. Java Project
- Define
  - Project Name
  - JRE version
  - Project layout
- Java Perspective opened



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Creating a Java Project

The New Java Project wizard can be used to create a new java project.

There are many ways to open this wizard:

- From File menu, choose New > Java Project
- Right click anywhere in the Project Explorer and select New > Java Project
- Clicking New button in the Tool bar and selecting Java Project
  - Using the New Java Project wizard

Page 1: Enter Project Name and select the Java Runtime Environment (JRE) or choose the default.

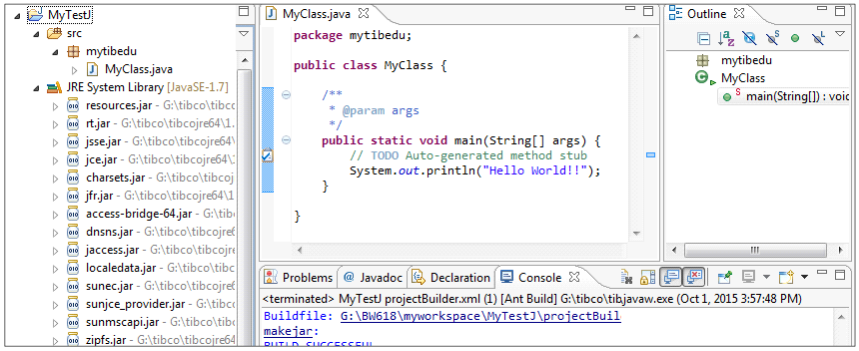
Select Project Layout, whether there would be a separate folder for the sources code and class files. Click Finish button to create the project or click on the Next button to change the java build settings.

Page 2: (optional)

- Change the Java Build Settings like setting the Project dependency
- Add additional jar files to the build path.

Java Perspective

- Class outline view: methods
- System and external libraries



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Java Perspective

Java Perspective: Designed for working with Java projects. It consists of an editor area and the following views:

Package Explorer , Outline , Problems , Javadoc , Declaration.

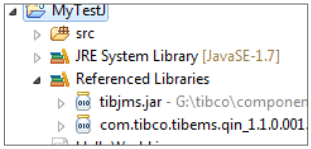
Java Debug : For debugging your Java program. It includes an editor area and the following views.

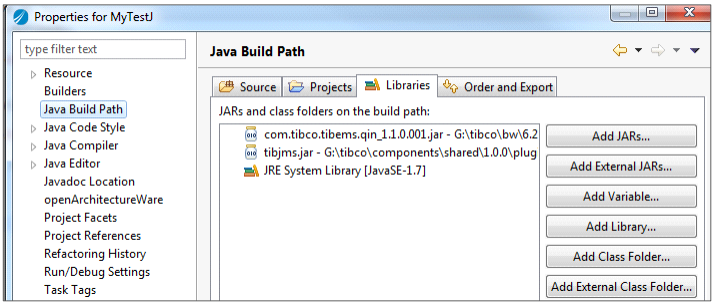
Debug , Breakpoints , Variables , Expressions , Outline , Console and Tasks.

Many view of the Java and Java debug perspectives are similar to the ones presented by BusinessWorks (Design and Debug) perspectives.

Adding External JAR Files

- Configure Java Build Path
  - Add external JARs
  - Confirmation
    - Check *Referenced Libraries*





TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. • © TIBCO Software Inc.

Adding External JAR Files

When your Java project requires JAR libraries (Java Archive) to function, you have to configure your project to include the libraries in its build path. Fortunately, Eclipse makes this process simple and easy to remember.

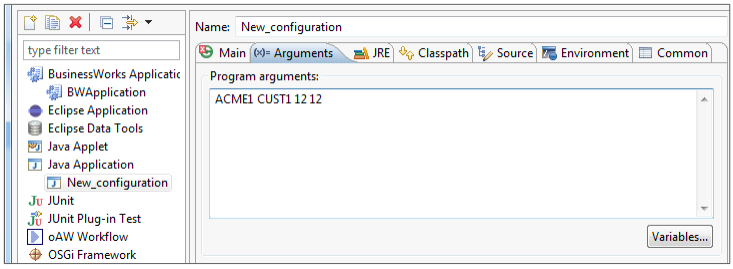
Recommended but optional: Define a LIB folder, and copy your external JARs to the LIB folder. To add a JAR file to the build path, select Build Java Build Path in the Properties of the Project.

There, select the option to add internal or external JAR files. If you have defined a LIB folder in your project, choose Add JARs. If you have not created a LIB folder, choose Add External JARs.

In either case, verify that the JAR file appears in the build path list as shown.

Running Java Projects

- Available: Run and Debug
  - Similar to BusinessWorks applications
- Define *run / debug* configuration
  - Add runtime arguments
  - Debug options

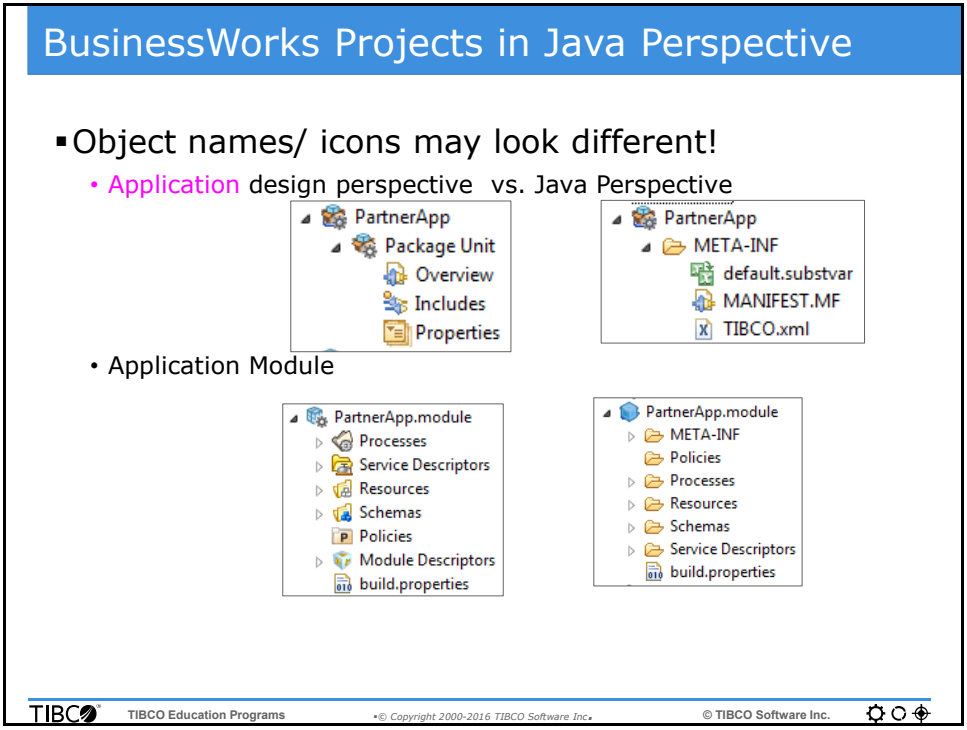


TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Running Java Projects

Run Configurations... / Debug Configurations... : Press the New button in the toolbar to a new launch configuration.

Main: Defines the class to be launched. Check the Stop in main checkbox if you want the program to stop thereArguments : Arguments to be passed to the application/ virtual machine . Also specify the working directory to be used by the launched application.
JRE: JRE version used to run or debug the application. Select from defined JREs, or define a new JRE.
Classpath: location of class files used when running or debugging an application. By default derived from the associated project's build path, override here.
Source: Display source when debugging a Java application. Derived from the associated project's build path, override here. Environment: By default, the environment is inherited from the Eclipse runtime. Override or append here. Variables specified here replace values in the underlying native environment. You can "Append environment to native environment" or "Replace native environment with specified environment". Common : Store the launch configuration in a specific file and specify which perspectives become active when the launch configuration is launched.



BusinessWorks Projects in Java Perspective

When you look at BusinessWorks project through Java perspective, you may notice several changes:

The special folders in BusinessWorks perspective appear as normal folders in the Java perspective. The Application Package Unit appears as a folder named META-INF. The Module Descriptors folder in an Application Module also appears as a META-INF folder.

Astute Java developers will realize that this denotes the Manifest file in a Java development environment. Rest assured that the objects themselves do not change. Simply change the perspectives and you will see the objects as you created them.

Review

- BusinessStudio can be used for developing Java programs
- BusinessWorks objects look different in Java perspective
- Run or Debug configurations to run Java projects

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Review

Lab Overview

- Define Java Project
  - Reference JMS JAR files
  - Copy pre-define code
- Run Java project
  - Run configuration with arguments
  - Send JMS message

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab G Java Projects

Overview

In this lab you will define a Java project and import code to send JMS Queue messages. This will verify JMS connectivity with BusinessWorks.

Objectives

- *Adding a Java Project,* page 218

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous labs have been completed

Directions

- Complete the exercises that follow

Lab G - Exercise 1: Adding a Java Project

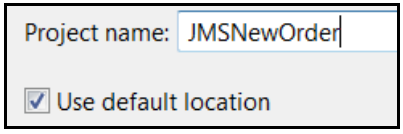
Overview

In this exercise you will define a Java project and import code to send JMS Queue messages.

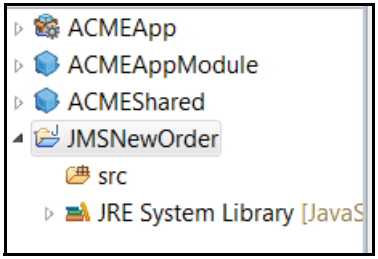
Steps

1. Define new Java project to send JMS messages.

- Launch TIBCO Business Studio with workspace: **/home/tibco/bw6edu/workspace**
- From the Main Menu in Business Studio, choose *File* > *New* **> *Project***
  - From the Wizard, choose **Java Project**
  - Click **Next**
    - Project Name: **`JMSNewOrder`**

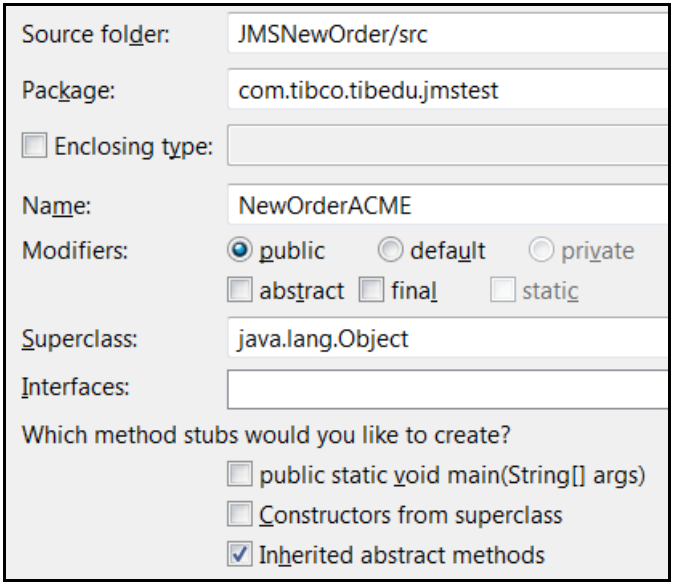


  - Click **Finish**
- You are asked whether to open the Java Perspective
  - Click **NO**
- A new project is defined as shown below



2. Add a new class named NewOrderACME to the Java project.

- Browse to *JMSNewOrder* > ***src*** folder
  - Right-click and select *New* > ***Class***
    - Package: **`com.tibco.tibedu.jmstest`**
    - Name: **`NewOrderACME`**
    - Keep other defaults as shown



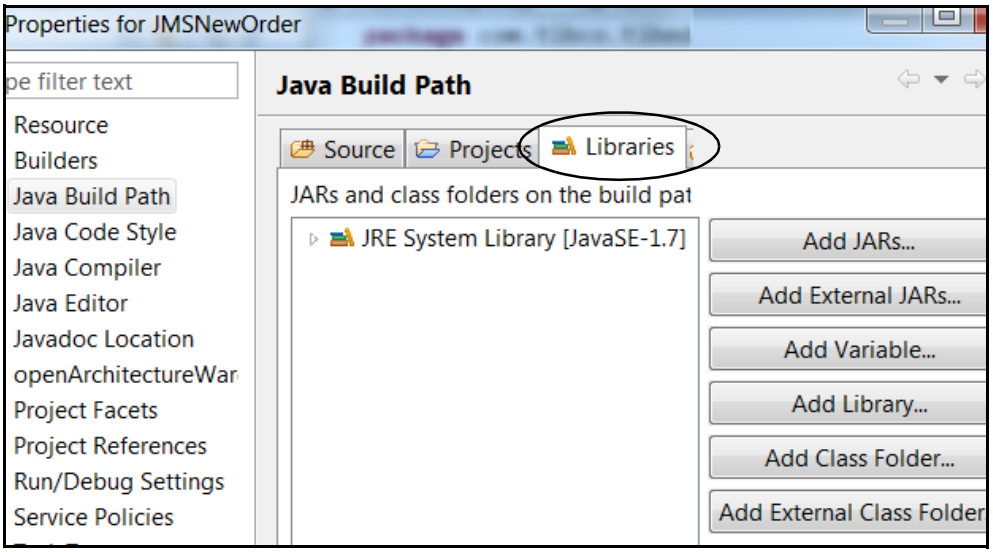
- Click **Finish**
- The new class definition is opened. **Delete** the pre-built lines from this file

3. Copy the pre-built code for the NewOrderACME class.

- In File Explorer (bottom left) screen, browse to the **code** sub-folder
  - Drag the **NewOrderACME.txt** file to the open Java code window, to open it
- **Copy** all the code from NewOrderACME.txt file
- **Paste** the code into **NewOrderACME.java** to completely replace it
- **Save** your changes, and the code displays errors which will be resolved next

4. Update project build path to add JMS jars.

- From the top menu, choose *Project* > ***Properties***
- Choose **Java Build Path**
  - Now click on the **Libraries** Tab



- Click **Add External JARs...** (right side menu)
  - Browse to `/opt/tibco/soa/ems/8.3/lib/`**`jms-2.0.jar`**
  - Similarly, add `/opt/tibco/soa/ems/8.3/lib/`**`tibjms.jar`**
  - Click **OK** and all errors should be resolved

5. Run the java application and identify the required arguments.

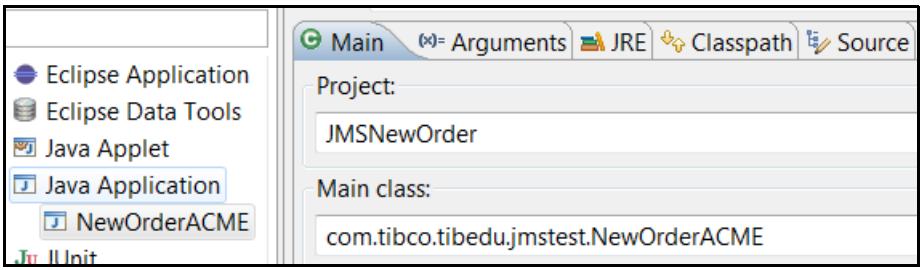
- Right-click on the java code and choose *Run As* > ***Java Application***
- Check output displayed in the console view

```
Usage: java NewOrderACME <CustID> <SKU1> <Qy1> <SKU2> <Qty2>
```

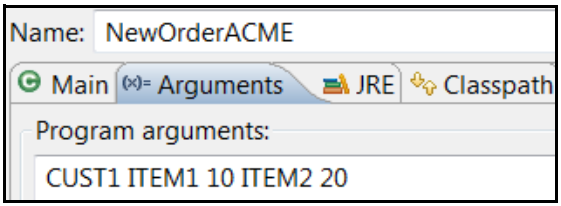
- This confirms that 5 arguments are to be provided

6. Update run configuration to provide arguments to the Java program.

- Select *Run* > ***Run Configurations...***
  - Confirm *Java Application* > ***NewOrderACME*** is selected (if not, select it)



- Choose ***Arguments*** tab, and add the arguments shown below:



  - Click **Apply** and then click **Run**
  - Examine output in the Console window

```
destination is: ACME.NewOrder
Sending TextMessage={ Header={ JMSMessageID={null} JMSDestination={null} JMSRepl
yTo={null} JMSDeliveryMode={PERSISTENT} JMSRedelivered={false} JMSCorrelationID=
{null} JMSType={null} JMSTimestamp={0} JMSDeliveryTime={0} JMSExpiration={0} JMS
Priority={4} } Properties={ } Text={<?xml version='1.0' encoding='UTF-8'?><ns0:N
ewOrderInfo xmlns:ns0='http://www.tibco.com/schemas/testschemas/Schema.xsd'> <ns
0:CustomerID>CUST1</ns0:CustomerID> <ns0:Priority>CUST1</ns0:Priority> <ns0:Orde
rDate>2014-08-08T11:11:12.484-08:00</ns0:OrderDate><ns0:LineItems> <ns0:ItemSKU>
ITEM1</ns0:ItemSKU> <ns0:Quantity>10</ns0:Quantity></ns0:LineItems>ns0:LineItems
> <ns0:ItemSKU>ITEM2</ns0:ItemSKU> <ns0:Quantity>20</ns0:Quantity></ns0:LineItem
s> </ns0:NewOrderInfo>} }
```

  - You may have to scroll to see complete output
- The program terminates after it finishes running

7. Debug the BusinessWorks Application and verify that the message is processed.

- Define a new debug configuration named **JMSTest**
  - Select these applications: **ACMETester** and **JMSTest**

  Note: Select only the items shown below

  - **Apply** and **Debug**
  - Observe the message in the console:

```
Started BW Application [ACMETester.app:1.0]
- +++Receiver1-NewOrderMessage for CustID:<?xml version='1.0'
```

- The application receives the message as a single string
  - We need to update the receiver to process the XML string next

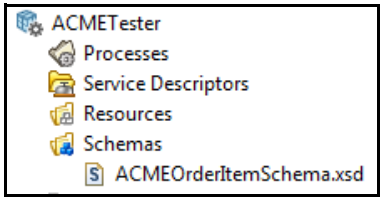
  Note: If you see these messages, it means that there were pending messages from the JMSSend Application. These can be ignored"

```
)21: All Application dependencies are resolved for
Started BW Application [ACMETester.app:1.0]
- +++Receiver1-NewOrderMessage for CustID:787877
- +++Receiver1-NewOrderMessage for CustID:787872
- +++Receiver1-NewOrderMessage for CustID:787874
- +++Receiver1-NewOrderMessage for CustID:787873
- +++Receiver1-NewOrderMessage for CustID:787875
- +++Receiver1-NewOrderMessage for CustID:787876
- +++Receiver1-NewOrderMessage for CustID:787871
- +++Receiver1-NewOrderMessage for CustID:787878
- +++Receiver1-NewOrderMessage for CustID:7878710
- +++Receiver1-NewOrderMessage for CustID:787879
```

  - **Stop** the tester

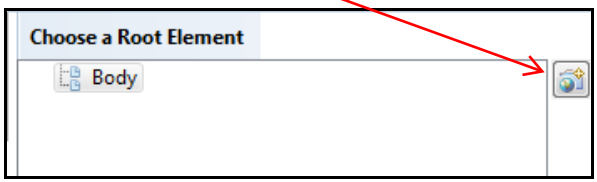
8. Import the schema for the new order into your ACMETester application module.

- In *File Explorer*, expand `/home/tibco/`**`bw6edu/dev/resources`** folder
  - Select **ACMEOrderItemSchema.xsd**
- Drag the file to the Business Studio to *Project Explorer* > *ACMETester* > ***Schemas*** folder



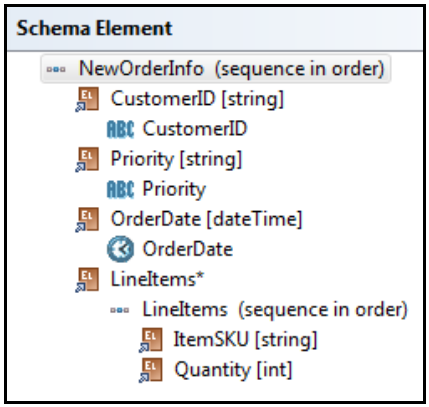
9. Update the JMS-Queue-Receiver activity to use XML text in NewOrderInfo format.

- Open *ACMETester* > *Processes* > *testers* > ***JMSTest* > *JMSReceiveMessage*** activity
- In *Properties* > ***General*** tab,
  - Update Message Type: **XML Text**
- Still in Properties, select the **Output Editor** tab
- Click on **Select Body Element** button



- Locate *ACMETester* > *Schemas* > ***ACMEOrderItemSchema.xsd***

- Choose **NewOrderInfo**, confirm that the details are loaded, **save** changes



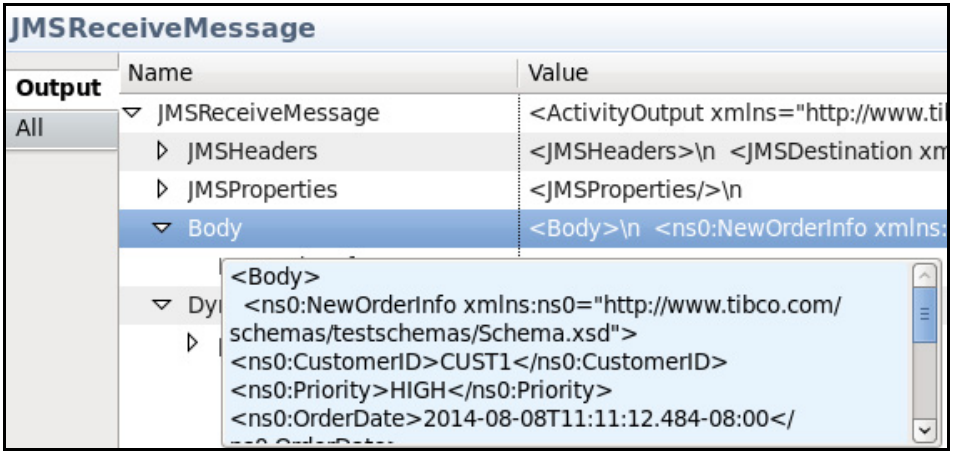
10. Rerun the Java program to send a JMS message again.

- In Project Explorer, open *NewOrderACME.java*
  - Right-click and select *Run As* >***Java Application***

```
<terminated> NewOrderACME [Java Application] G:\tibco\tibcojre64\1.7.0\bin\javaw.exe (Aug 31, 2015 9:33:11 PM)
destination is: ACME.NewOrder
Sending TextMessage={ Header={ JMSMessageID={null} JMSDestination={null} JMSReplyTo={null}
```

11. Restart the BusinessWorks tester and confirm that the message is received as XML.

- In Debug configuration, run the **JMSTest** configuration and observe the results:
- In the Debug window locate the **JMSReceiveMessage** activity
- Examine Job data, confirm that the XML message has been processed



- **Stop** the tester when done
- **Shutdown** Business Studio

Notes

Unit 8 Developing REST Clients

This unit examines the REST client capabilities of BusinessWorks. It examines the fundamentals of JSON (JavaScript Object Notation) and discusses the conversion capabilties between XML and JSON.

Topics

- *Objectives,* page 228
- *Representational State Transfer (REST),* page 229
- *SOAP vs REST,* page 230
- *JavaScript Object Notation (JSON),* page 231
- *REST and JSON Palette Overview,* page 232
- *Invoke REST API,* page 233
- *Parse JSON and Render JSON,* page 234
- *Conversion between XML and JSON,* page 235
- *Normal Conversion,* page 236
- *Badgerfish Conversion,* page 237
- *Business Case Implementation,* page 238
- *Review,* page 239
- *Lab Overview,* page 240

Objectives

- Describe procedure for developing REST clients
- Explain functionality of Invoke API activity
- Discuss XML to JSON conversion options
- Examine benefits of using Badgerfish

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives

Representational State Transfer (REST)

- REST: Not a "standard"
- For designing networked applications
- Communications protocol normally HTTP
  - CRUD (Create/Read/Update/Delete)
- Operation request:
  - Transfers all state information needed by server

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Representational State Transfer (REST)

Representational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant Web service design model.

REST supports CRUD operations, and it uses the common HTTP methods for this purpose.

GET instructs the server to transmit the data identified by the URL to the client. Data should never be modified on the server side as a result of a GET request so a GET request is read-only.

PUT request is used when you wish to create or update the resource identified by the URL.

Delete: Delete resource identified by URL of the request.

POST: Generally used to create new data on the server.

ActiveMatrix BusinessWorks currently allows the following HTTP operations to be performed on resources: GET, PUT, DELETE, and POST. Both XML and JSON are supported as data serialization formats along with support for definition of custom status codes, key-value parameters, and query parameters.

SOAP vs REST

- SOAP Request

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

- REST Request

```
http://www.acme.com/phonebook/UserDetails/12345
```

- Typical REST response formats
  - XML
  - CSV
  - JSON

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

SOAP vs REST

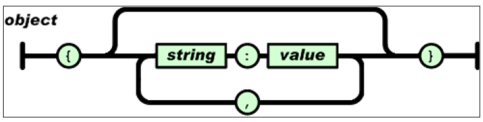
SOAP relies exclusively on XML to provide messaging services.

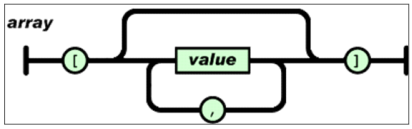
REST provides a lighter weight alternative. Instead of using XML to make a request, REST relies on a simple URL in many cases. In some situations you must provide additional information in special ways, but most Web services using REST rely exclusively on obtaining the needed information using the URL approach. Unlike SOAP, REST doesn't have to use XML to provide the response. You can find REST-based Web services that output the data in Comma Separated Value (CSV), JavaScript Object Notation (JSON) and Really Simple Syndication (RSS).

JavaScript Object Notation (JSON)

- Open standard
- Alternative to XML
  - Increased readability
    - Attribute-value pairs
    - Array list of values
  - Easier to parse in JavaScript clients
- Transmits data between server and web application

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    }
}
```





TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

JavaScript Object Notation (JSON)

To send a JSON response, it is important to understand the basics of this format.

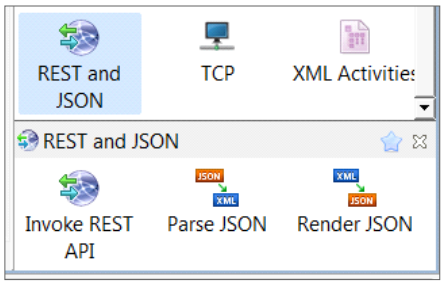
JavaScript Object Notation: text-based open standard designed for human-readable data interchange. Derived from JavaScript scripting language, it has simple data structures and associative arrays, called objects. It is language-independent, with parsers available for many languages. JSON defines a small set of formatting rules for the portable representation of structured data.

JSON data is human readable, and it has name value pairs, enclosed in quotes. Name and value are separated by a colon.

This format is described in RFC 4627.

REST and JSON Palette Overview

- Activities
  - Invoke REST API
  - Parse JSON
  - Render JSON



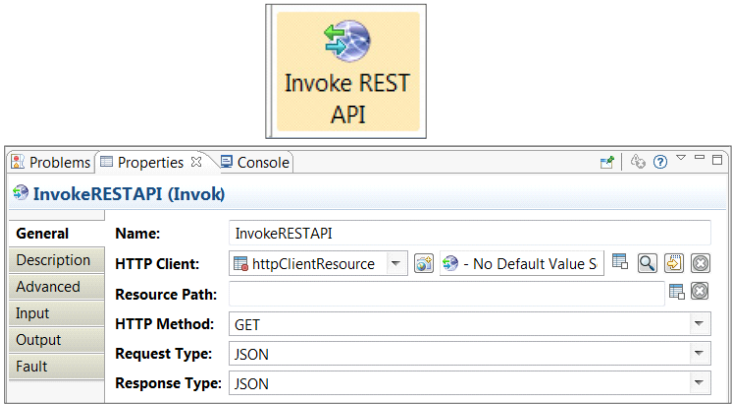
TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

REST and JSON Palette Overview

The activities of REST and JSON Palette can be used to invoke RESTful web services and expose BusinessWorks processes as RESTful web services. The activities in this group can also be used to convert data between JSON format and XML format.

Invoke REST API

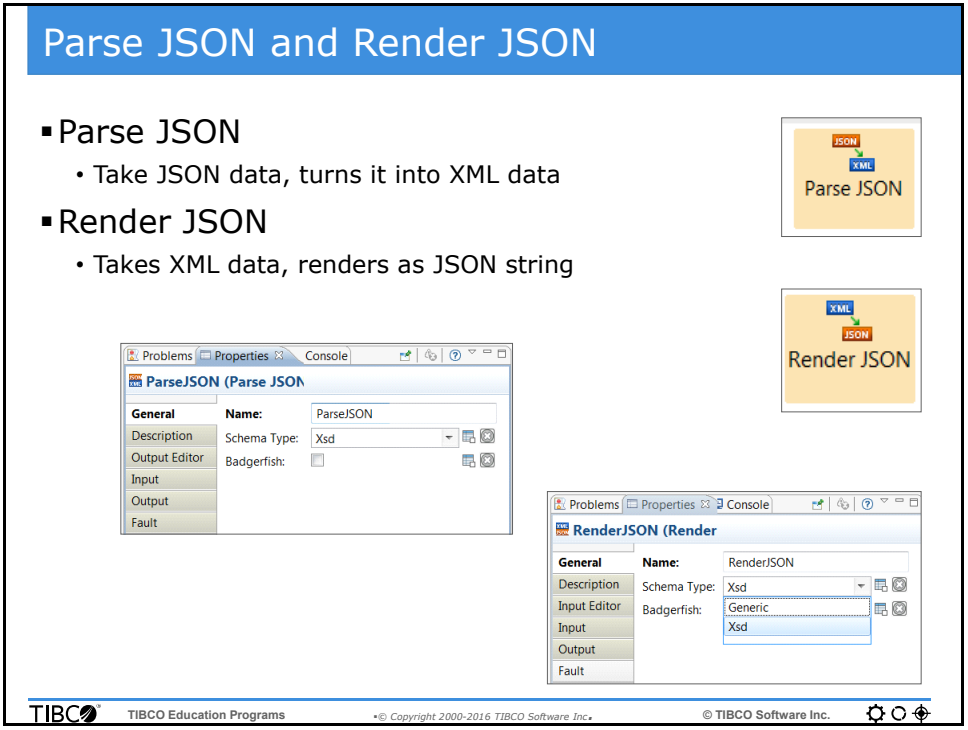
- Invokes RESTful web services
- Receives responses from Service Provider



TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Invoke REST API

The Invoke REST API asynchronous activity invokes RESTful web services and receives responses from the service provider. The Invoke REST API activity supports both Path and Query parameters. For Path Parameters, each path parameter needs to be enclosed in { } bracket. Query parameters are defined using the pattern /staticUrl/?



Parse JSON and Render JSON

The Parse JSON activity takes the JSON data, processes it, and converts it into an XML data. From the Output Editor you can define or reference an XML schema for the activity output. You can use the BW JSON Utils tool to generate an XML schema file with a specified JSON file. The output for the Parse JSON activity varies depending on the data schema you specified in the Schema Type field in the General tab.

The Render JSON activity takes XML data and renders it as a JSON string. The input data for the Render JSON activity is in XML format. The schema type of the XML data varies depending on the schema type you specified in the Schema Type field on the General tab. The Output tab contains the translated data in JSON string format.

Conversion between XML and JSON

- Schema types for mapping and validating XML data:
  - Generic
  - XSD
    - XML schema
- Conversion Rules:
  - Normal conversion
  - Badgerfish conversion

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Conversion between XML and JSON

The following are the supported schema types:

- Generic - Converts an XML string to a JSON string
- XSD - Converts an XML string to a JSON document defined using a schema specified in the Output Editor

The default value in this field is the XSD schema type.

Conversion rules:

- Normal Conversion
- Badgerfish conversion

Normal Conversion

- If Badgerfish is unchecked
  - Parse JSON activity
  - Render JSON activity
- Data loss possible
  - Repeated conversion
  - Depending on input data

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
{
  "table": {
    "xmlns": "http:\/\/www.w3.org\/TR\/html4\/",
    "tr": {
      "td": [
        "Apples",
        "Bananas"
      ]
    }
  }
}
```

```
<table>
    <xmlns>http://www.w3.org/TR/html4/</xmlns>
    <tr>
        <td>Apples</td>
        <td>Bananas</td>
    </tr>
</table>
```

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Normal Conversion

Compared to Badgerfish conversion, normal conversion is lossy but it generates much leaner and cleaner data.

When repeatedly converting data from XML to JSON and then converting back to XML, the resulting XML data may be not the same as the original XML data. It depends on your input data.

When repeatedly converting data from JSON to XML and then converting back to JSON, the resulting JSON data may be not the same as the original JSON data. It depends on your input data.

Badgerfish Conversion

- Accommodates element attributes & namespaces
- Adds overhead to JSON syntax
  - Namespace, attribute, and CDATA are preserved
- Each element given array for attributes and values
  - Attribute names preceded with **@** symbol
  - Values are represented with **$** symbol

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
{
  "table":{
    "@xmlns":"http:\/\/www.w3.org\/TR\/html4\/",
    "tr":{
      "td":{
        "$":"Apples"
      },
      "td#1":{
        "$":"Bananas"
      }
    }
  }
}
```

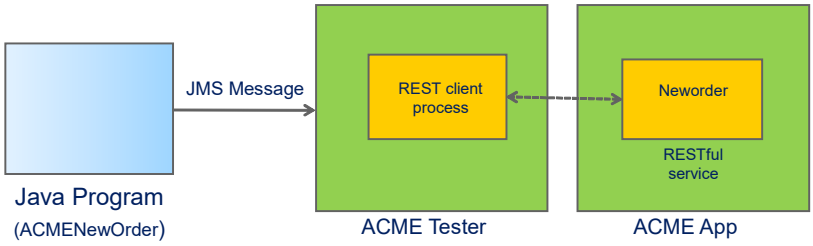
TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Badgerfish Conversion

Badgerfish conversion is a comprehensive conversion for parsing and rendering data between JSON format and XML format. The mapping covers XML namespace, XML attribute, CDATA, and so on. But XML comments, XML prologs, and DTD definition are ignored during the conversion.

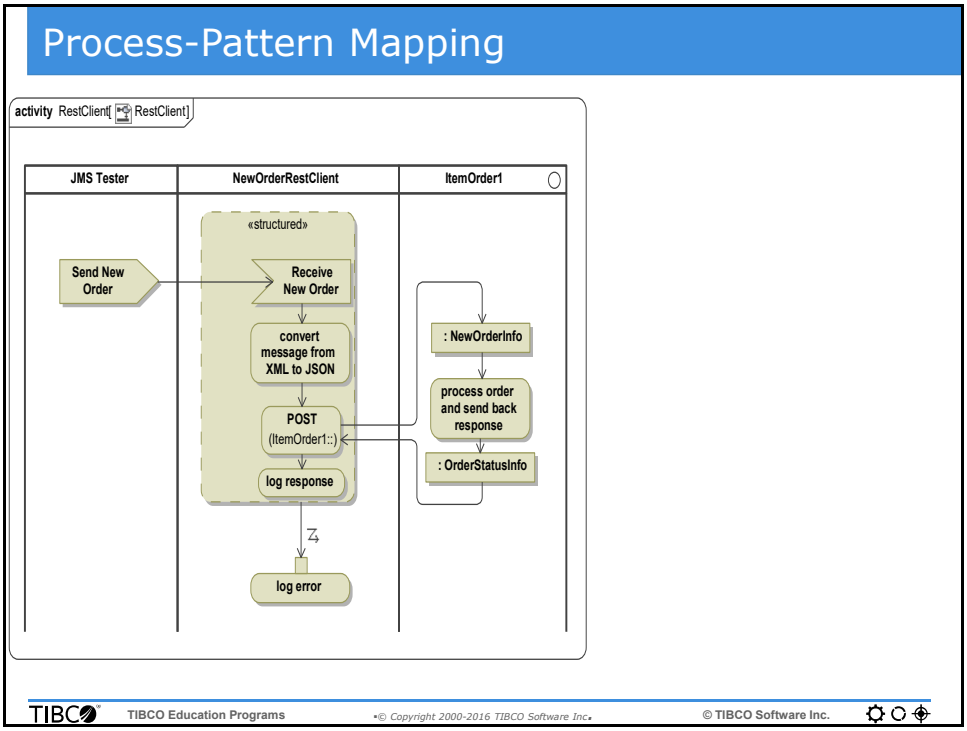
Business Case Implementation

- Design REST Client process
  - In ACME Tester Application
    - Receive Order Message
    - Invoke NewOrder REST Service



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Business Case Implementation


Process-Pattern Mapping
activity RestClient[RestClient]
JMS Tester
NewOrderRestClient
ItemOrder1
«structured»
Send New Order
Receive New Order
convert message from XML to JSON
POST (ItemOrder1::)
log response
log error
: NewOrderInfo
process order and send back response
: OrderStatusInfo
TIBCO
TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Review

Review

- REST and JSON Palette for client development
- Normal vs. Badgerfish conversion options
- Invoke API requires HTTP client connection

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Notes

Lab H REST Clients

Overview

In this lab you will create the tester process that will be used to test the New Order RESTful service. To do this, you will receive a JMS message and then convert it to JSON format before invoking the RESTful service.

Objectives

- *Build RESTful Client, page 244*
- *Use a RESTful Client,* page 251

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous lab has been completed

Directions

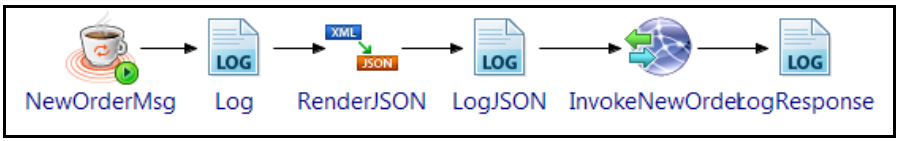
- Complete the exercises that follow

Lab H - Exercise 1: Build RESTful Client

Overview

In this exercise you will build a process that receives a JMS message and converts the message to JSON format. The JSON message is then used to invoke the New Order RESTful service.

The completed process looks as shown below;



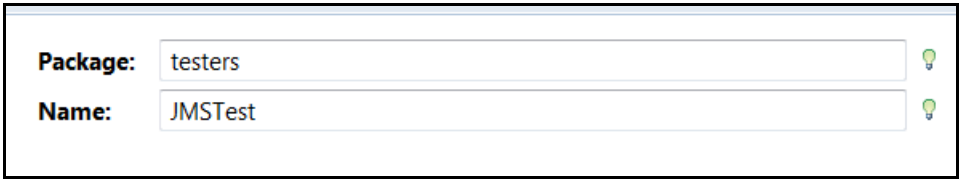
Steps

1. Close the JMSSend project to avoid processing of Queue messages.

- In Business Studio select **JMSSend** and **JMSSend.application**
  - Right-click and choose Close Project...

2. Change the name of the tester process. and the JMS receive activity.

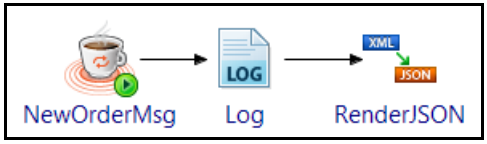
- In TIBCO Business Studio, open *ACMETester* > *Processes* > *testers* > ***JMSTest***
- Click Process boundary and view *Properties* of the process



- Click on the light bulb to the right of name to refactor (rename the process)
- New Name: **`NewOrderRESTClient`**
- Also, rename the activity **JMSReceiveMessage** to **`NewOrderMsg`**

3. Add a Render to JSON activity and configure it.

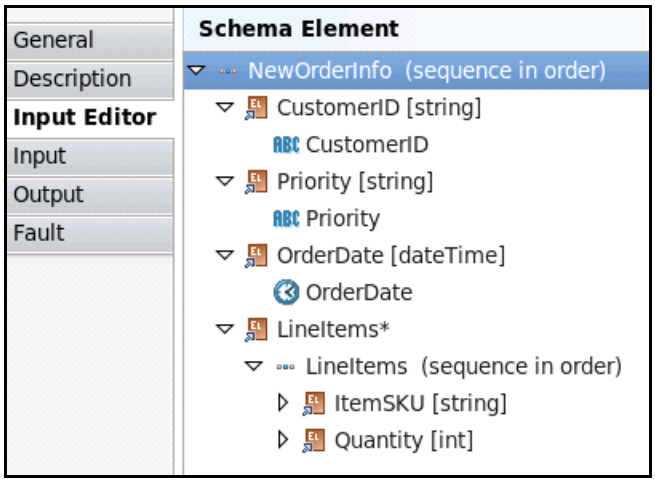
- Add a **RenderJSON** activity next to the log activity as shown:



4. Update the RenderJSON activity to use the NewOrderInfo as input schema.

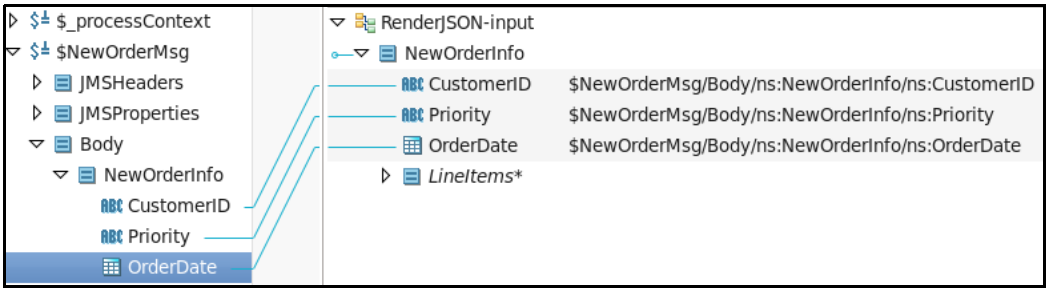
- For **RenderJSON** activity, choose *Properties* > ***Input Editor***
- Click on **Select Input Editor Element** button (shown below)

  - Locate *ACMETester* > *Schemas* > ***ACMEOrderItemSchema.xsd***
  - Choose **NewOrderInfo**, confirm that the details are loaded

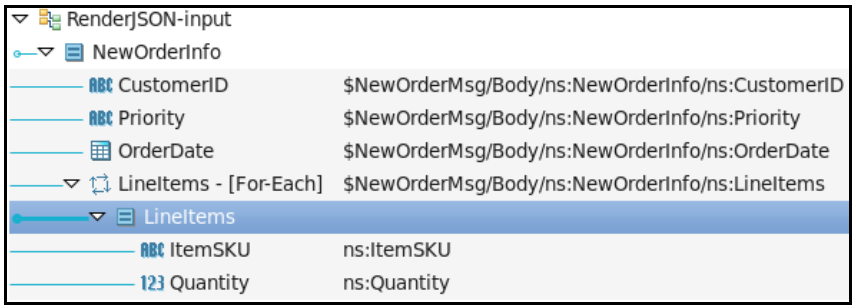


5. Map the incoming schema to the schema you just defined.

- To map the incoming input, click ***Input*** tab
  - Map the items shown below



  - Map *NewOrderInfo* > *LineItems** to **LineItems***
    - Select **For each...**
    - Select **Next** and **Finish**
  - Verify your mappings



  - **Save** your changes

6. Add a log activity to record the JSON output produced by the render activity.

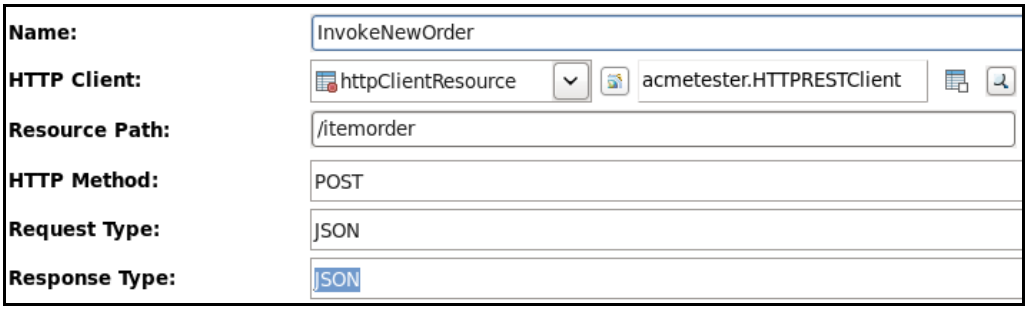
- Add a **log** activity next to **RenderJSON** activity and configure it
- Transition from **RenderJSON** to **Log** activity
  - *General* Tab:
    - Name: **`LogJSON`**
  - *Input* Tab:
    - Message: **concat(" JSON Order: ", `$RenderJSON/jsonString`)**

ABC message	concat(" JSON Order :", $RenderJSON/jsonString)

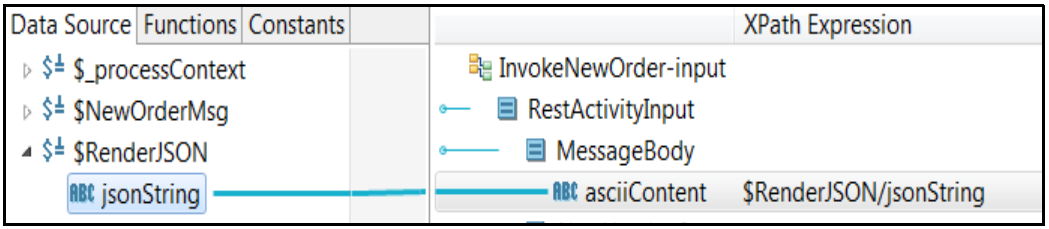
- **Save**

7. Add an activity to Invoke the RESTful New order service.

- Add an **Invoke REST API** (REST and JSON palette) activity next to **LogJSON**
- Configure the activity, ***General*** tab
  - Name: **`InvokeNewOrder`**
  - HTTP Client: Click **Choose/Create** (Magnifying glass icon)
    - Select **Create Shared Resource**
    - Resource Name: **HTTPRESTClient**, click Finish
  - Confirm that the HTTP Client field is filled up
- Back in *General* tab (changes in bold)
  - Resource Path: **/itemorder** Escribirlo
  - HTTP Method: **`POST`**
  - Request Type: JSON
  - Response Type: JSON



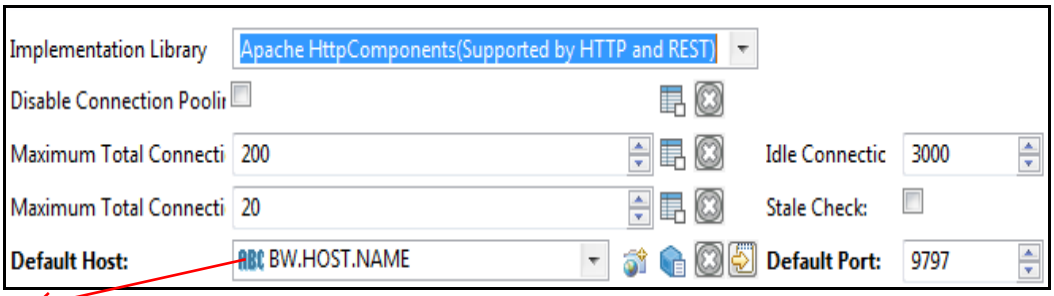
- **Input Tab:**
  - Map *jsonString* to *asciiContent* as shown:



- Confirm settings as shown above
- **Save**

8. Update the port for the HTTP Client resource.

- Browse to *ACMETester> Resources >* ***acmetester***
- Double-click and open **HTTPRESTClient**
- Change Implementation Library to the one supported by REST as shown:



En mi caso tuve que seleccionar la Module Property BW.HOST.NAME Sólo tenía "localhost" en un inicio.

- Update Default Host to Module Property **HOST.NAME**
- Update Default Port to **`9797`**
- **Save** and close the view

9. Add log activity to record response received from the RESTful service.

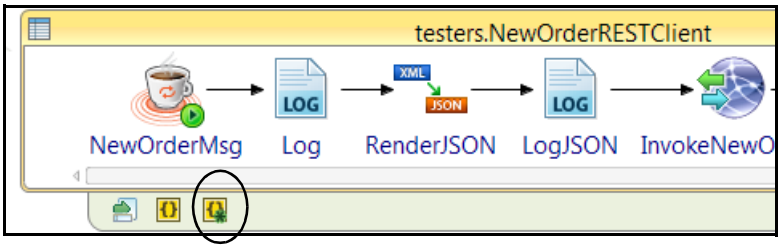
- Add a **Log activity** next to **InvokeNewOrder**
- Transition from **InvokeNewOrder** to **Log** activity
  - *General* Tab:
    - Name: **`LogResponse`**
  - *Input* tab:

ABC message	concat (" New Order Response, Status Code : ",xsd:string($InvokeNewOrder/statusCode), " Content : ", $InvokeNewOrder/MessageBody/asciiContent)

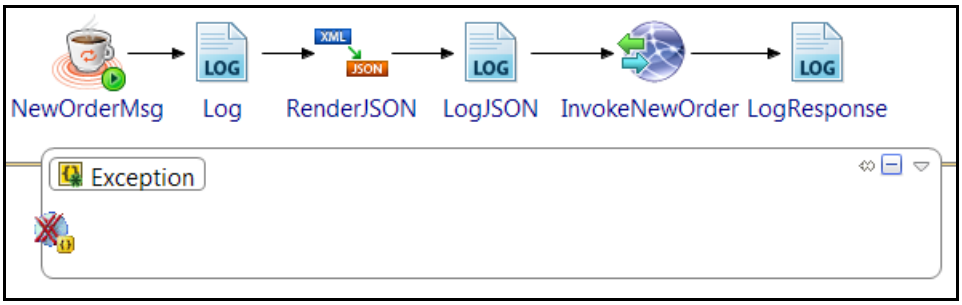
- **Save** changes when done

10. Add an error handler to catch any errors generated during execution.

- Hover under the bottom of the process diagram boundary
- The error handling menu appears as shown:

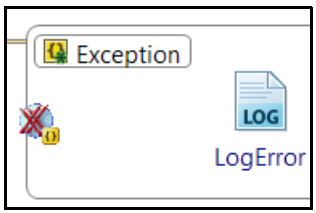


- Click on the **Create Catch All** icon (circled above)
  - An exception handler sub routine is created as shown below



11. Add a log activity to document the error details.

- Add a log activity in the exception routine
  - *General* Tab
    - Name: **`LogError`**
  - *Input* Tab
    - message: **$FaultDetails/StackTrace**



- **Save** your changes
- Confirm that there are no problems with your solution

Lab H - Exercise 2: Use a RESTful Client

Overview

In this exercise you will test the RESTful client you have just created. To test, you will define a Java project and import code to send JMS Queue messages.

Steps

1. Run the java application and confirm that a message is sent.

Iniciar el EMS Server para ejecutar este Ejercicio

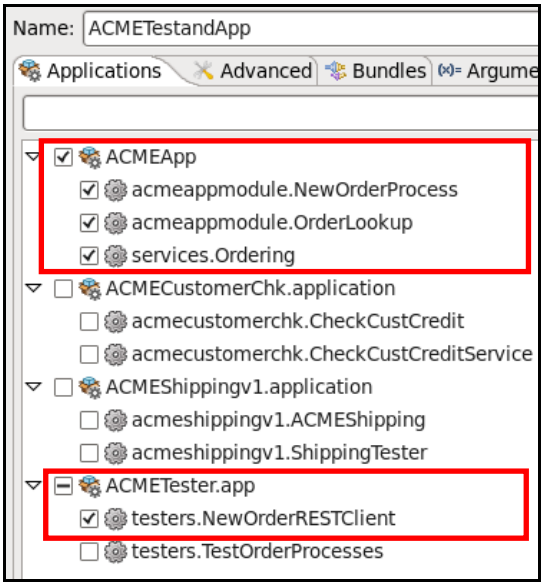
- Locate and open the **JMSNewOrder** project
- Browse to *src* > ***com.tibco.tibedu.jmstest***
- Double-click to open **NewOrderACME.java**
- Right-click on the java code and choose *Run As* > ***Java Application***
  - Examine output in the Console window

```
destination is: ACME.NewOrder
Sending TextMessage={ Header={ JMSMessageID={null} JMSDestination={null} JMSRepl
yTo={null} JMSDeliveryMode={PERSISTENT} JMSRedelivered={false} JMSCorrelationID=
{null} JMSType={null} JMSTimestamp={0} JMSDeliveryTime={0} JMSExpiration={0} JMS
Priority={4} } Properties={ } Text={<?xml version='1.0' encoding='UTF-8'?><ns0:N
ewOrderInfo xmlns:ns0='http://www.tibco.com/schemas/testschemas/Schema.xsd'> <ns
0:CustomerID>CUST1</ns0:CustomerID> <ns0:Priority>CUST1</ns0:Priority> <ns0:Orde
rDate>2014-08-08T11:11:12.484-08:00</ns0:OrderDate><ns0:LineItems> <ns0:ItemSKU>
ITEM1</ns0:ItemSKU> <ns0:Quantity>10</ns0:Quantity></ns0:LineItems>ns0:LineItems
> <ns0:ItemSKU>ITEM2</ns0:ItemSKU> <ns0:Quantity>20</ns0:Quantity></ns0:LineItem
s> </ns0:NewOrderInfo>} }
```

  - You may have to scroll to see complete output
- The program terminates after it finishes running

2. Run the BusinessWorks Applications.

- Choose *Run* > *Run Configurations*
- Create a new test with the name of ***ACMETestandApp***
  - Include ACMEApp and ACMETester ~~ACMETest~~ as show below



- Run *ACMETestandApp*

Así debe ser la salida:

```
13:56:58.170 INFO  [Thread-25] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300006: Started BW Application [ACMEApp:1.0]
13:57:05.176 INFO  [bwEngThread:In-Memory Process Worker-3] c.t.b.p.g.Log.ACMETester.Log - +++ Receiver1-NewOrderMessage for CustID:  CUST1 HIGH 2014-08-08T11:11:12.484-08:00 ITEM1
10 ITEM2 20
13:57:05.187 INFO  [bwEngThread:In-Memory Process Worker-3] c.t.b.p.g.Log.ACMETester.LogJSON -  JSON Order: {"CustomerID":"CUST1","Priority":"HIGH","OrderDate":"2014-08-08T11:11:
12.484-08:00","LineItems":[{"ItemSKU":"ITEM1","Quantity":10},{"ItemSKU":"ITEM2","Quantity":20}]}
13:57:05.692 INFO  [bwEngThread:In-Memory Process Worker-5] c.t.b.p.g.Log.ACMETester.LogResponse - New Order Response, Status Code : [200] Content : [{"OrderID":"NOT AVAILABLE","
CustomerID":"NA","Status":"NA","OrderAmount":0.0,"HandlingAmount":0.0,"TotalAmount":0.0,"DateChanged":"2017-05-14T13:57:05.531-07:00"}]
```


3. Confirm that the REST client works successfully.

- Right-click on the **NewOrderACME** java code and choose *Run As* > ***Java Application***
- Return to the console and verify that the application runs and receives the JMS Message

```
00006: Started BW Application [ACMEApp:1.0]
Log - Received Lookup request for OrderID: TEST1
Log - New Order JMS Message Received
LogJSON -  JSON Order :{"CustomerID":"CUST1","Priority":"HIGH","OrderDate":
 New Order Response, Status Code : 200 Content : {"OrderID":"NOT AVAILABLE"
```

- If you encounter any errors, refer to the **troubleshooting** section at the end of this lab
- Keep the BusinessWorks application running for the next step

4. Examine error handling capabilities of the process.

- Open the **NewOrderACME.java** program
- Update it to remove an XML tag, which makes the XML invalid
  - Remove the tag marked with the circle below:
  - *Before update*

```
msg = session.createTextMessage();
String xmlString = "<?xml version='1.0' encoding='UTF-8'?>"
            + "<ns0:NewOrderInfo xmlns:ns0='http://www.tibco.
xmlString = xmlString + CustomerID + Priority + OrderDate +
            + "</ns0:LineItems>";
```

  - *After removing tag:*

```
msg = session.createTextMessage();
String xmlString = "<?xml version='1.0' encoding='UTF-8'?>"
            + "NewOrderInfo xmlns:ns0='http://www.tibco.com/
xmlString = xmlString + CustomerID + Priority + OrderDate +
            + "</ns0:LineItems>";
```

  - **Save** your changes
- Right-click the Java program and choose *Run As* > ***Java Application***

5. Observe the errors in the BusinessWorks engine.

- The engine displays the error captured by the LogError activity as shown:

```
rker-1] c.t.b.p.g.Log.ACMETester.Log - +++Receiver1-NewOrderMessage for CustID: CUST1 HIGH 2014-08-08T1
rker-1] c.t.b.p.g.Log.ACMETester.LogJSON -  JSON Order: {"CustomerID":"CUST1","Priority":"HIGH","OrderD
rker-3] c.t.b.p.g.Log.ACMETester.LogResponse -  New Order Response, Status Code :200 Content: {"OrderID
rker-4] c.t.b.p.g.Log.ACMETester.LogError - Error occurred when attempting to receive a JMS message-{Pr
 com.tibco.bw.palette.jms.runtime.receive.AbstractJMSReceive.onException(AbstractJMSReceive.java:172)
 com.tibco.bw.jms.shared.primitives.JMSReceiver$SessionController.run(JMSReceiver.java:244)
 java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
 java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
 java.lang.Thread.run(Thread.java:745)
 com.tibco.bw.jms.shared.api.exceptions.JMSMessageReceiveException: JMS Message Deserialization failed.
 com.tibco.bw.jms.shared.primitives.JMSReceiver$SessionController.run(JMSReceiver.java:238)
```

- Keep the BusinessWorks Application running

6. Fix the error in the Java code.

- Open **NewOrderACME.java** program
- Restore the missing XML tag as shown below:

```
msg = session.createTextMessage();
String xmlString = "<?xml version='1.0' encoding='UTF-8'?>"
            + "<ns0:NewOrderInfo xmlns:ns0='http://www.tibco.
xmlString = xmlString + CustomerID + Priority + OrderDate +
            + "</ns0:LineItems>";
```

- **Save** changes
- Run the Java program again (*Run As* > ***Java Application***)
- Verify that the BusinessWorks application runs successfully without errors
- **Stop** the running BusinessWorks application

7. Export your solution built so far as LabREST.zip.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File: `/home/tibco/bw6edu/dev/exports/RESTClient.zip`
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created

Troubleshooting

- This error indicates improper setting of the HTTP Client resource:

```
Caused by:
java.lang.NoClassDefFoundError: org/apache/http/config/RegistryBuilder&#xd;
at org.glassfish.jersey.apache.connector.ApacheConnector.createConnectionMan
at org.glassfish.jersey.apache.connector.ApacheConnector.getConnectionManage
at org.glassfish.jersey.apache.connector.ApacheConnector.&lt;init&gt;(ApacheCon
```

- This error indicates that the URI is incorrect

```
INFO  [bwThread:In-Memory STWorkProcessor-4] c.t.b.p.g.Log.ACMEAppModule.Lo
TIBCO-BW-PALETTE-REST-2000027: Client received a 4xx response for invocatio
resource path http://localhost:9797/itemorder-{ActivityName=InvokeNewOrder,
ProcessName=testers.NewOrderRESTClient, ModuleName=ACMEAppModule}
at com.tibco.bw.palette.restjson.runtime.RestInvoke$RestInvokeRunnable.run
(RestInvoke.java:254)
at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source)
at java.util.concurrent.FutureTask.run(Unknown Source)
at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
```

Notes

Unit 9 Using Groups and Loops

In this unit you examine the transaction capabilities of TIBCO ActiveMatrix BusinessWorks.

Topics

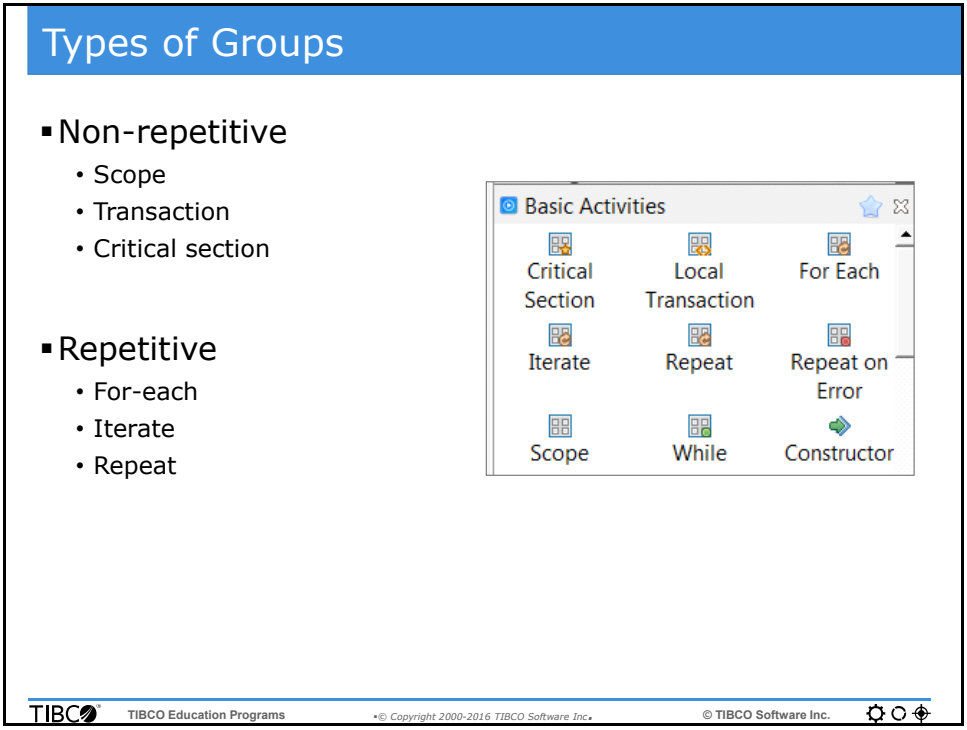
- *Objectives,* page 258
- *Types of Groups,* page 259
- *Grouping For Repetitive Actions,* page 260
- *Using Scope,* page 261
- *Error Handling Details,* page 262
- *Implementing Transactions,* page 263
- *Using Checkpoints with Transactions,* page 264
- *Process Inventory Definition,* page 265
- *Grouping Best Practices,* page 266
- *Review,* page 269
- *Lab Overview,* page 270

Objectives

- Knowledge
  - Discuss group resource capabilities
  - Describe transaction capabilities of ActiveMatrix BusinessWorks
- Skills
  - Develop inventory update process
    - Using JDBC transaction group
  - Test transactional processing

TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Objectives

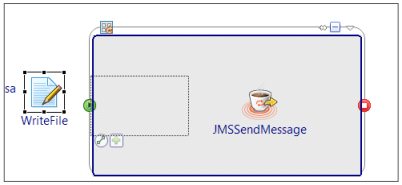


Types of Groups

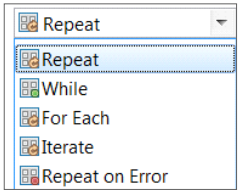
Groups enable you to put one or more activities together and configure the group as needed.

Grouping For Repetitive Actions

- Defining a group
  - Select multiple activities
    - Choose group type
  - Add group type
    - Drag activities into group



- Types of repetitions
  - For-each
  - Iterate
  - While
  - Repeat on error



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Grouping For Repetitive Actions

Loops are groups with conditions which follow a pattern at runtime like initialize the loop, update the loop at each iteration, and test conditions for the loop to stop iterating. The following types are available:

For Each - used to loop for a specific number of iterations with a counter ranging from a start value to an end value.

Iterate - This loop has a simple index variable that can be used to count each iteration and the loop executes for the number of iterations specified.

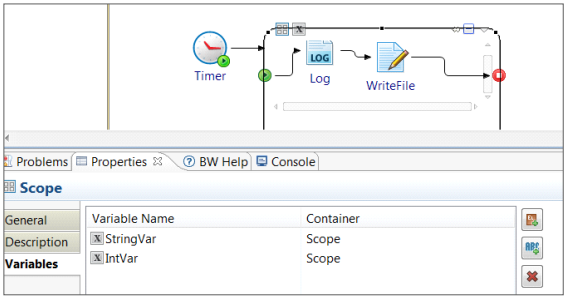
Repeat - This loop has a simple index variable that can be used to count each iteration and has a conditional expression to determine when to stop.

Repeat on Error - This loop involves a retry mechanism. I f any activity in the loop throws a fault, the condition expression is evaluated to determine if the loop should be repeated.

While - This loop has a simple index variable that can be used to count each iteration and has a conditional expression to determine when to stop. The condition for the While loop is tested at the beginning of each iteration and the loop may never be executed if the condition is initially false.

Using Scope

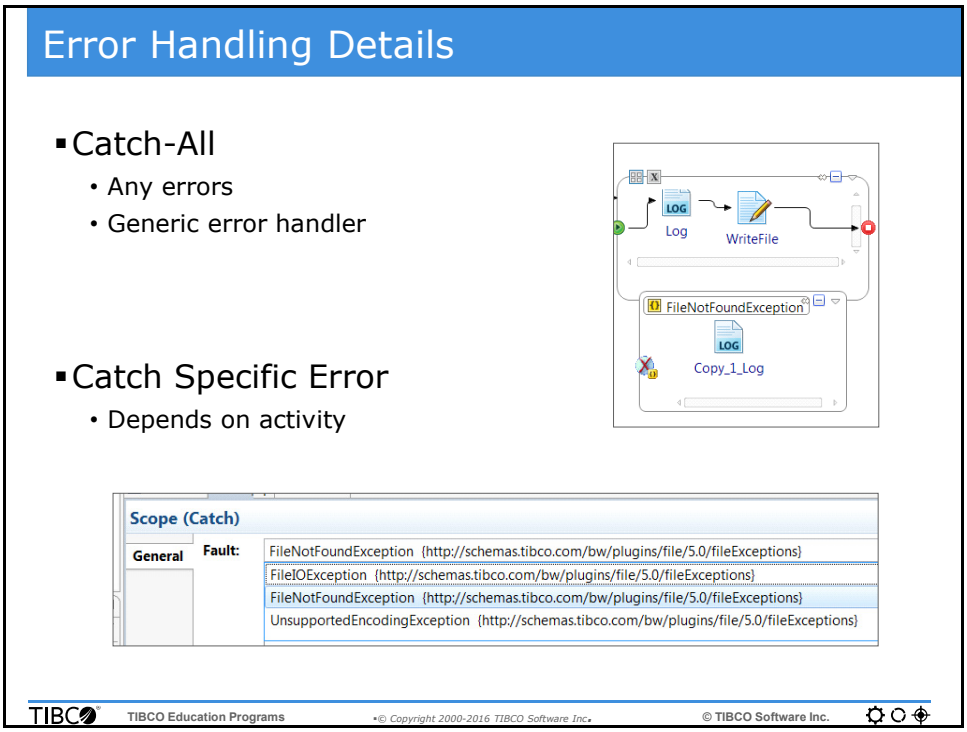
- Scope
  - Enclose one or more activities
  - No default action
- Usage
  - Add variables
  - Error handling



TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Using Scope

Scope group comprises a collection of nested activities. The scope is shared by all the nested activities. You can use a Scope to contain a unit of work, by making it easy to manage and execute, and, if necessary, reverse. For example, if a customer cancels an online order, the money must be returned and the online order canceled, without affecting other orders. You can use a Scope to manage these activities as a unit.



Error Handling Details

Fault handlers are used to catch faults or exceptions and create fault-handling procedures to deal with potential errors.

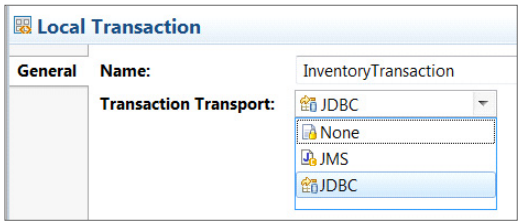
Fault handlers are defined at the scope level allowing you to catch faults or exceptions thrown by activities within a scope. There are two types of fault handlers:

- Catch Specific Fault
- Catch All Faults.

Fault handlers can be defined at the process level, or at a scope level within a process. The diagram below shows two fault handlers - one defined at the process level and the other at an inner scope level.

Implementing Transactions

- Available for
  - JDBC activities
  - JMS Activities



- Multiple activities participate in transaction
  - JDBC: must use the same database connection
  - JMS : share JMS session, same messaging style
- Transaction management
  - By the database
  - JMS by EMS server

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Implementing Transactions

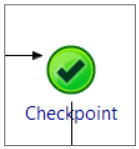
The following are the core activity types that have local transactional capabilities:

- JMS
- JDBC

A Local Transaction group has to be further configured as a specific type of a Local transaction or Transaction Transport. The Transaction Transport type determines the activities that can participate in this local transaction group. For example, “JDBC” Transaction Transport only provides transaction support for certain JDBC activities.

Using Checkpoints with Transactions

- Save critical process data and state
  - Recover process after failure
  - Saving options: file or database



- Each process instance keeps only its most recent checkpoint data
- Must not appear within a transaction group
  - Best practice: place before and after a Repeat-on-Error group that suspends

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Using Checkpoints with Transactions

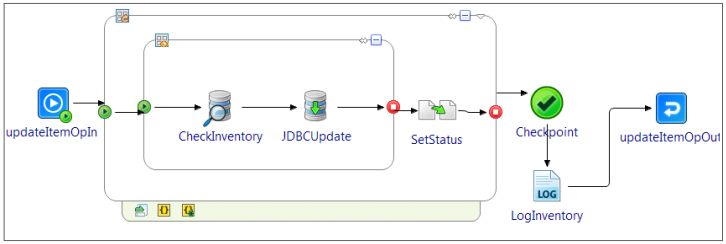
A checkpoint activity saves the state and data of a process instance so that it can be recovered at a later time in case of a failure.If a ActiveMatrix BusinessWorks engine fails, all processes that have a Checkpoint activity can be recovered and resume execution from the last checkpoint executed in the process instance.

Only the most recent state is saved by a checkpoint. If you have multiple checkpoints in a process, only the state from the last executed checkpoint is available for recovering the process.

Checkpoints save the state of the entire process instance. A process (parent process) can call another process (sub-process) in two ways: in-line and non-inline. An in-line sub-process is executed as part of the parent process job, while the non-inline sub-process spawns a new job. When a checkpoint activity is executed in an in-line sub-process, the checkpoint saves the state of the current process instance, including the state of the parent process(es). However, when a checkpoint occurs in an non-in-line sub-process, the checkpoint saves the state of the spawned process instance only.

Process Inventory Definition

- Subprocess to *Process Order*
- Checks and updates inventory
  - In a single JDBC transaction
- Checkpoint saves state after transaction commit
- Error handling if transaction fails (rolls back)



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Process Inventory Definition

The inventory sub process is designed to show the usage of two kinds of groups: the transaction group and the iterate group.

For each line item, the item details are fetched from the database and updated. The select and update is performed as part of a single transaction.

The transaction is within the iterate and not the other way around, why?

Since we are designing the system for multiple requests, it is important that we update each line item and commit the transaction as quickly as possible. Holding up the database for say 20 line items would slow down processing for the other orders.

In a traditional EAI, where the number of messages is low, you could have all line items updated in a single transaction. In a WOA (Web Oriented Architecture), the logic is updated as shown.

Grouping Best Practices

- Use group activities for:
  - Error handling
  - Iterations and flow control
  - Synchronizing instances
  - Transactions
- Use *Group* to implement transactions in ActiveMatrix BusinessWorks
  - JDBC
  - JMS

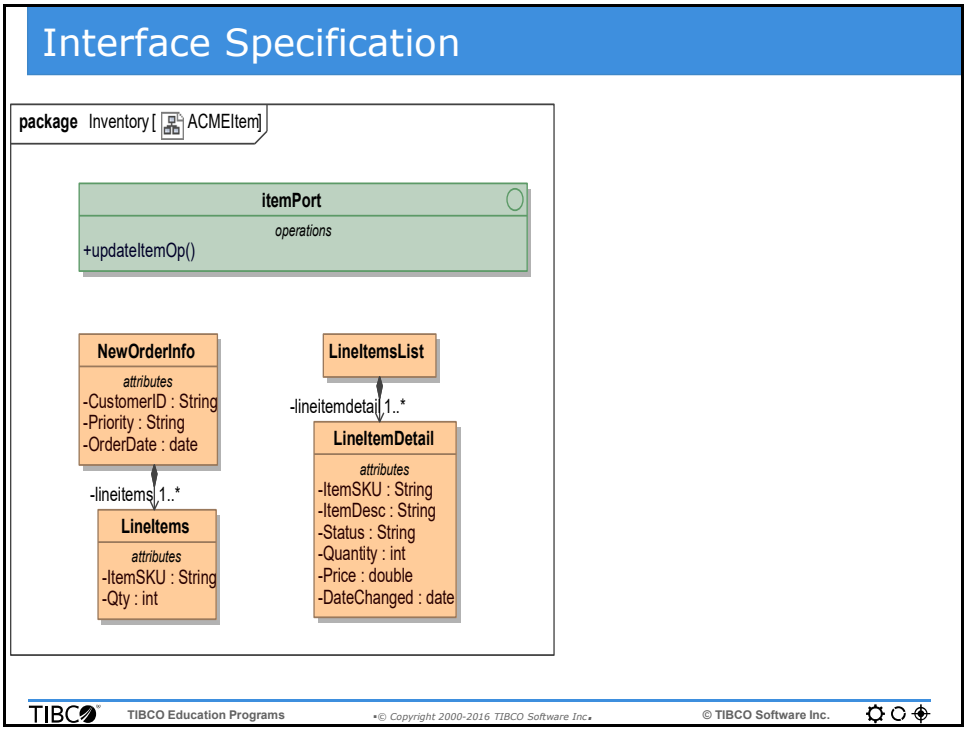
TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Grouping Best Practices

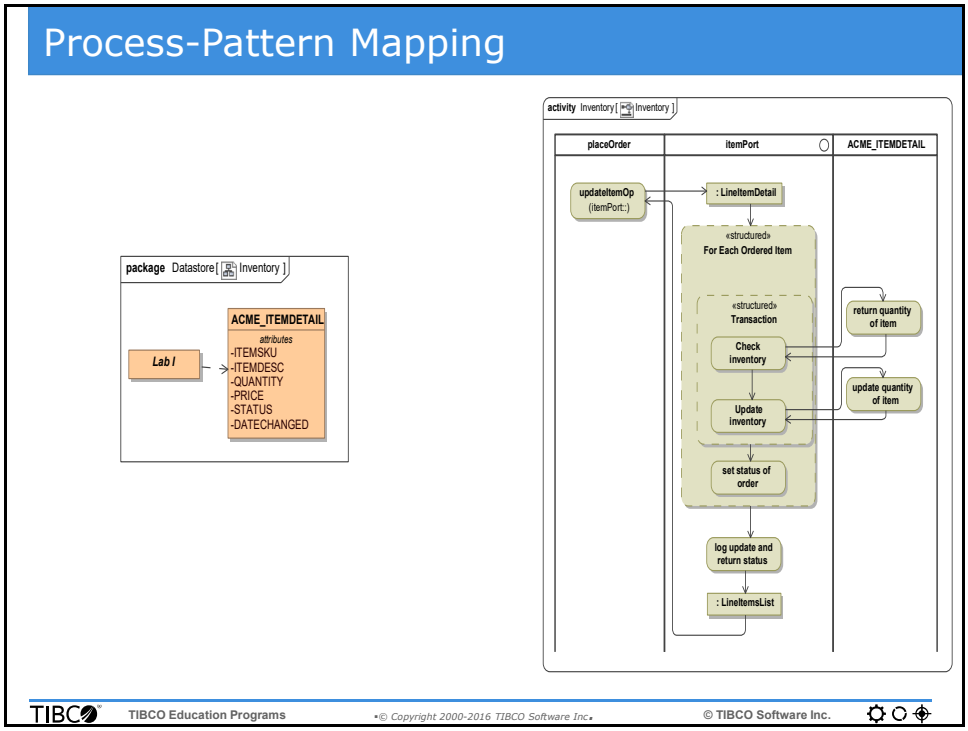
In later modules, we will examine the many uses of a group (scope).

To begin with, groups are used for iterations and transactions. Later we will see how groups are used for exception handling and compensation logic.

BusinessWorks 6 does not support an external transaction manager (Like BusinessWorks 5). Only local transactions are supported for the JDBC and JMS activities.



Interface Specification



Process-Pattern Mapping

Review

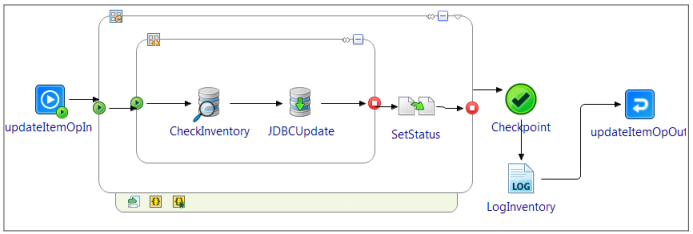
- Group functionality
  - Common error handling, Iteration, Synchronize instances
  - Alternative paths through group activities, Transactions
- Transaction JMS or Database:
- Database transaction: JDBC activities use **same database**
- Checkpoint: saves process data for recovery after failure

TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Design Item Inventory Process
  - Implement JDBC transaction
  - Iterate over all line items



TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Notes

Lab I Loops and Transactions

Overview

In this lab you will create the inventory process. This process will query and update the item table for each item requested in a new order. Select and update of all items in an order are done in a single transaction to maintain consistent database records.

Objectives

- *Adding Transactions, page 274*
- *Testing Transactions,* page 285

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous lab has been completed

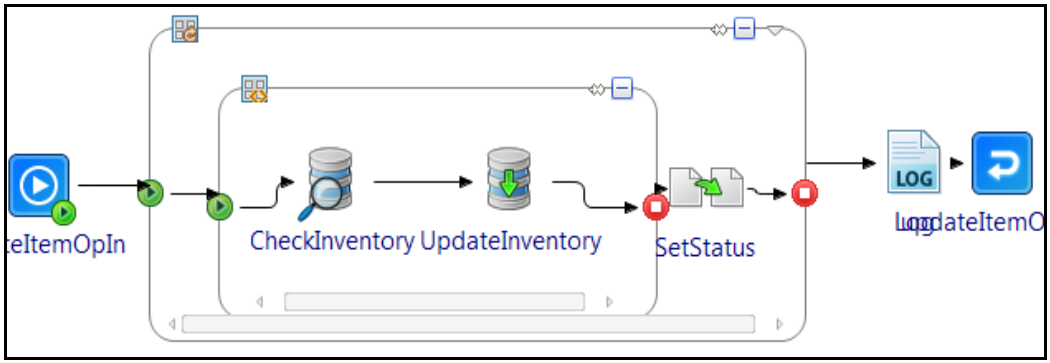
Directions

- Complete the exercises that follow

Lab I - Exercise 1: Adding Transactions

Overview

In this exercise you will define a new process to update the item table based on a new order request. To ensure consistency all items in an order are updated in a single transaction. This is a picture of the completed process.



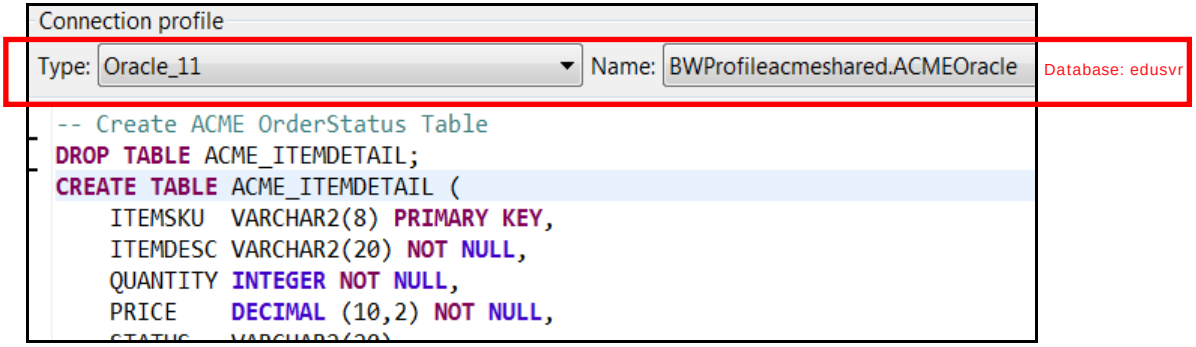
Steps

1. Open the database perspective and load script to create new table for Items.

- Launch Business Studio for workspace: **`/home/tibco/bw6edu/workspace`**
- Browse to *ACMEShared* > *Resources* > ***acmeshared***
  - Double-click **ACMEOracle** (JDBC Connection) to view its details
  - Click **Test Connection** to establish the database connection
- From the *File Explorer* (tab in bottom left window)
  - Expand the **Scripts** folder, and locate **ACME_ItemDtl.sql**
  - Drag it to the **General** section (*top half*) of the JDBC Connection
- This opens SQL Scrap Book view

2. Confirm that the script is loaded and execute it.

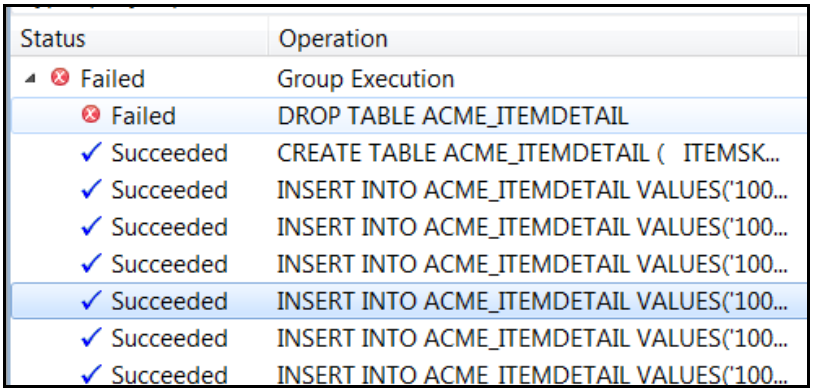
- Confirm that the script is loaded, and the database is connected



- Right click anywhere on the script itself and choose **Execute All**
  - For "Error during execution, continue?" click **Yes**

Tip: This executes the script and defines the ACME_ItemDetail table and inserts six rows into the table. Rows for ItemSKUs 1001 through 1006 are inserted.

- Below the script window, you should see the SQL Results window
  - It will show a status of **Failed**, since the DROP TABLE command fails
  - View the details by expanding the view as shown



Status	Operation
Failed	Group Execution
Failed	DROP TABLE ACME_ITEMDETAIL
Succeeded	CREATE TABLE ACME_ITEMDETAIL (ITEMSK...
Succeeded	INSERT INTO ACME_ITEMDETAIL VALUES('100...
Succeeded	INSERT INTO ACME_ITEMDETAIL VALUES('100...
Succeeded	INSERT INTO ACME_ITEMDETAIL VALUES('100...
Succeeded	INSERT INTO ACME_ITEMDETAIL VALUES('100...
Succeeded	INSERT INTO ACME_ITEMDETAIL VALUES('100...
Succeeded	INSERT INTO ACME_ITEMDETAIL VALUES('100...

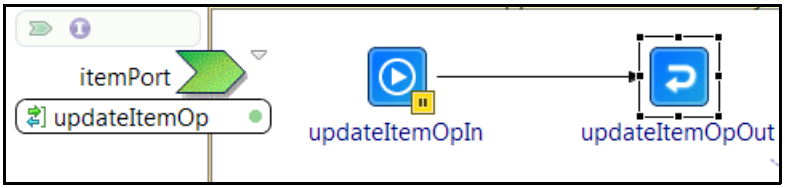
  - Close the *SQL Scrap book* view (window) - **do not save changes**
  - Also close the *SQL Results* view

3. Create a new process named ItemInventory in ACMEAppModule.

- Browse and open *ACMEAppModule >Processes >* ***acmeappmodule***
  - Right-click to create *New* ***> BusinessWorks Process***
    - Name: **`ItemInventory`**
  - Click **Finish**

4. Update the process to implement the update inventory operation.

- Select *ACMEShared >Service Descriptors >* ***ACMEItemWSDL.wsdl***
- Drag this WSDL on to the **ItemInventory** process
  - Choose **Implement Operation**
  - Since this WSDL has only one operation, it is added as shown:



- **Save** changes

5. Configure the input activity.

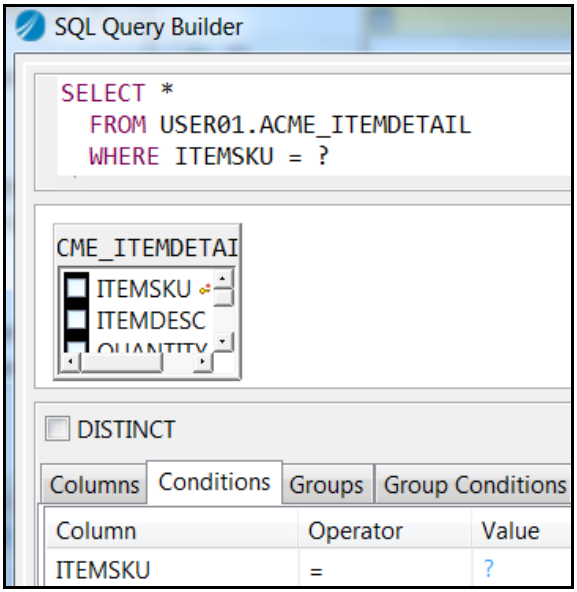
- Click **updateItemOpIn**, then choose **Properties**
  - *General* tab: Create Instance - **Checked**
  - *Output* tab: Verify *updateItemRequest > input >* ***NewOrderInfo*** is produced
- **Save** your changes

6. Add a JDBC Query activity that uses the previously defined JDBC resource.

- From *Palettes* > ***JDBC***, select the **JDBC Query** activity
  - Drop the activity between **updateItemOpIn** and **updateItemOpOut**
  - Verify all activities are connected
- Select JDBCQuery activity and configure it as follows:
  - *General* Tab:
    - Name: **`CheckInventory`**
    - JDBC Shared Resource: **ACMEOracle** Hacer click en la lupa y seleccionar ACMEOracle
  - Save the changes

7. Construct the query for selecting records from ItemDetail table.

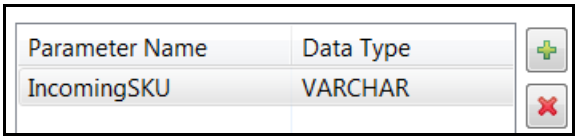
  - *Statement*: Use SQL Query Builder, to design the query shown below:



- **Save** your changes

8. Complete the JDBC Query activity by adding a parameter and mapping it.

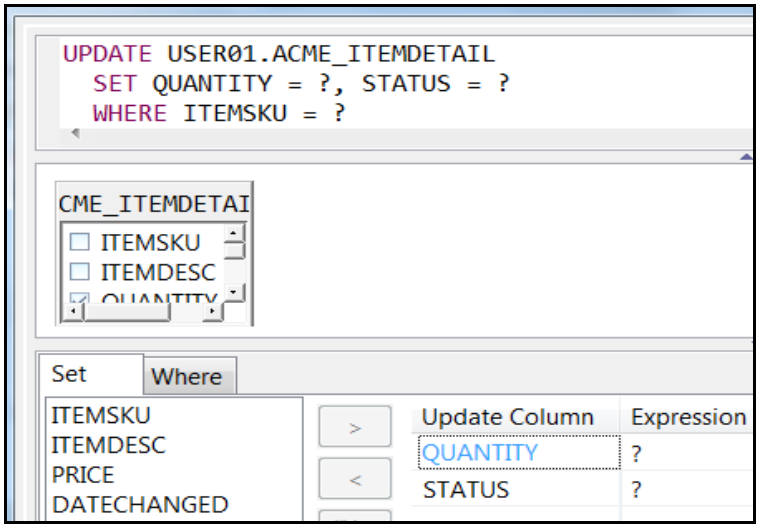
- Still in JDBCQuery (CheckInventory), Properties
  - *Statement* tab: Add a parameter and change its name to **`IncomingSKU`**



- Click **Fetch** IMPORTANTE click en el botoón FETCH antes de guardar.
- **Save** your change (Input mapping will be done in a few steps)

9. Add a JDBC update activity to update the ItemInventory table.

- Select the **JDBC Update** activity, drop it next to *CheckInventory* (JDBC Query)
- Configure the activity as follows:
  - *General* tab:
    - Name: **`UpdateInventory`**
    - JDBC Shared Resource: **ACMEOracle** (automatically selected)
  - *Statement* tab:
    - Use SQL Editor to build the update statement as shown:



10. Complete the definition of the UpdateInventory activity.

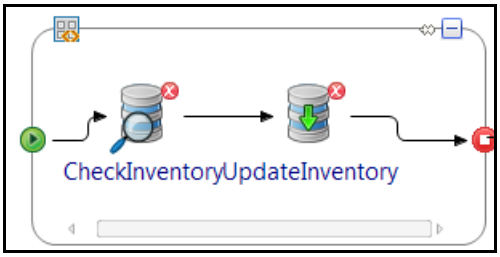
- In *Statement* tab, add three parameters and configure them as shown:

Parameter Name	Data Type
InQty	INTEGER
InStatus	VARCHAR
InSKU	VARCHAR

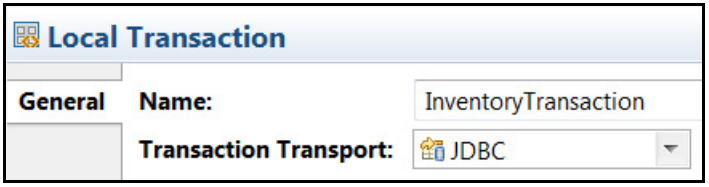
- **Save** changes (Input mapping to be done later)

11. Define a transaction group for the two database activities.

- Select **CheckInventory** and **UpdateInventory** activities (Use <Ctrl> Key)
- Right-click on any of the two activities
  - Select *Create Group* > ***Local Transaction***

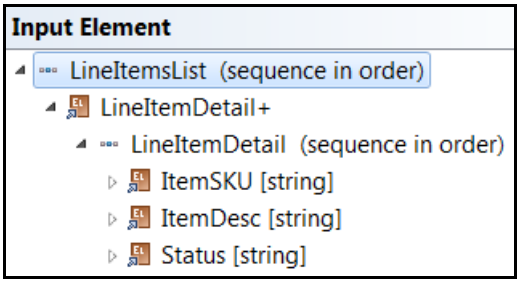


- Click on the group to select it, and in *Properties* > ***General*** tab:
  - Name: **`InventoryTransaction`**
  - Transaction Transport: **JDBC**



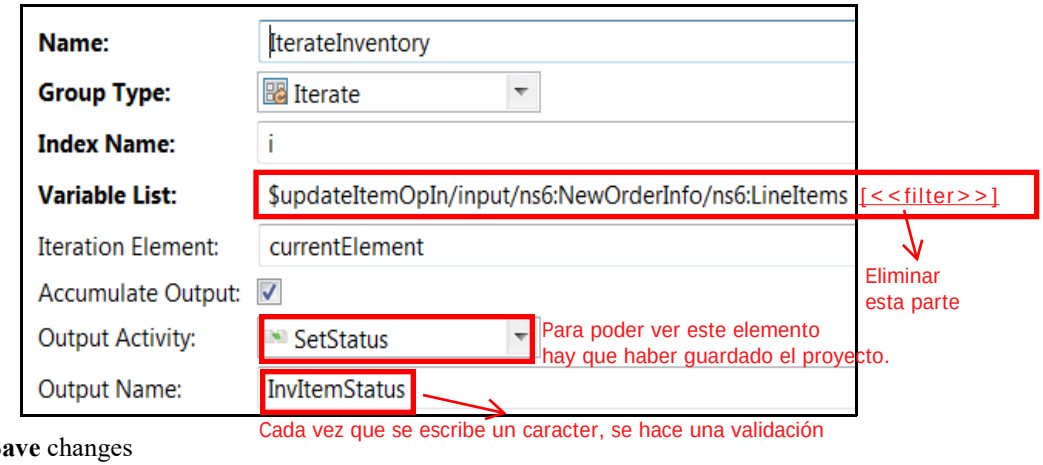
12. Add a mapper activity to collect results from the database activities.

- Add a **mapper** activity between the transaction group ~~between *transaction*~~ and *updateItemOpOut* and configure it:
  - In *General* tab, set Name: **`SetStatus`**
  - *Input Editor*: Select Input Element: *ACMEOrderItemSchema.xsd* > ***LineItemsList*** (Partial view shown)



13. Create an iterate group activity around the transaction and mapper activity.

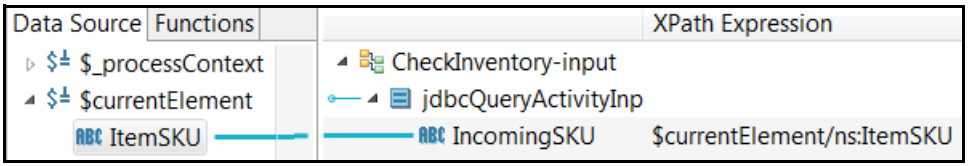
- Select **InventoryTransaction** and SetStatus (<Ctrl> Key)
  - Right click and choose *Create Group* > ***Iterate***
- Configure the group as shown below:
  - Variable List: `$updateItemOpIn/input/NewOrderInfo/LineItems`



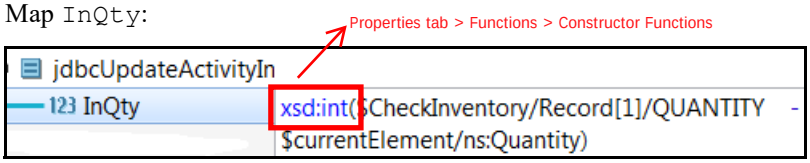
- **Save** changes

14. Complete the input mappings for the database activities.

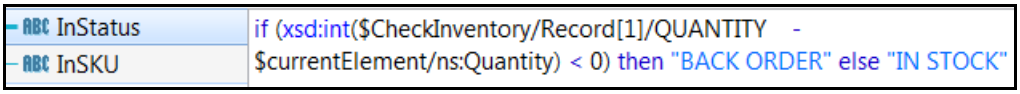
- Select *CheckInventory*> *Properties* > ***Input***
  - `IncomingSKU --> $currentElement/ItemSKU`



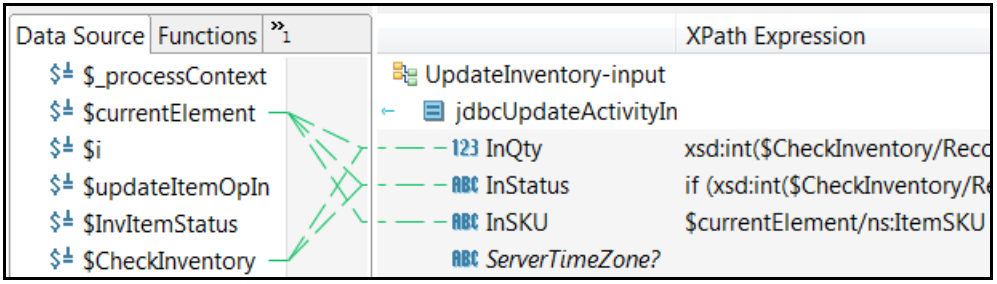
- Select *UpdateInventory* > *Properties* > ***Input***
  - Map `InSKU --> $currentElement/ItemSKU`
  - Map `InQty`:



  - Map `InStatus`:



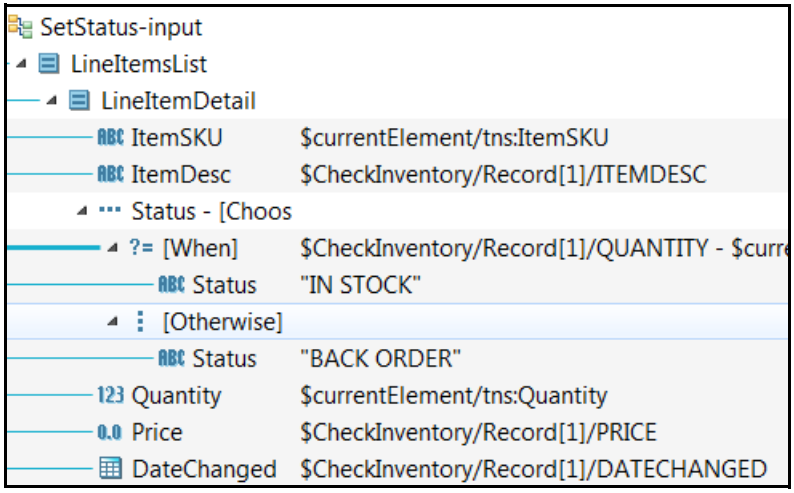
  - Verify mapping and **Save**



  - A partial view of the mapping is shown above

15. Complete the input mapping for SetStatus activity.

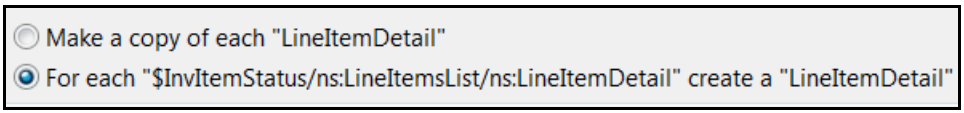
- Select **SetStatus** activity, then *Properties* > *Input*
  - `ItemSKU --> $currentElement/ItemSKU`
  - `ItemDesc --> $CheckInventory/Record[1]/ITEMDESC`
  - `Status`: Right-Click and choose **Surround with Choose...**
  - In the pop up window, *number of when conditions: 1* and click OK
    - When: `$CheckInventory/Record[1]/QUANTITY - $currentElement/Quantity >=0`
    - `Status --> "IN STOCK",`
    - Otherwise
    - `Status --> "BACK ORDER"`
  - `Quantity--> $currentElement/Quantity`
  - `Price--> $CheckInventory/Record[1]/PRICE`
  - `DateChanged -->CheckInventory/Record[1]/DATECHANGED`



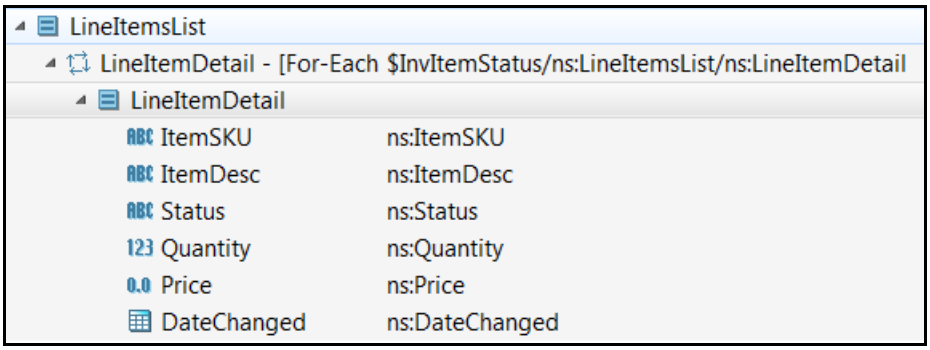
  - Confirm that there are no mapping errors
  - **Save** changes

16. Complete the mapping for the response sent by the process.

- Select *updateItemOpOut* > *Properties* **> *Input***
- For *ItemStatusReply* > *input* > *LineItemsList* > *LineItemDetail+*
  - Drag *InvItemStatus* > *LineItemsList* > *LineItemDetail+*
  - The Wizard appears, choose **For each**



  - Click **Next** and accept all defaults and select **Finish**
  - **Save** changes and verify mapping



  - Save changes

17. Add a log activity to indicate that inventory processing is completed.

- Add a **log** activity between the **iterate** group and **updateItemOpOut**
  - Name: **`LogInventory`**
  - Input
    - `message --> concat (" Inventory Updated for Customer ID: ", $updateItemOpIn/input/ns:NewOrderInfo/ns:CustomerID)`
- **Save** changes and verify that the process has no errors
- There is still one error in the solution, which will be resolved during testing

NOTA - Mi respaldo: Lab I Ex 1 Sol.zip tendrá este error evidentemente. Esta solución se prodrá usar para iniciar el ejercicio 2 del mismo Lab I.

Lab I - Exercise 2: Testing Transactions

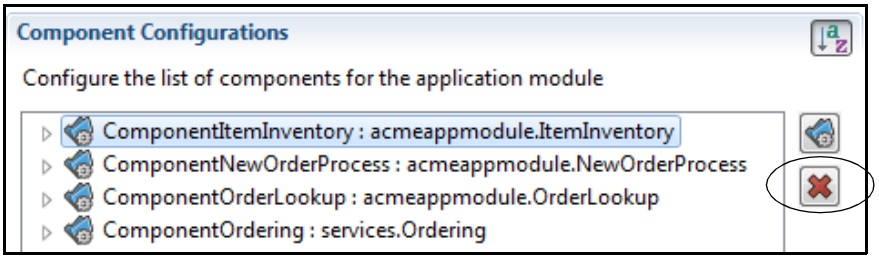
Overview

In this exercise you will update new order process to invoke update inventory. You will also update the new order REST service to call new order web service. You will test everything and fix any errors along the way.

Steps

1. Resolve the error with the application module.

- Browse to *ACMEAppModule* > ***Module Descriptors***
  - Double-click **Components** to view details
  - The list of starter processes for the module are displayed

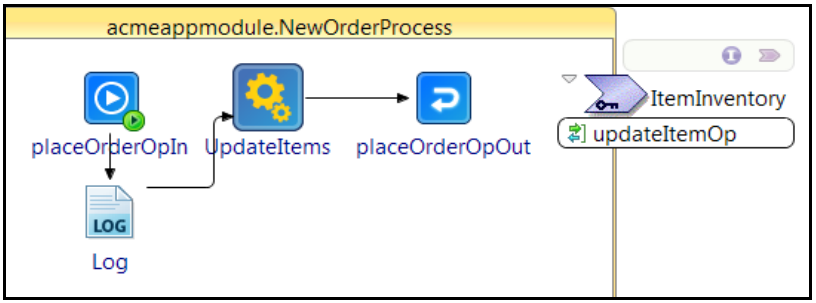


  - Select **ItemInventory** and click **X** to delete the starter
  - **Save** changes and there should be no more problems with the application

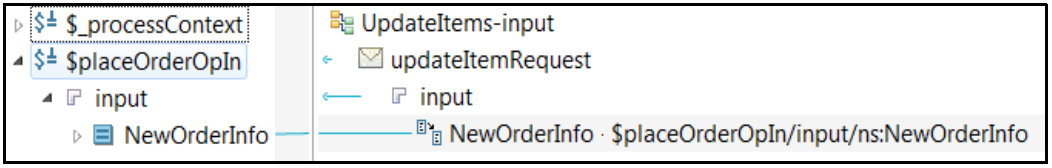
Tip: Deleting the component from the startup list converts this component to a sub-process. This is the key difference - a sub-process cannot be invoked by an external program.

2. Update new order process to include call to update inventory.

- Browse to *ACMEAppModule* > *Processes* > ***acmeappmodule***
  - Open (double-click**) NewOrderProcess**
- Drag **ItemInventory** process onto **NewOrderProcess**
  - Rename **Invoke** activity to **`UpdateItems`**
  - **Connect** *log* to *UpdateItems* and *UpdateItems* to *placeOrderOpOut*
- Updated process looks as shown below:



- The reference to UpdateInventory service is automatically added (right of process diagram)
- Complete mapping for **UpdateItems** - *Properties* > ***Input***
  - Drag *$placeOrderOpIn* > *input* > ***NewOrderInfo*** to *updateItemRequest* > *input* > ***NewOrderInfo***
  - In the wizard, choose **"Make a Copy of"**

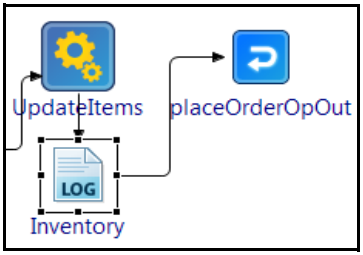


- **Save** changes

3. Add a log activity to indicate that inventory has been updated.

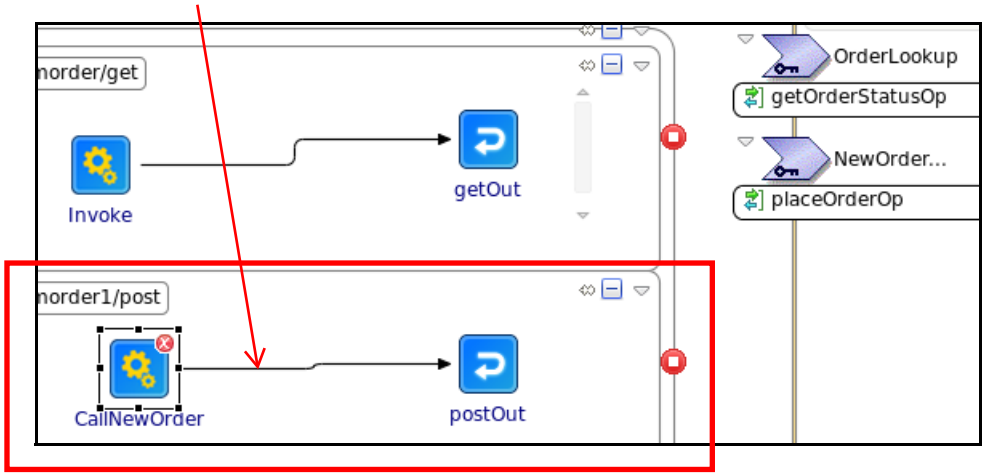
- Add a **Log** activity between *UpdateItems* and *placeOrderOpOut*
- Update this activity
  - Name: **`Inventory`**

    En el tab Input:
  - Message: **`"Item Inventory Processing Completed"`**



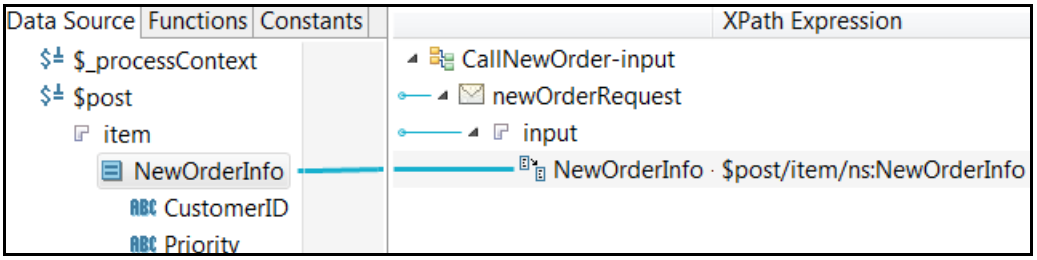
4. Update the New Order RESTFul service to invoke New Order process.

- Open *ACMEAppModule* > *Processes* > *services* > ***Ordering***
- Locate the process that handles the **POST** request
  - Drag *Processes* > *acmeappmodule* > ***NewOrderProcess*** next to **postOut**
  - This creates an invoke activity
- Rename the **invoke** activity to **`CallNewOrder`**
  - **Connect** CallNewOrder to PostOut



5. Map incoming data as input to the CallNewOrder activity.

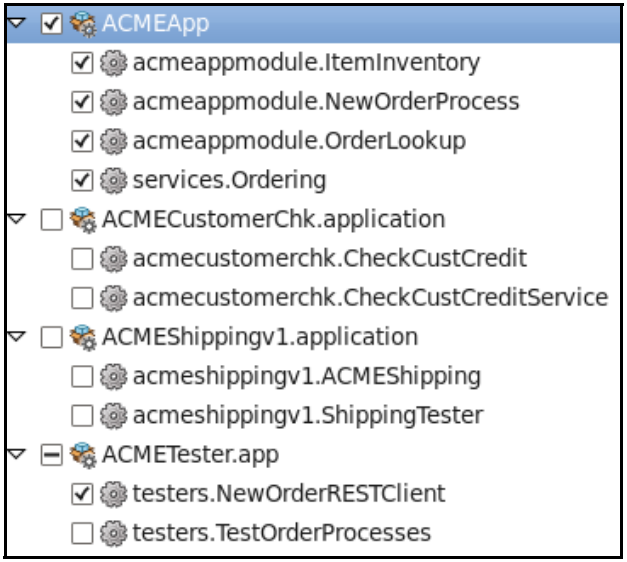
- For *CallNewOrder* > *Properties* > ***Input***
- Map `$post/item/NewOrderInfo` as shown below:
  - Choose option to **Make A Copy Of**



  - **Save** changes and verify that there are no problems/errors

6. Update the ACMETest configuration to include ACMEApp and the REST client process.

- To update ACMETest configuration in **Run** > **Debug Configurations...** > **ACMETest**
  - Also select *ACEMApp* as shown
  - Select the *NewOrderRESTClient* process



7. Run the ACMETest configuration and confirm that order lookup works successfully.

- Launch the debugger and run the **ACMETest** configuration
  - Confirm that the **ACMEApp** and **ACMETester** applications are started up
- Run the Java Program to send a JMS message, which triggers the NewOrder Service.
  - In Project Explorer, browse to ***JMSNewOrder > src > com.tibco.tibedu.jmstest***
  - Right-click **NewOrderACME.java** and choose *Run As* > ***Java Application***
  - Observe the output displayed in the BusinessWorks application console
- The BusinessWorks Application displays the error shown below:

```
Data=<tns:jdbcUpdateActivityInput xmlns:tns="http://www.tibco.com/namespaces/tnt
/plugins/jdbc+73fdf35b-5af1-42f8-96d5-bb9d565c845b+input" xmlns:ns="http://www.t
ibco.com/schemas/testschemas/Schema.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSc
hema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><InQty /><InStatus>I
N STOCK</InStatus><InSKU>ITEM1</InSKU></tns:jdbcUpdateActivityInput>
]
Caused by: com.tibco.bx.core.faults.BxException: Activity input data validation
error. The input data for activity [UpdateInventory] in process [acmeappmodule.I
temInventory], module [ACMEAppModule] does not conform to the activity input sch
ema. The initial value '' is not valid with respect to the simple type definitio
n '{http://www.w3.org/2001/XMLSchema}int'.   Schema=
schema-element({http://www.tibco.com/namespaces/tnt/plugins/jdbc+73fdf35b-5af1-4
2f8-96d5-bb9d565c845b+input}jdbcUpdateActivityInput)  {http://genxdm.org/typed/l
ocal-types}type-3
```

Note: The NewOrderACME.java program uses two ItemSKUs - ITEM1 and ITEM2. These items do NOT exist in the ACME_ItemDetail table, so the updateInventory activity fails. As a result, the ItemInventory process throws an error, which is propagated up to the NewOrder Process. Due to this, the REST client fails as well.

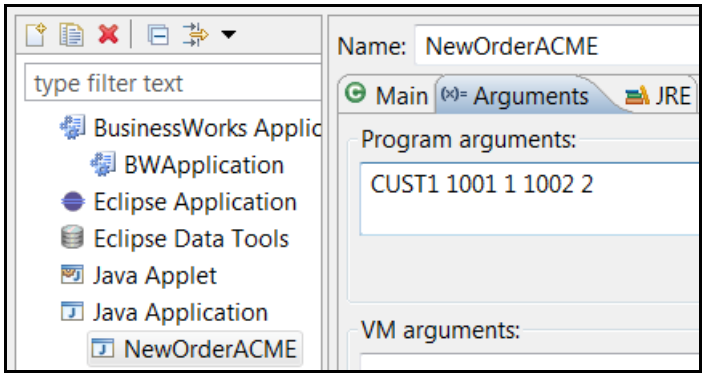
To fix the error, the Java program must submit existing ItemSKUs along with the New Order Request. You can handle errors in ItemInventory, so that the process does not give runtime errors This is an exercise left to you, for later.

- **Stop** running the BusinessWorks Application

8. Update the Java program run configuration to submit valid ItemSKUs.

- From the top menu in Business Studio, choose *Run* > ***Run Configurations.***
  - Browse to *Java Application* > ***NewOrderACME***
  - Now choose the ***Arguments*** tab and update Program Arguments to the following

    `CUST1 1001 1 1002 2`



  - Click **Apply** then click **Run**
- Verify that the program runs and terminates
  - Confirm that the appropriate ItemSKUs are submitted

```
destination is: ACME.NewOrder
Sending TextMessage={ Header={ JMSMessageID={null} JMSDestination=
 JMSDeliveryMode={PERSISTENT} JMSRedelivered={false} JMSCorrelatio
} JMSTimestamp={0} JMSDeliveryTime={0} JMSExpiration={0} JMSPriori
Text={<?xml version='1.0' encoding='UTF-8'?><ns0:NewOrderInfo xmln
com/schemas/testschemas/Schema.xsd'> <ns0:CustomerID>CUST1</ns0:Cu
HIGH</ns0:Priority> <ns0:OrderDate>2014-08-08T11:11:12.484-08:00</
tems> <ns0:ItemSKU>1001</ns0:ItemSKU> <ns0:Quantity>1</ns0:Quantit
ineItems> <ns0:ItemSKU>1002</ns0:ItemSKU> <ns0:Quantity>2</ns0:Qua
/ns0:NewOrderInfo>} }
```

9. Run the BusinessWorks Application and verify it runs successfully.

- Launch the debugger and run the **ACMETest** configuration
- Confirm that the **ACMEApp** and **ACMETester** applications are started up
- Verify that the output messages shown below are displayed:

```
ster.LogJSON -  JSON Order: {"CustomerID":"CUST1","Priority":"HIGH","OrderDate":
"2014-08-08T11:11:12.484-08:00","LineItems":[{"ItemSKU":"1001","Quantity":1},{"I
temSKU":"1002","Quantity":2}]}
18:10:47.235 INFO  [bwEngThread:In-Memory Process Worker-3] c.t.b.p.g.Log.ACMEAp
pModule.Log -  Received New Order Message for Customer ID: CUST1
18:10:48.353 INFO  [bwEngThread:In-Memory Process Worker-8] c.t.b.p.g.L.A.LogInv
entory -  Inventory Updated for Customer ID: CUST1
18:10:48.395 INFO  [bwEngThread:In-Memory Process Worker-1] c.t.b.p.g.L.ACMEAppM
odule.Inventory - Item Inventory Processing Completed
18:10:48.648 INFO  [bwEngThread:In-Memory Process Worker-3] c.t.b.p.g.Log.ACMETe
ster.LogResponse -  New Order Response, Status Code :200 Content: {"OrderID":"NO
T AVAILABLE","CustomerID":"NA","Status":"NA","OrderAmount":0.0,"HandlingAmount":
0.0,"TotalAmount":0.0,"DateChanged":"2017-01-03T18:10:48.453-08:00"}
```

ACMEAp

ACMEApM

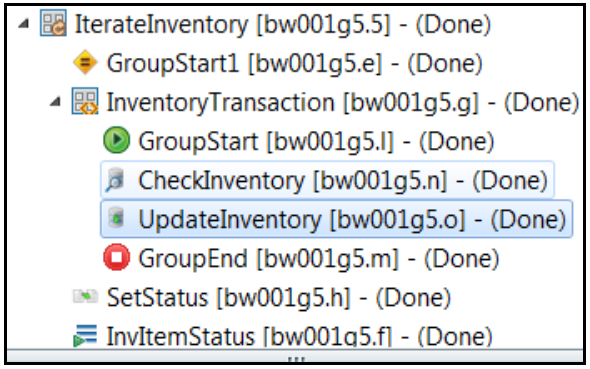
En algunas ejecuciones
este paso lanzaba un error
de uso del puerto 9797.
Lo que hice fue depurar nuevamente
la configuración ACMETest, luego
la clase JAVA y funcionó.

Alternativamente usé el comando:

```
fuser -k 9797/tcp
```

Para cerrar el puerto 9797

- In the Debugger, verify that the ItemInventory process has been completed



- **Stop** the BusinessWorks Application

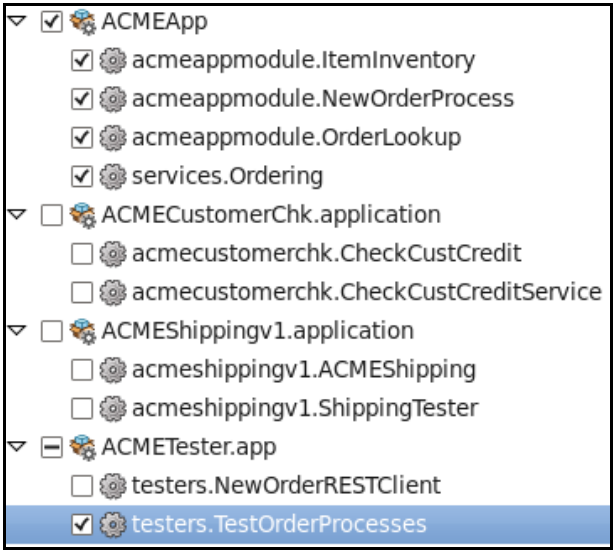
10. Update the configuration and profile to test OrderLookUp process.

- Update profile to invoke Lookup Order and test again
  - In the **ACMETester.app** Properties view, switch default profile to **Test**

ACMETester.app > Package Unit > Properties

Properties	[Test]	default
Application		
ACMETester		
testingOption	LookupOrder	NewOrder

- **Save** the changes
- To update ACMETest configuration in **Run > Debug Configurations... > ACMETest**
  - Also select *ACEMApp* as shown
  - Select the *TestOrder* process



  - Click **Apply** and **Debug**

11. Observe the Console window and confirm that order lookup executes successfully.

- Confirm that the application has started successfully

```
18:18:24.091 INFO  [Thread-24] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK
-300006: Started BW Application [ACMETester.app:1.0]
18:18:24.119 INFO  [Framework Event Dispatcher: Equinox Container: 90920d05-24d2
-0016-1c0e-d37b06fed4f1] com.tibco.thor.frwk.Application - Started by BusinessSt
udio, ignoring .enabled settings.
18:18:24.119 INFO  [Framework Event Dispatcher: Equinox Container: 90920d05-24d2
-0016-1c0e-d37b06fed4f1] com.tibco.thor.frwk.Application - Started by BusinessSt
udio, ignoring .enabled settings.
18:18:24.744 INFO  [bwEngThread:In-Memory Process Worker-1] c.t.b.p.g.Log.ACMETe
ster.Log - ******** Started Timer Application at: 6:18:24 PM********
18:18:35.101 INFO  [bwEngThread:In-Memory Process Worker-3] c.t.b.p.g.Log.ACMEAp
pModule.Log - Received Lookup request for OrderID: TEST1
```

- The *Lookup Request* is executed and this proves that the application is running

Tip: If the Lookup order is not called, it indicates that the application has not started up successfully. Increase the sleep time to 10 seconds (initervalInMilleSec = 10,000)

12. Export your solution as ZIP file.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File: `/home/tibco/bw6edu/dev/exports/MBW6I_Inventory.zip`
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created

Notes

Notes

Unit 10 Embedding Java Code

This unit examines how external Java code can be imported into a BusinessWorks project. It also examines how product bundles can be used to resolve classpath references for Java code.

Topics

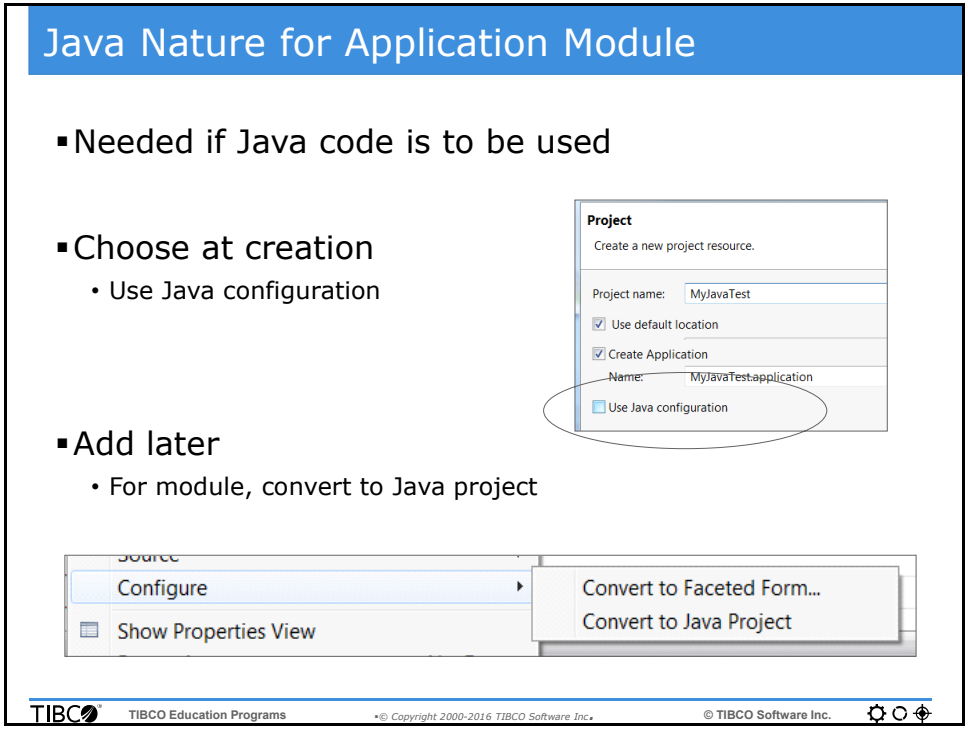
- *Objectives,* page 298
- *Java Nature for Application Module,* page 299
- *Confirming Java Nature,* page 300
- *Adding External Java Code to Module,* page 301
- *Using Product Bundles,* page 302
- *Using the Mapper,* page 303
- *Variables in Mapper,* page 304
- *Inline Schema Editor,* page 305
- *Review,* page 308
- *Lab Overview,* page 309

Objectives

- Knowledge
  - Explain how to add external Java code in BusinessWorks
  - Discuss how product bundles can be used
  - Explain the use of mapper activity
- Skills
  - Add Java code to your solution
  - Add mapper activity to generate OrderID

TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. • © TIBCO Software Inc.

Objectives



Java Nature for Application Module

External Java code can be added to a BusinessWorks application using a few options.

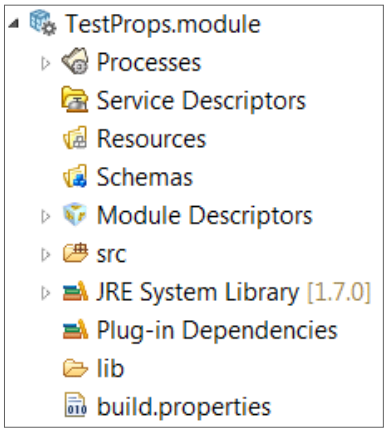
Any Java code added to an application is automatically compiled, and any errors are displayed in the problems tab.

To enable an Application to work with Java code, it must be converted to "Java Nature". An application with Java nature has the necessary Java settings such as system libraries and CLASSPATH settings, which allows for editing, compiling and running/testing Java code.

An application can be created with Java nature at creation time, by checking the Use Java Configuration option. It can also be converted later by using the option shown above.

Confirming Java Nature

- Special Folders created
  - src
  - lib
- Java libraries added



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Confirming Java Nature

When an application module is updated to have Java nature, it automatically adds a few objects. The src, JRE System Library and lib folders are added.

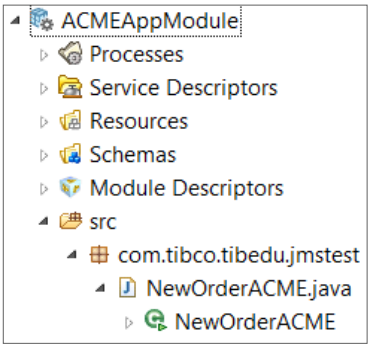
The src folder is used to hold your Java code, in package format.

The JRE System library folder contains the Jar files for your core Java runtime environment.

The lib folder is used to store any other required external Jar files.

Adding External Java Code to Module

- Purpose
  - Consolidate resources for projects
    - Tester programs
- Code not bundled in EAR file
- How to use:
  - Add Code in SRC folder
  - Class definition, package



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

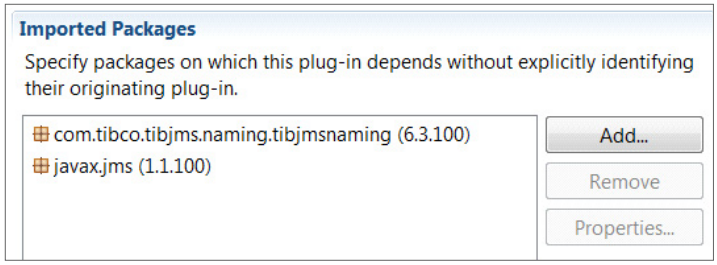
Adding External Java Code to Module

To add your Java code, browse to the src folder, and define the packages/classes you need.

This Java code is not bundled with the EAR file during deployment. So use this feature only when you need tester programs during application development.

Using Product Bundles

- Use product packages instead of external JARs
  - Many available!
- Improved portability for your code



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Using Product Bundles

If your tester Java program needs to use a third party Jar file (for example JMS jar files), it is a good idea to check the availability of the Jar file in the TIBCO Product installation itself.

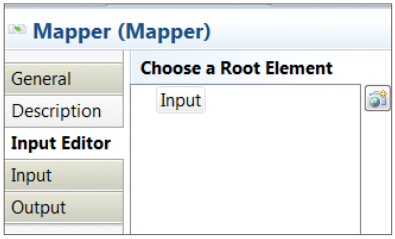
TIBCO BusinessWorks bundles well known open source Jar files as part of its installation for messaging, security, logging and other common functions.

Using these Jar files allows portability of your code, since you can simple export this workspace and import into a new workspace without any need for packing any extra Jar files.

Using the Mapper

- Mapper function
  - Add new process variables
  - All variables available downstream

- Using a mapper
  - Add variable(s)
    - Use Input Editor
  - Set values
    - Use Input tab



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Using the Mapper

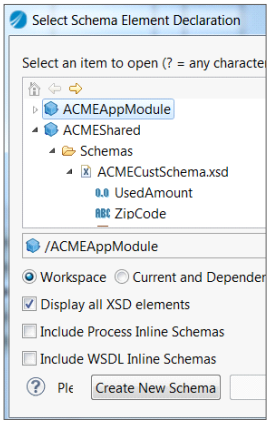
Mapper: Adds new process variables to the process. You can map the data values from the current list of process variables to the elements of the variable with the Mapper activity.

Use the input editor tab to add different types of variables to the process. Set the values for these variables in the input tab.

These variables are now available downstream in the process, to ALL process activities.

Variables in Mapper

- Declare variable(s)
  - Inline schema
  - Primitive
  - Complex element
- Scope
  - Workspace
  - Current, dependent modules
- Can use existing schema
  - Primitive or complex elements

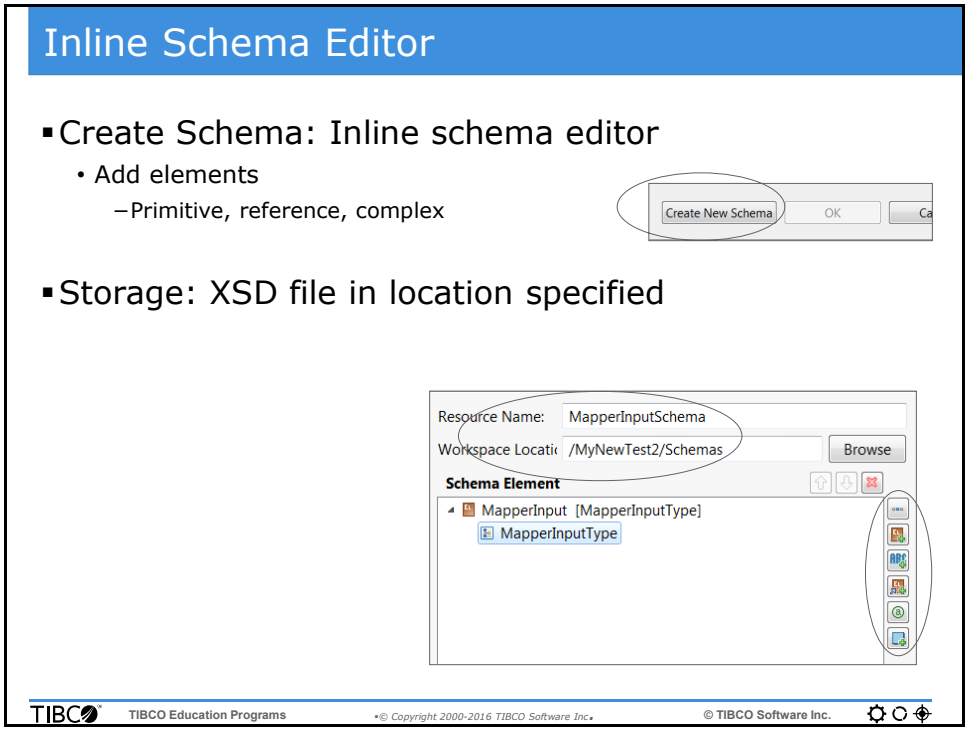


TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Variables in Mapper

Input Editor: Defines the structure of the process variable to add to the process. You can use a simple datatype, or you can define a group of data elements on this tab. You can also reference XML schema stored in the project. The data specified here becomes the input and output schema of the Mapper activity. This data is then available to other activities in the process.

This variable can be an inline schema, primitive element, or a complex element. An inline schema editor is available and it allows you to define a brand new schema within the mapper itself. The schema is stored as an XSD file in the project.



Inline Schema Editor

The inline schema editor is opened when you click on the create new schema button in input editor of the mapper activity (or any other activity).

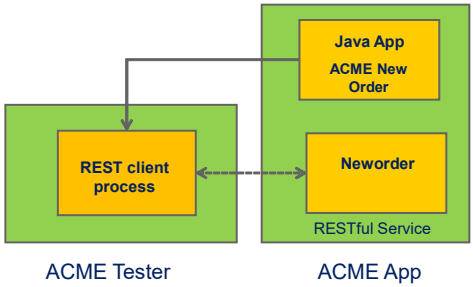
This editor is used to define ad-hoc schemas as they are needed. You can add elements of different types by clicking the icon on the right as shown.

You can also add attributes and the *any* element type (element without a type, which can be coerced later).

The schema itself is stored as an XSD file, using the name and location specified by you as shown.

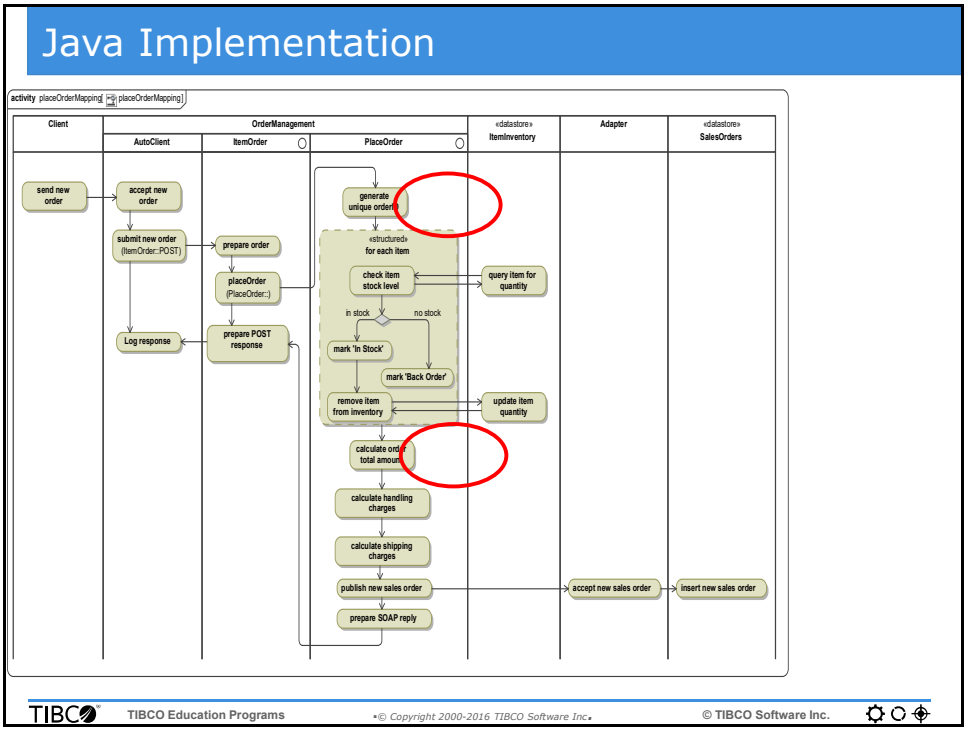
Business Case Implementation

- Add Java project to ACME App
  - Consolidate for easy management



TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Business Case Implementation



Java Implementation

Review

- Application must have Java nature to support external Java code
- Use product bundles for common libraries
- Embedded code not bundled with EAR
- Mapper activity is used to add new variables

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Import Java Program
  - Configure Build Path by using product bundles
- Use mapper to define order ID, total amount
  - Define inline schema
- Test Results

TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab J Using Java Code

Overview

In this lab you will examine the various ways in which Java code can be integrated into your BusinessWorks solution. You will first import the JMS Java code into your application so you can close the Java project you had created earlier.

Objectives

- *Importing Existing Code, page 312*
- *Get OrderID and Price,* page 316

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous lab has been completed

Directions

- Complete the exercises that follow

Lab J - Exercise 1: Importing Existing Code

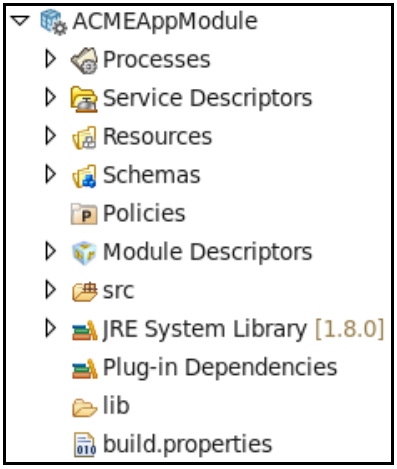
Overview

In this exercise you will import the existing JMS code into your application.

Steps

1. Update your application to have a Java nature.

- Launch Business Studio for workspace: **`/home/tibco/bw6edu/workspace`**
- In Project Explorer, browse to **ACMEAppModule**
  - Right-click and choose *Configure* > ***Convert to Java Project***
- Expand the **ACMEAppModule** folder
  - Notice new folders **src** and **lib** have been added

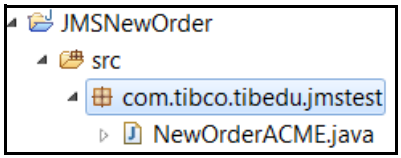


  - The JRE system libraries are also included
  - In addition, there is a new Plug-in Dependencies folder as well

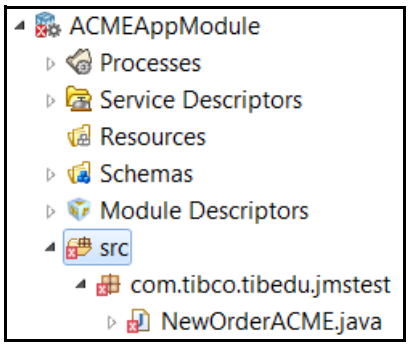
Tip: Do NOT delete any folders/directories added by BusinessStudio at any point during development. Doing so will make the development unstable and lead to unpredicatble results.

2. Copy and paste NewOrderACME java program to your application.

- Expand the Java Project created earlier - *JMSNewOrder* > ***src***
  - Right-click on **com.tibco.tibedu.jmstest** and choose **Copy**

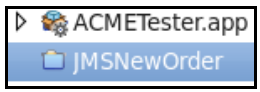


- Browse to *ACMEAppModule* > ***src***
  - Right-click on **src** folder and click **Paste**
  - Verify that the package and java program have been copied over
  - There are errors in the code, to be resolved soon



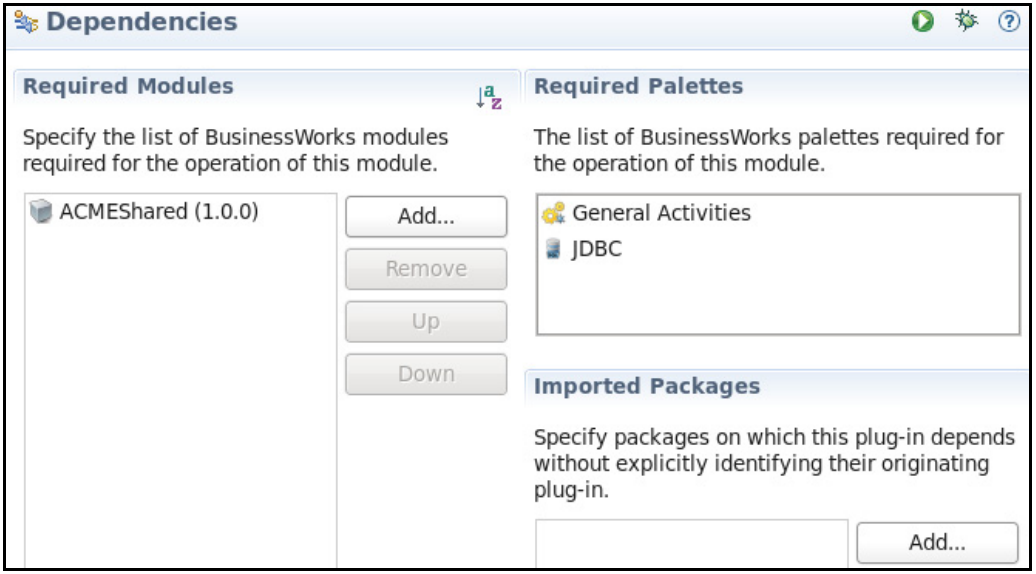
3. Close the original java project.

- Right-click on **JMSNewOrder** project
  - Choose **Close Project**
  - The project icon changes, indicating it is closed

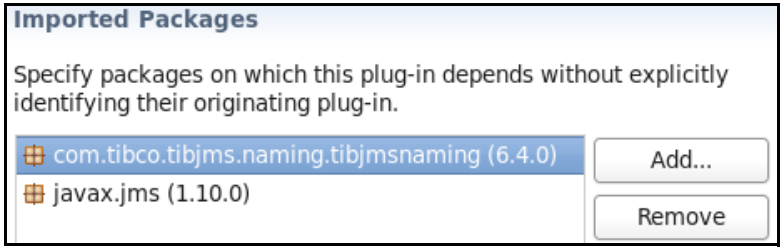


4. Resolve errors with the imported Java code by adding required dependencies.

- Browse to *ACMEAppModule* > ***Module Descriptors***
  - Double-click **Dependencies** to open details



- In **Imported Packages**, click **Add...**
  - In the pop-up window, type **`javax.j`** the list of packages appears
    - Select **javax.jms (1.10.0)** and click **OK**
  - Similarly, add **`com.tibco.tibjms.naming.tibjmsnaming (6.4.0)`**



- **Save** your changes, and all errors with your Java code should be resolved
- **Close** module properties view

5. Update the project settings and run the Java code.

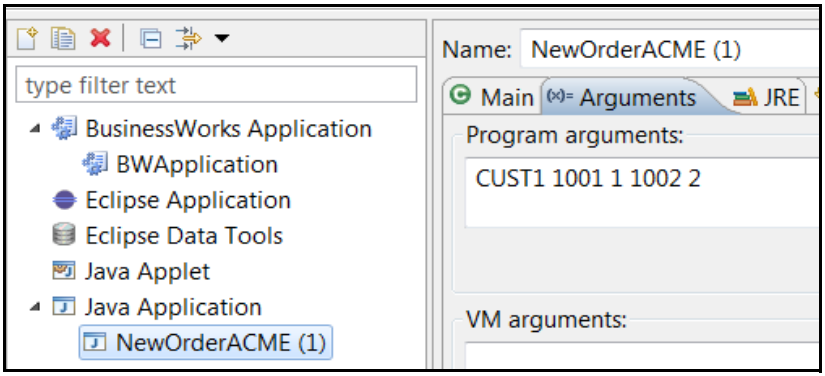
- Attempt to run the Java code that you just imported
  - Right-click ***src*** > *com.tibco.tibedu.jmstest* > ***NewOrderACME.java***
  - Select *Run As* > ***Java Application***
  - The program runs, but does not send any messages, as it is missing arguments

```
Usage: java NewOrderACME <CustID> <SKU1> <Qy1> <SKU2> <Qty2>
```

Tip: You have to run once so that a run configuration is automatically created for you.

- Provide arguments to the program
  - From the top menu, choose ***Run > Run Configurations...***
  - Browse to *Java Application* > ***NewOrderACME(1)***
  - Choose *Arguments* tab and enter these arguments

    CUST1 1001 1 1002 2



  - **Apply** changes and click **Run** to test the application again
- Verify it runs by observing the console window

```
destination is: ACME.NewOrder
Sending TextMessage={ Header={ JMSMessageID={null} J
 JMSDeliveryMode={PERSISTENT} JMSRedelivered={false}
} JMSTimestamp={0} JMSExpiration={0} JMSPriority={4}
'1.0' encoding='UTF-8'?><ns0:NewOrderInfo xmlns:ns0=
mas/Schema.xsd'> <ns0:CustomerID>CUST1</ns0:Customer
```

Lab J - Exercise 2: Get OrderID and Price

Overview

In this exercise you will update the new order process to generate a unique OrderID and calculate a total price for the order. These values are used to calculate handling charges and delivery details.

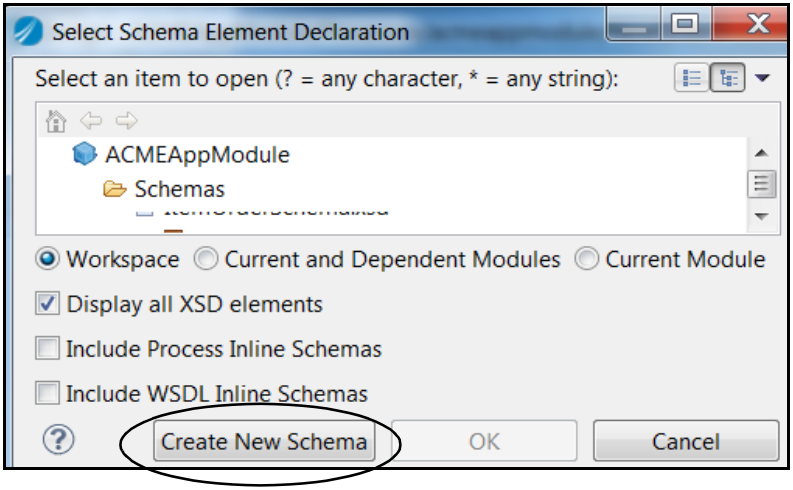
Steps

1. In Business Studio, open the new order process.

- Browse to *ACMEAppModule* > *Processes* > ***acmeappmodule***
- Double-click **NewOrderProcess.bwp** to open it

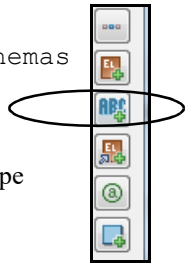
2. Add a mapper activity to hold an OrderID schema element.

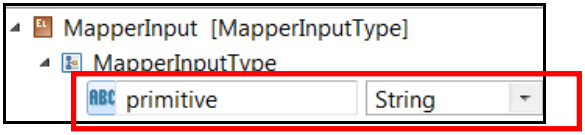
- Drag a **Mapper** activity and place it between **placeOrderOpIn** and **Log** activity
- Configure it as follows:
  - Name: **`GenerateOrderID`**
  - *Input Editor* - **Select Input Element** and choose **Create New Schema**



3. Define the new schema to hold the generated OrderID.

- Resource Name: **`OrderIDSchema`**
  - Workspace Location: `/ACMEAppModule/Schemas`
  - Select **MapperInputType**
    - Click **Add Primitive Element** (circled)
    - This adds an element below MapperInputType
    - Expand it to locate the element

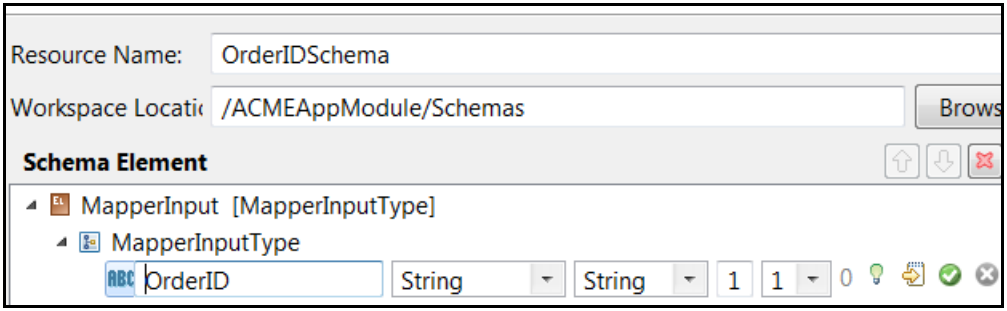




Hacer doble click para editar el elemento.

  - Change Element name to **`OrderID`**

    (Hit enter to apply changes)

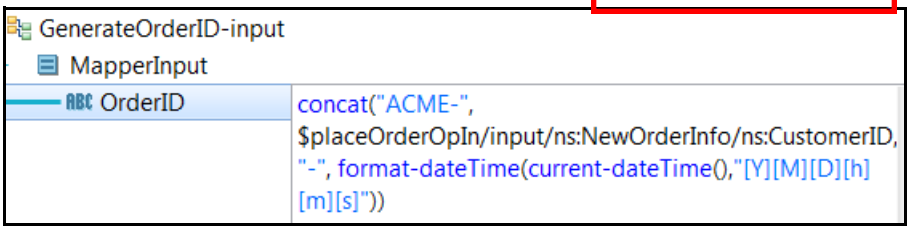


- Select the **MapperInput [MapperInputType]**
  - Now you can click **OK**
- **Save** changes

4. Complete the mapper activity to generate a unique Order after the request is received.

- For **Generate OrderID -** Properties
- Input for OrderID:
  `concat("ACME-",$placeOrderOpIn/input/ns:NewOrderInfo/ns:CustomerID, "-",`
  `format-dateTime(current-dateTime(),"[Y][M][D][h][m][s]"))`

Escribirlo

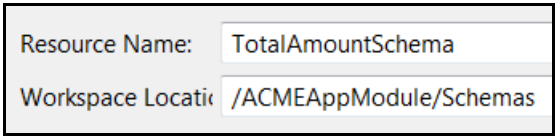


Tip: The OrderID is derived from the date time combination, since it is most likely to be unique. The CustomerID is added for easy reference later.

- **Save** changes

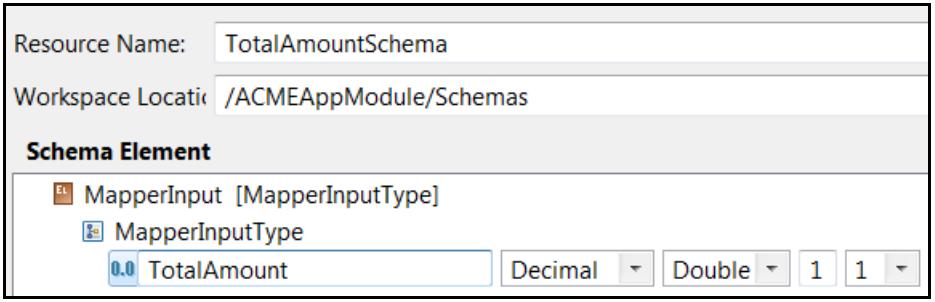
5. Add another mapper activity to calculate the TotalAmount after update inventory.

- Click **Mapper** activity and place it between **updateItems** and **Inventory** activity
- Configure it as follows:
  - Name: **`CalcTotalAmount`**
  - *Input Editor* - Select Input Element
    - Choose **Create New Schema**
    - Resource Name: **`TotalAmountSchema`**
    - Workspace Location: `/ACMEAppModule/Schemas`



6. Add the totalAmount element as a decimal type.

- Choose MapperInputType
  - Click **Add Primitive Element**
    - Change Element name to **`TotalAmount`**, Type: **Decimal - Double**
    - Hit **Enter** to apply the changes



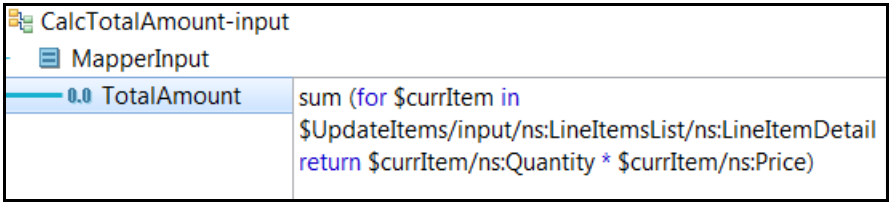
- Browse up to **MapperInput [MapperInputType]**
- Click **OK** and **Save** changes

7. Calculate the total amount based on prices returned by update inventory sub-process.

- In **CalcTotalAmount -** *Properties* **>** ***Input***
- Set TotalAmount using this formula:

```
 sum (for $currItem in
$UpdateItems/input/ns:LineItemsList/ns:LineItemDetail
return $currItem/ns:Quantity * $currItem/ns:Price)
```

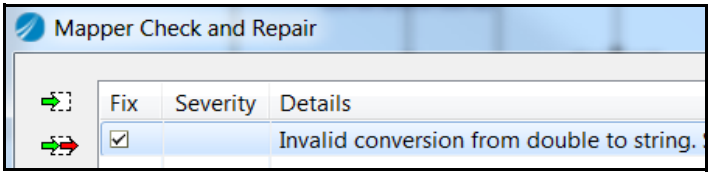
  - Use mapper check and ~~repair~~ to resolve any namespace mismatch issues Si fuera el caso.



Analysis: $currItem is a local variable you are defining here to store the result (sum) of the for loop; you therefore cannot make use of the drag and drop functionality of the XPath Editor for this element.

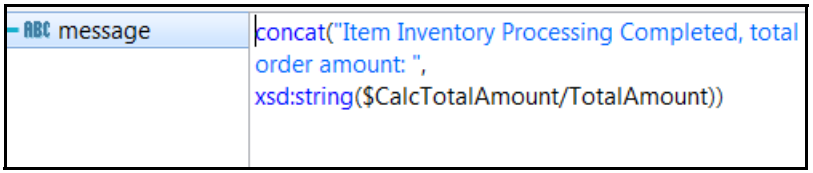
8. Update the Log activity to record the total order amount.

- Update the **Inventory** log activity as shown below:
- Update **Message**: **`concat("Item Inventory Processing Completed, total order amount: ", xsd:string($CalcTotalAmount/TotalAmount))`**
- Use Mapper check and repair to add the string conversion function



Hacer esto si fuera el caso.

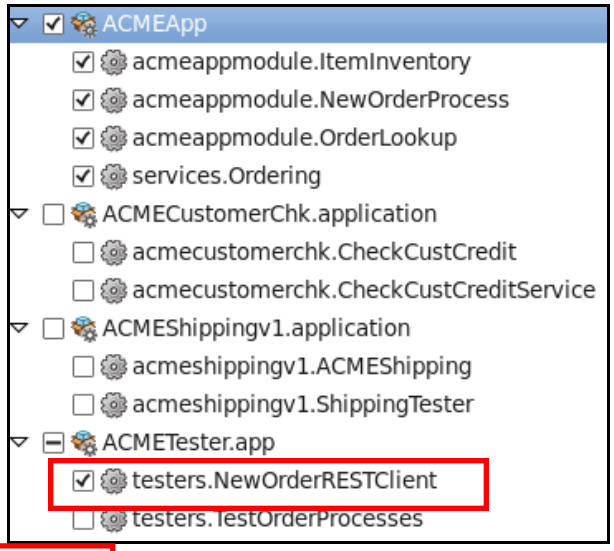
- Final result looks as shown below:



IMPORTANTE: Antes de ejecutar este paso, verificar en ACMETester.app > Package Unit > Properties > ACMETester.app el profile "default" sea el default con testingOption=NewOrder. Si no, genera error la ejecución.

9. Update the ACMETest configuration to include ACMEApp and the REST client process.

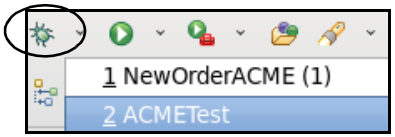
- To update ACMETest configuration in **Run > Debug Configurations... > ACMETest**
  - Also select ~~*ACEMApp*~~ ACMEApp as shown
  - Select the *NewOrderRESTClient* process



* Click Apply y Close

10. Run the solution to verify that OrderID and TotalAmount are generated properly.

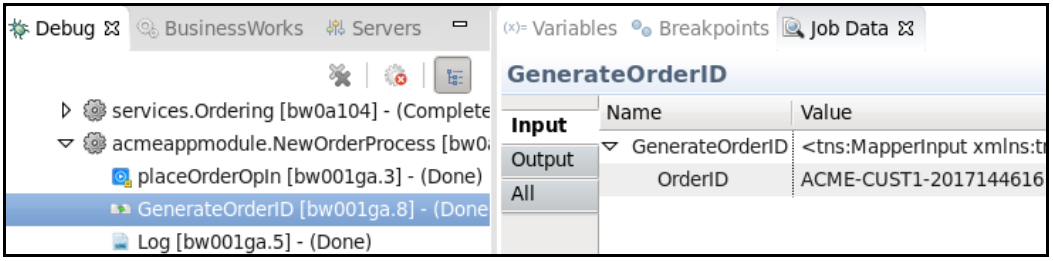
- In the top menu of Business Studio
  - Click on **Debug** icon (shown below) and select **ACMETest**



- After the BusinessWorks Application starts,
  - Click Debug icon and select **NewOrderAcme(1)**
- Confirm that both are executed successfully
  - Toggle between two consoles using the icon in the debug console view

11. Examine the details for the generated OrderID and Total Amount.

- In the debug details window (top left), locate the **NewOrderProcess** and examine its execution details
- Click on **GenerateOrderID** and view **Job Data**
  - Confirm that **OrderID** is generated properly



- Click on UpdateItems and examine the **Output**

LineItemDetail [1]	<ns:LineItemDetail xmlns:n
ItemSKU	1001
ItemDesc	Bluetooth Headset
Status	IN STOCk
Quantity	1
Price	4.999E1
DateChanged	2016-12-23T11:20:33-08:00
LineItemDetail [2]	<ns:LineItemDetail xmlns:n
ItemSKU	1002
ItemDesc	Bluetooth Speakers
Status	IN STOCk
Quantity	2
Price	7.999E1
DateChanged	2016-12-23T11:20:34-08:00

- First item (1001) is $49.99, second item (1002) is $79.99
  - Total Amount should be $49.99 + (2 * 79.99) = **209.97**

```
entory -  Inventory Updated for Customer ID: CUST1
16:21:02.767 INFO  [bwEngThread:In-Memory Process Worker-1] c.t.b.p.g.L.ACMEAppM
odule.Inventory - Item Inventory Processing Completed, total order amount: 209.9
7
```

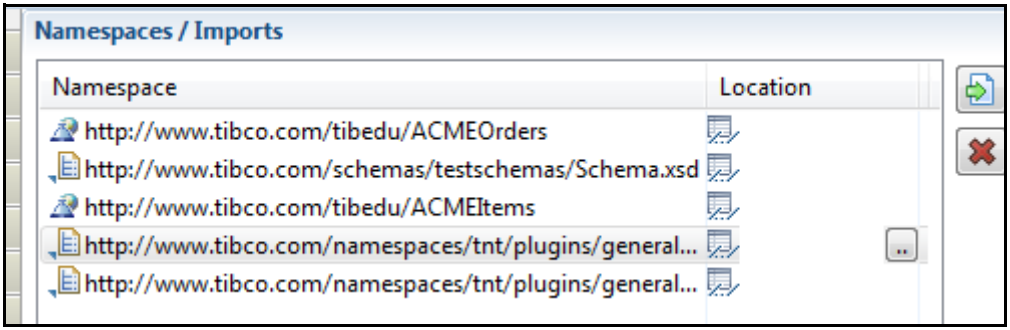
- **Stop** the tester when done
- **Switch** back to Design perspective

12. Export your solution as a ZIP file and shut down Business Studio.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File: `/home/tibco/bw6edu/dev/exports/`**`devUsingJava.zip`**
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created
- **Exit** Business Studio

Troubleshooting

- ACMETester error: returns a 4XX status code (URL not found)
  Launch OSGi console, and use **LA** command to determine which application failed
  Examine the details for the failure
  If the failure indicates a "Schema Not found Error"
  Go to that process indicated in the error, in process properties, locate dependencies
  Refresh all links to the dependencies, browse and select **locations**
  Example namespace dependencies screen shown below:



Notes

Unit 11 Building Plug-in Projects

This unit discusses the standard and OSGI java code.

Topics

- *Objectives,* page 326
- *Java Code: The OSGI Way,* page 327
- *OSGI Activator Classes,* page 328
- *Using a Plug-In Project,* page 329
- *Referencing a Plug-in Project,* page 330
- *Using Non-OSGI Java Code,* page 331
- *Business Case Implementation,* page 333
- *Java Plugins,* page 334

Objectives

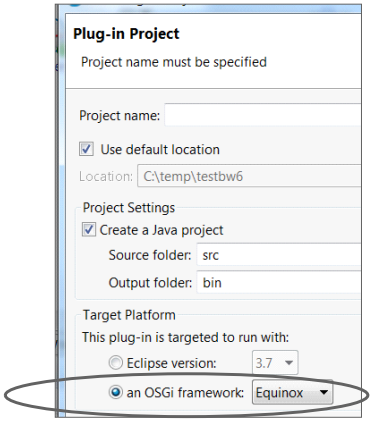
- Knowledge
  - Differentiate between standard and OSGi Java code
  - Examine various ways to invoke java methods
- Skills
  - Update process to handle external JARs
  - Define a plug-in project

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Objectives

Java Code: The OSGi Way

- Recommended for BusinessWorks 6
- How:
  - Define a Plug-in project
- Effect
  - OSGi-fied Java code created



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

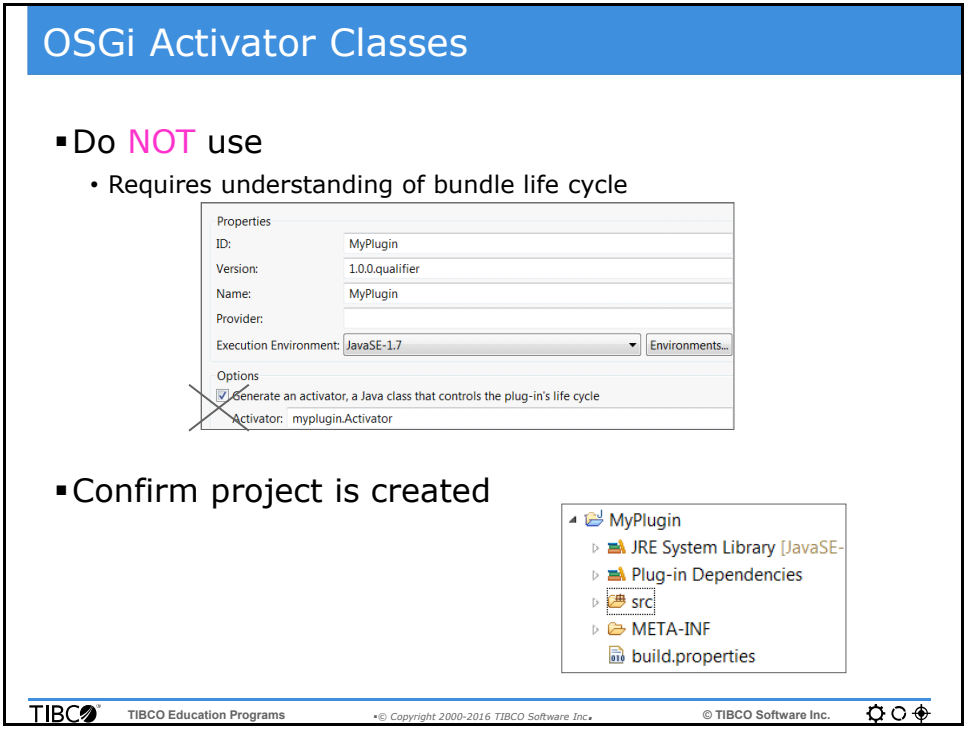
Java Code: The OSGI Way

ActiveMatrix BusinessWorks applications can also contain OSGi bundles that do not contain ActiveMatrix BusinessWorks artifacts. For example, you can create an application that contains a Java OSGi bundle, which is also referred to as a Java module.

There are two options for using Java code with BusinessWorks applications.

The recommended way is to convert Java code into OSGi bundles and then deploy it alongside the BusinessWorks application. To create an OSGi bundle, first define a plug-in project which can run with the Equinox OSGi framework as shown. This plug-in project can then be updated to include the Java code you want to use, including JAR files.

You can also use JAR files without converting them to OSGi bundles, and this is examined later.

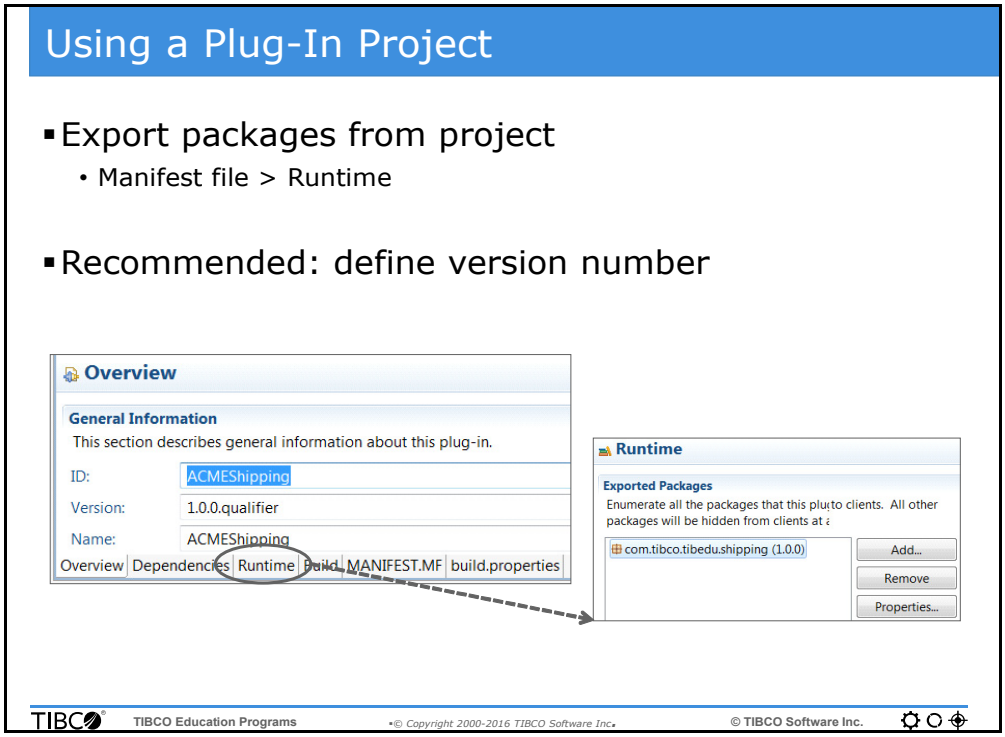


OSGI Activator Classes

When you define an OSGi bundle, you are presented with an option to define OSGi activator classes.

This class is instantiated whenever a bundle is activated. When an application is running in an AppNode, it may be activated and deactivated multiple times, depending on circumstances. So the activator can also be invoked multiple times as well.

Unless the developer understands the true lifecycle of an OSGi bundle, it is recommended that these classes not be used.



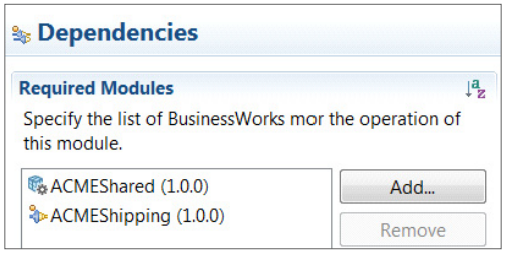
Using a Plug-In Project

Once a Java plug-in project is created, you must export its key packages using the option provided in Eclipse, shown above. Exposing the packages allows other OSGi bundles to connect to this OSGi bundle. This is also referred to as wiring.

When exposing a package for wiring by the OSGi bundle, you must provide a version number for your package. OSGI relies heavily on version numbers when packages are upgraded.

Referencing a Plug-in Project

- Update the Application Module
  - Add required dependencies
  - Only projects in the same workspace available



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

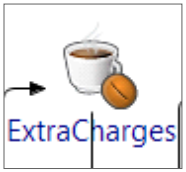
Referencing a Plug-in Project

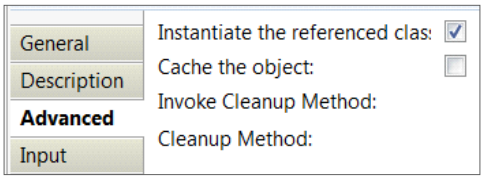
Once a Java project is configured with exported packages, it can be wired to a BusinessWorks application.

To do the wiring, add the package in the dependencies section of the application a shown. You can now reference the Java methods in your processes inside the module.

Using Non-OSGi Java Code

- Add JAR file to Application Module
  - Public static methods visible
  - Use Invoke Java activity
  - Instantiate referenced class!





TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Using Non-OSGI Java Code

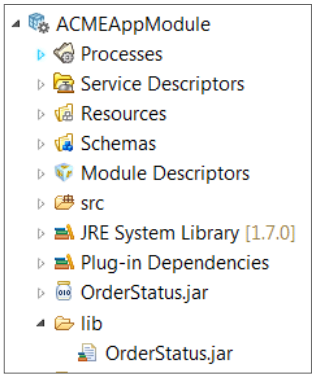
To use non-OSGi code in your application, you can add the Jar file to the application module. The Jar file is added to the Lib folder for an application with Java nature.

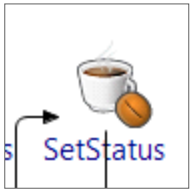
Once the Jar file is added, you can directly invoke any public static method in the Jar, by using the Java invoke activity.

You must instantiate any referenced class once, using the option shown above.

Using Non-OSGi Code

- Add JAR file to Module
  - LIB folder
- Using external code
  - Invoke Activity



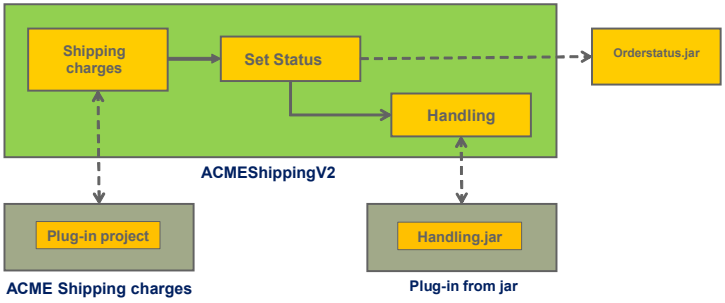


TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Using Non-OSGi Code

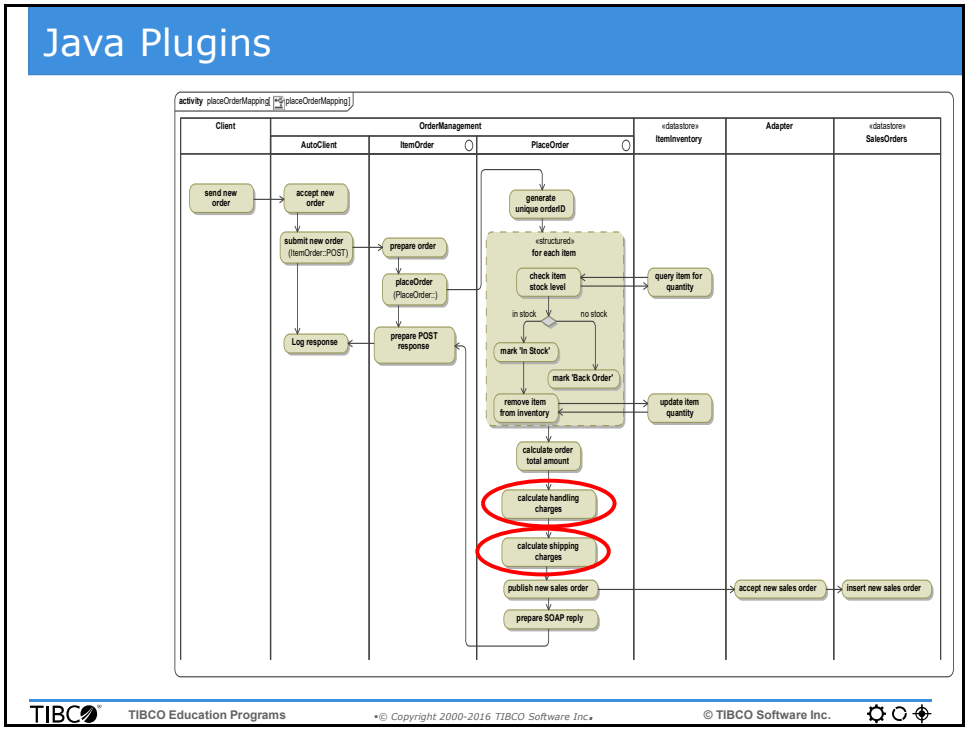
Business Case Implementation

- Update Shipping Application
  - Plug-in project from Java code
  - Plug-in project from Jar
  - Invoke Java method



TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Business Case Implementation



Java Plugins

Review

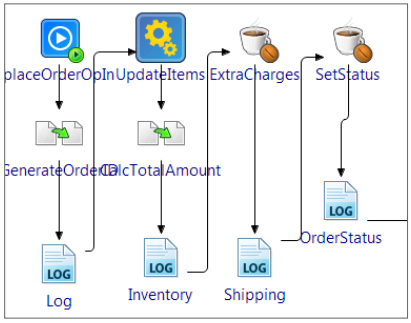
- Plug-in projects can be added
- Plug-in project must export packages
- Java invoke activity must instantiate plug-in class
- Non-OSGi Jars can be imported

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Use plug-in project
  - Create plug-in project
  - Implement shipping charges logic
  - Use Invoke activity
- Add external JAR
  - Handling charges
  - Test Solution



TIBCO Education Programs
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

Lab Overview

Notes

Notes

Lab K Plug-in Projects

Overview

In this lab you will continue using Java code by defining a plug-in project which allows you to use OSGi-fied Java code. You will also call a method in an external JAR file to demonstrate connectivity. You will use these features for calculating Shipping Charges and Delivery mode.

Objectives

- *Use Plug-In Project,* page 340
- *Call External Method,* page 349

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3
- Previous lab has been completed

Directions

- Complete the exercises that follow

Lab K - Exercise 1: Use Plug-In Project

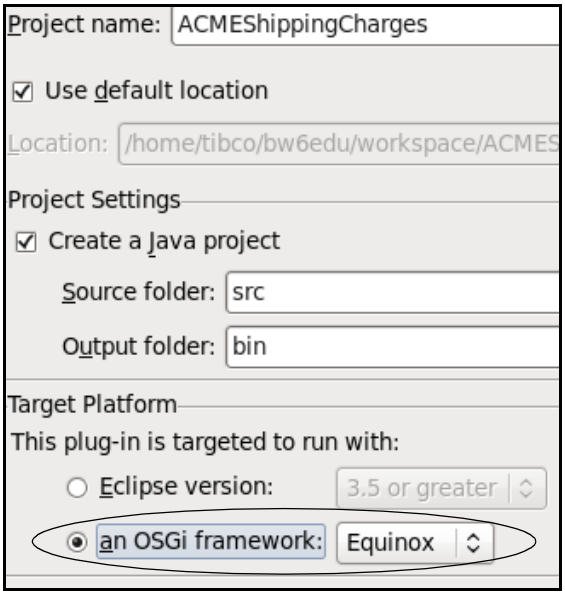
Overview

In this exercise define a new Plug-in project to convert your business logic for calculating handling charges in Java, into a OSGi bundle. Then, access this code using the Java Invoke activity.

Steps

1. Define a new plug-in project to handle business logic.

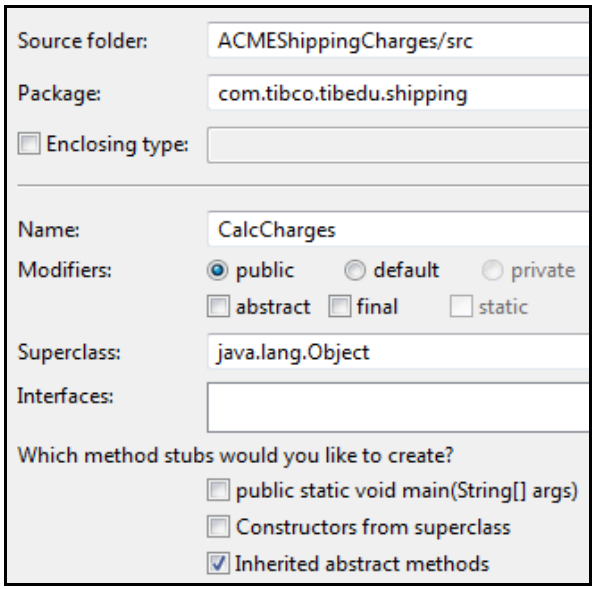
- From the top menu in Business Studio, choose *File* > *New* > ***Project***
  - Choose **Plug-in Project**, click **Next**
    - Project Name:**`ACMEShippingCharges`**
    - Use default location: **checked**
    - Create Java Project: **checked**
    - Plug-in targeted to run with:
      an OSGi framework: Equinox



- Click **Next**
- **Uncheck:** Generate an activator, A Java class that controls the plug-in's life cycle
- Click **Finish**
- When asked to open associated perspective, click **NO**

2. Define a Java class in your plug-in project.

- The Plug-in project overview window is opened, **close** it
- In Project Explorer, browse to *ACMEShippingCharges* > ***src***
  - Right-click and choose *New* > *Class*
  - Package: **`com.tibco.tibedu.shipping`**
  - Name: **`CalcCharges`**



  - Keep other values as shown and click **Finish**

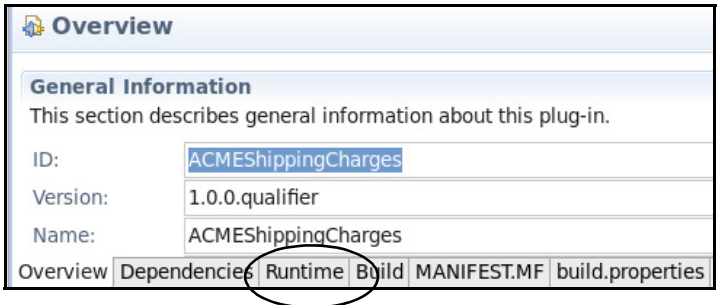
3. Copy the pre-built code to complete the Java class definition.

- In File Explorer (bottom left) screen, browse to the **code** sub-folder
  - Drag the **ACMECharges.txt** file next to the open Java code window, to open it
- **Copy** all the code from *ACMECharges.txt* file
  - **Paste** the code into **CalcCharges.java** to completely replace it
- Confirm that there are no errors and **save** your changes
- **Close** the text file and Java program view

4. Update the plug-in project to expose the newly created package to other clients.

- Browse to *ACMEShippingCharges* > *META-INF* > ***MANIFEST.MF***
  - Double-click MANIFEST.MF to open it

    (It presents similar tabs to those presented by *Module Descriptors*)



- Click on *Runtime* tab (circled above)
  - Under **Exported Packages** section, click **Add...**
    - Select **com.tibco.tibedu.shipping** and click **OK**
  - Now click **Properties...**
    - Type `1.0` and click **OK**

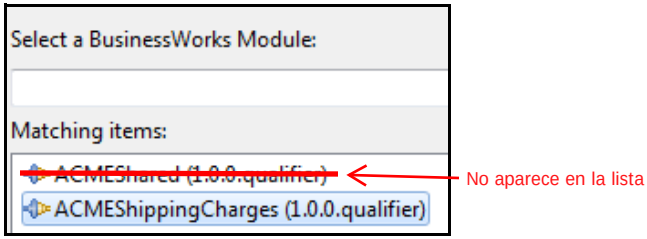
Tip: When using exporting bundles, it is always a good idea to specify version numbers, so that other bundles can adjust their compliance needs.

- **Save** changes and close the MANIFEST view

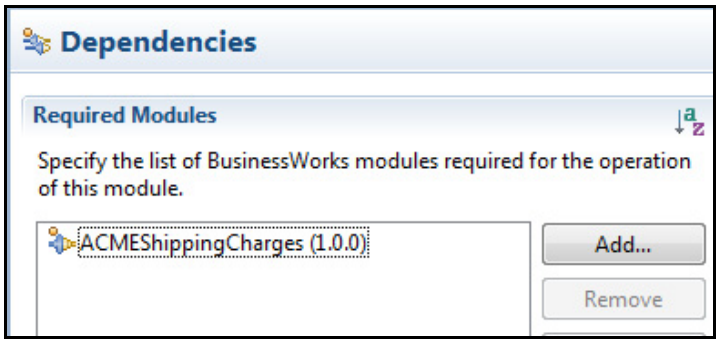
> ***Tip:*** You have just define a OSGi bundle that can be accessed by another OSGi bundle module. The next step is to access it from a BusinessWorks module.

5. Configure *ACMEShippingv1* module with Java nature and update the dependencies section.

- Right-click *ACMEShippingv1, choose Configure* > ***Convert to Java Project***
  - Verify that the **src** and **lib** folder are added to the project
- Browse to *ACMEShippingv1* > *Module Descriptors* > ***Dependencies***
  - Double-click **Dependencies** to open it
- Click *Dependencies* tab, and locate the **Required Modules** section
- Click **Add...** and select the ACMEShippingCharges ~~**ACMEShipping**~~ module as shown below:



- **Save** your changes



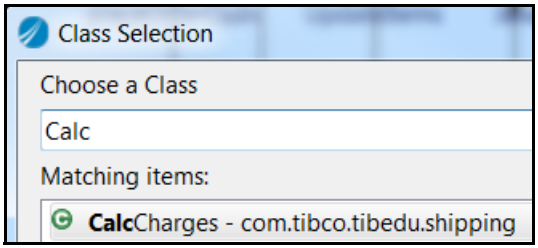
- Close the view

6. Add a Java Invoke activity to calculate shipping charges.

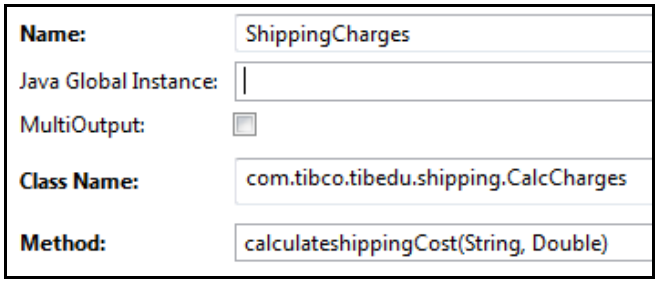
- Browse to the *ACMEShippingv1* > *Processes* > ***ACMEShipping***
- Add a **Java Invoke** after the log activity
- Configure the Activity, in *Properties* > ***General*** tab:
  - Name: **`ShippingCharges`**

7. Browse and locate the Shipping class and method.

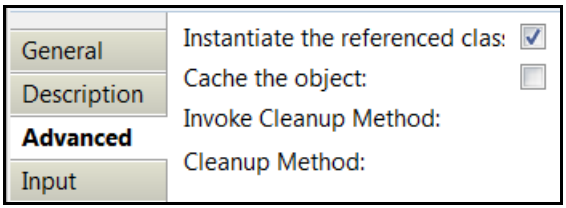
- Still in the *General* Tab, for **Class Name**: Browse and in the empty box, type **`Calc`**



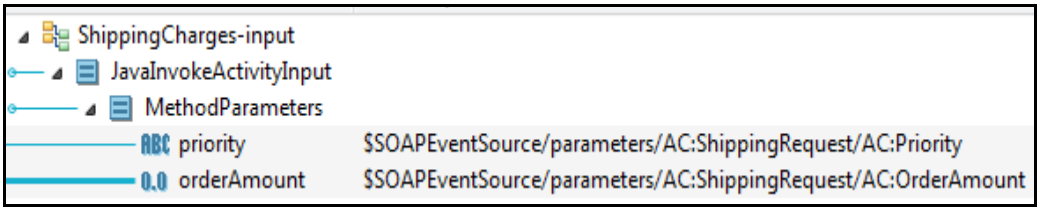
  - The CalcCharges class is displayed, select it and click **OK**
  - Method: **calculateshippingCost()** (Browse and choose)



- In ***Advanced*** Tab, instantiate the Referenced Class: **Checked**



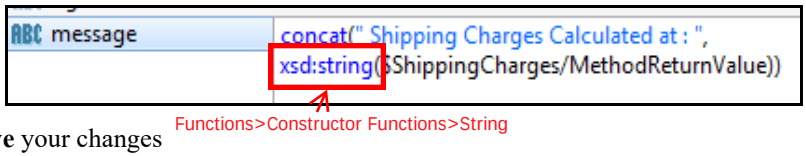
- Input Tab: **Map**



- **Save** your changes and verify there are no errors

8. Add a log activity to record the value returned by the Java code.

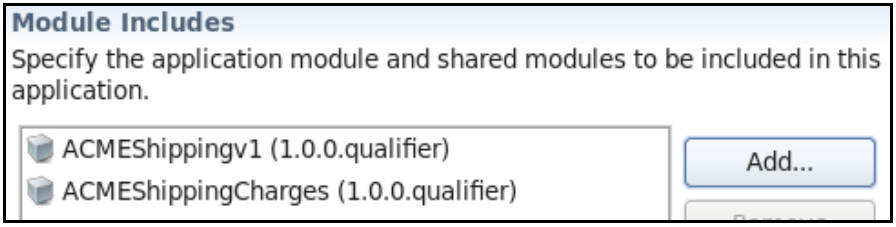
- Add a **Log** activity after the JavaInvoke activity and configure it
- Name: **`Shipping`**
- *Input* > ***Message***: Add concat function as shown:



- **Save** your changes

9. Add the plug-in project (bundle) to your application before running it.

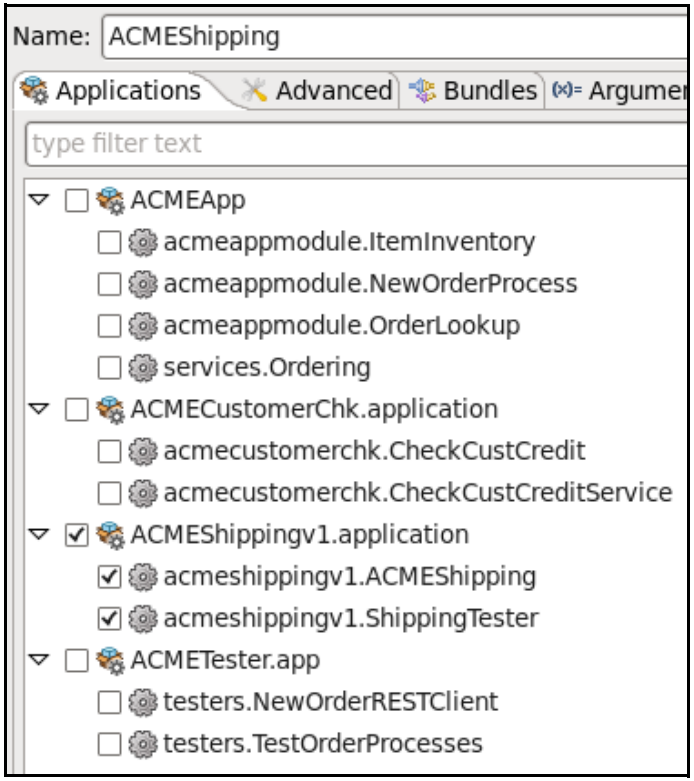
- Browse to *ACMEShippingv1.application* > ***Package Unit***
  - Double click **Includes** to open options
  - Click **Add**.. and select **ACMEShippingCharges**



  - **Save** changes and close the view

10. Launch the debugger and run the BusinessWorks application and Java program.

- Run ACMEShipping with the Debugger



- Locate Process Launcher tab, then choose *ACMEShipping* > *DoShippingOp* > ***Request***
- Enter this data:
  - CustID = **`8889`**
  - OrderAmount = **`1212`**
  - Priority = **`HIGH`**
  - OrderDate = Today (select by click ... button)

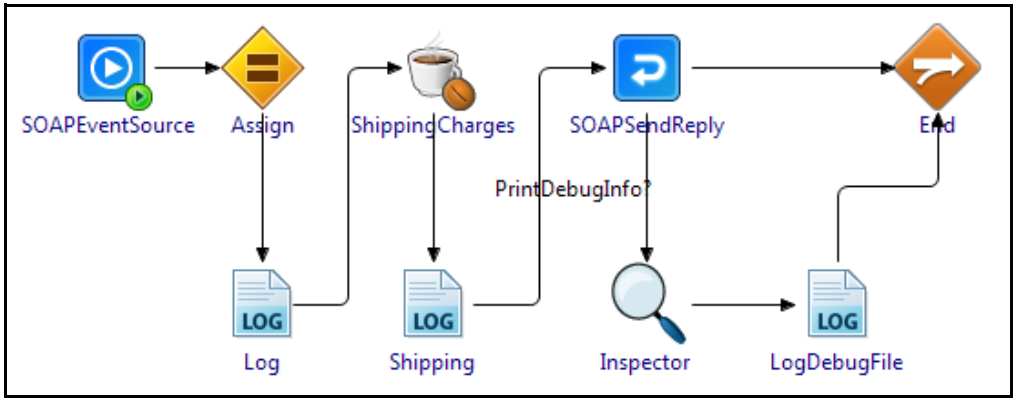
A la primera no muestra los campos el Launcher.
Solución: Parar la ejecución, reiniciar BST y ejecutar nuevamente la configuración ACMEShipping

11. Observe the results returned in the console

- Examine the console for the BusinessWorks application

```
18:00:15.491 INFO  [bwEngThread:In-Memory Process Worker-1] c.t.b.p.g.Log.ACMESh
ippingv1.Log - Received Shipping  Request for CustiID: 8889. JobId [bw0a104], Pr
ocessInstanceId [bw0a104], Activity [Log], Process [acmeshippingv1.ACMEShipping]
, Module [ACMEShippingv1:1.0.0.qualifier], Application [application-debugger:1.0
].
18:00:15.525 INFO  [bwEngThread:In-Memory Process Worker-1] c.t.b.p.g.L.ACMEShip
pingv1.Shipping -  Shipping Charges Calculated at : 48.48
```

- Confirm that the Shipping Charges are: **48.48**
- In process launcher, change Priority to LOW
  - Execute (run) the process again
  - Confirm that Shipping Charges are : **24.24**
- The updated process is as shown below:



- **Stop** the tester

Lab K - Exercise 2: Call External Method

Overview

In this exercise, import a pre-built JAR file into your application module. Then, use the Java Invoke activity to invoke a method in the JAR file.

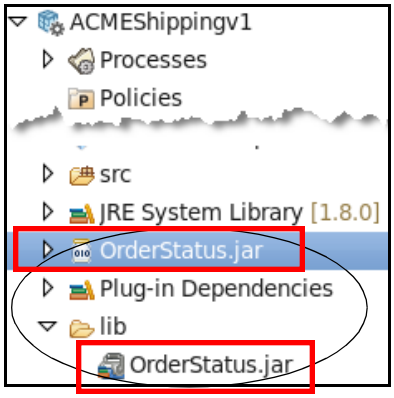
Steps

1. Switch to design perspective.

- Click on the menu option to switch back to the design perspective
  - Option on top left corner

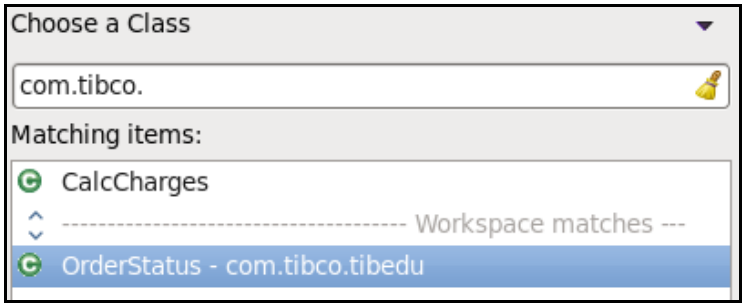
2. Locate the external JAR file and add it to the application module.

- In File Explorer (Business Studio), browse to the **code** folder
  - Locate **OrderStatus.jar**
  - Drag it to ***ACMEShippingv1 > lib***
  - Select option to **Copy**
  - Observe that the file is copied to two locations as shown

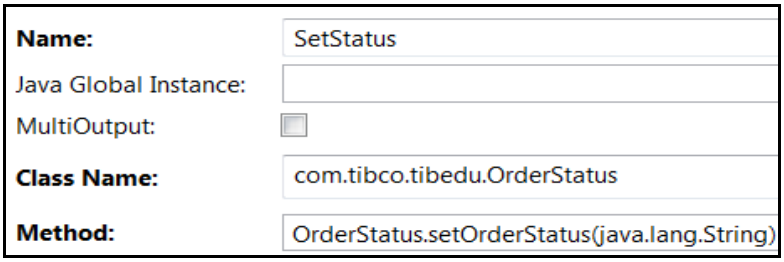


3. Update the new order process to access a method in this JAR file.

- In ACMEShipping process, add a **Java Invoke** activity between the **Shipping** (log) activity and SOAPSendReply
- Configure the activity to invoke *setOrderStatus* method
  - Name: **`SetStatus`**
  - Class Name: **OrderStatus** (browse to select)
    - Search for **`com.tibco.`**
    - Select **OrderStatus - com.tibco.edu**



  - Method: **setOrderStatus**



Tip: When a Jar file is added to an Application Module, there is no need to change any configuration settings such as build path or module dependencies. All public methods are immediately available to processes in the module.

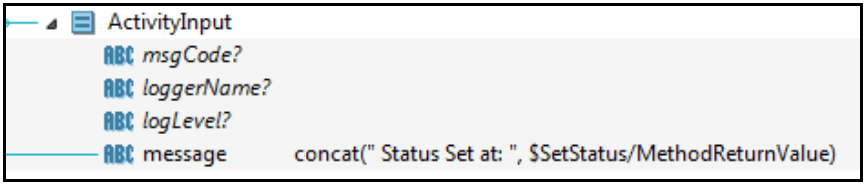
The application module must have Java nature to support this functionality.

- Provide the input to the Java method
  - In *Input* tab: **Map** priority as shown

- **Save** changes
- Confirm that there are no problems with the activity

4. Add a log activity to display the results of the java invoke activity.

- Add a **Log** activity next to the **SetStatus** (Java Invoke)
- Update the activity as shown:
  - Name: **`OrderStatus`**
  - Input: Map as shown:



- **Save** changes

5. Run the solution and verify that the order status is set properly.

- Start the Tester
  - Debug ~~Run~~ the BusinessWorks Application (ACMEShipping)
  - Using Process Launcher, submit the same request as before (Priority = **HIGH**)
- Verify the results in the BusinessWorks Console window:

CustID = 8889
OrderAmount = 1212
Priority = HIGH
OrderDate = Today (select by click ... button)

```
ippingv1.Log - Received Shipping  Request for CustiID: 8889. JobId [bw0a102], Pr
ocessInstanceId [bw0a102], Activity [Log], Process [acmeshippingv1.ACMEShipping]
, Module [ACMEShippingv1:1.0.0.qualifier], Application [application-debugger:1.0
].
11:36:55.563 INFO  [bwEngThread:In-Memory Process Worker-4] c.t.b.p.g.L.ACMEShip
pingv1.Shipping -  Shipping Charges Calculated at : 48.48
11:36:55.597 INFO  [bwEngThread:In-Memory Process Worker-4] c.t.b.p.g.L.A.OrderS
tatus -  Status Set at: AIR_FREIGHT
```

Tal vez haya que hacer click en "request" del lado izq. para que habilite el botón de Start.
También click en el botón de Refresh.

- Confirm that Order Status is **AIR_FRIEGHT**
- Run process again, with Priority = LOW

```
ippingv1.Log - Received Shipping  Request for CustiID: 8889. JobId [bw0a103], Pr
ocessInstanceId [bw0a103], Activity [Log], Process [acmeshippingv1.ACMEShipping]
, Module [ACMEShippingv1:1.0.0.qualifier], Application [application-debugger:1.0
].
11:39:19.401 INFO  [bwEngThread:In-Memory Process Worker-7] c.t.b.p.g.L.ACMEShip
pingv1.Shipping -  Shipping Charges Calculated at : 24.24
11:39:19.414 INFO  [bwEngThread:In-Memory Process Worker-7] c.t.b.p.g.L.A.OrderS
tatus -  Status Set at: 3_DAY
```

- Confirm that Order Status is **3_DAY**
- Stop the tester when done
- Keep Business Studio open for the next exercise

Lab K - Exercise 3: Packaging JARs

Overview

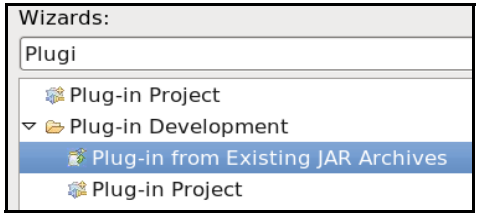
In this exercise use version 6 recommended practices to define an OSGi bundle for the external Java code. Then use this bundle in your BusinessWorks application.

Steps

1. Define an OSGi bundle project containing the JAR file.

Ir a File en el menú superior de BST

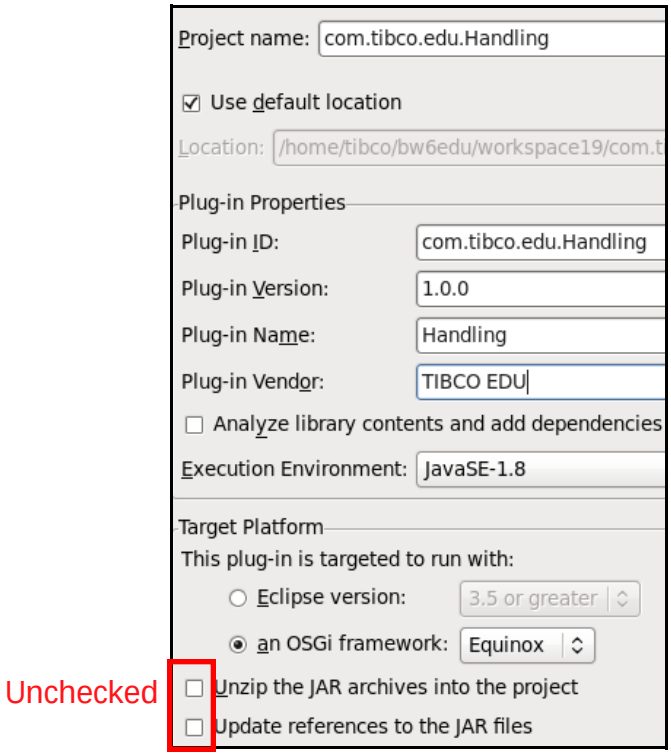
- ~~In Project Explorer, right-click~~ and choose *New* > ***Other.....***
  - In the list that appears, start typing **Plug**
  - Now choose **Plug-in from Existing JAR Archives**



  - Click **Next**
  - Now click A**dd External** and select `/home/tibco/bw6edu/dev/Files/Handling.jar`
  - Click **Next**
- The next screen requires explanation and it discussed on the next page

2. Provide plug-in project properties.

- Provide these properties
  - Project name: **`com.tibco.edu.Handling`**
  - Plug-in ID: **com.tibco.edu.Handling** (filled in automatically)
  - Plug-in Name: **`Handling`**
  - Plug-in Vendor: **`TIBCO EDU`**
  - Target Platform: **Select an OSGI Framework (Equinox)**
  - **Uncheck:** Unzip the JAR Archives

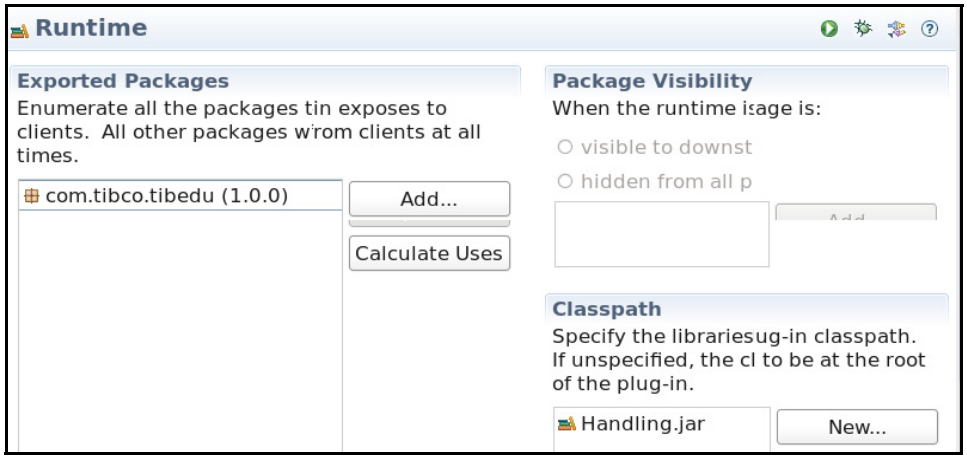


- Click **Finish** and click **NO** when asked to open Plug-in perspective

Note: Recommendations:
- Project name: should use reverse domain naming convention
- Plug-in ID should match project name
- Do not Unzip Jar file

3. Examine the manifest file and confirm settings for the plug-in project.

- By default, the MANIFEST.MF file is opened
- Click on *Runtime* tab and confirm that classpath specifies **Handling.jar**
- Exported Packages contains: com.tibco.tibedu(1.0.0)



- Close MANIFEST.MF view when done

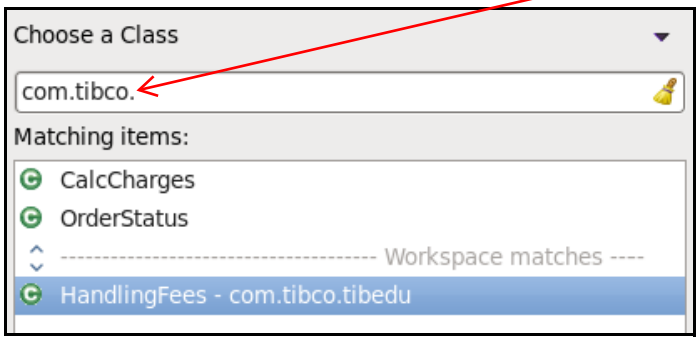
4. Reference the plug-in project as a shared module in your Shipping application module .

- Browse to and open ***ACMEShippingv1 >*** *Module Descriptors* > ***Dependencies***
- In **Required Modules**, click **Add..**
- Select **com.tibco.edu.Handling (1.0.0)** and click OK

De regreso en el proceso ACMEShipping.bwp

5. Add a Java invoke activity to calculate handling charges.

- Add a Java Invoke activity ~~after~~ before *SetStatus* ~~next to the Log activity~~ ~~*(OrderStatus)*~~
- General : Name: **`HandlingCharges`**
  - Class Name: **`com.tibco.tibeu.HandlingFees`** Escribir hasta com.tibco. para que aparezca en la lista



  - Method: getHandlingFee(String, Double)

Name:	HandlingCharges
Java Global Instance:	
MultiOutput:	☐
Class Name:	com.tibco.tibedu.HandlingFees
Method:	HandlingFees.gethandlingFee(java.lang.String, java.lang.Double)

- **Save** changes

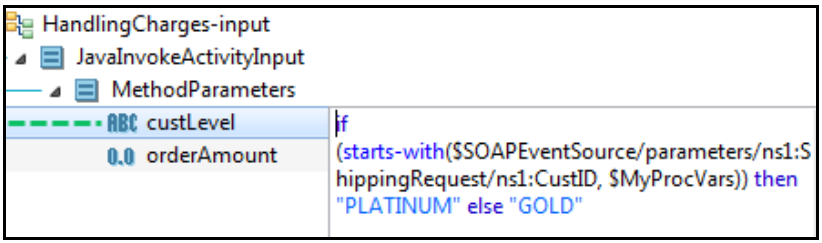
6. Provide mapping values for handling charges.

- For CustLevel, use this mapping:

  ```
  if
  (starts-with($SOAPEventSource/parameters/ns1:ShippingRequest
  /ns1:CustID, $MyProcVars)) then "PLATINUM" else "GOLD"
  ```

- For orderAmount , map

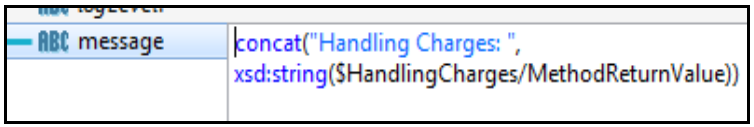
  ```
  $SOAPEventSource/parameters/ns1:ShippingRequest/ns1:OrderAmo
  unt
  ```



Tip: If the customer ID starts with a special sequence of characters, the level is set to PLATINUM, This special sequence of characters is defined in a Process Variable named MyProcVars.

7. Add a log activity to display handling charges.

- Add a log activity after the **HandlingCharges** activity
- Name: **`Handling`**
- Input: Configure as shown:



- **Save** changes

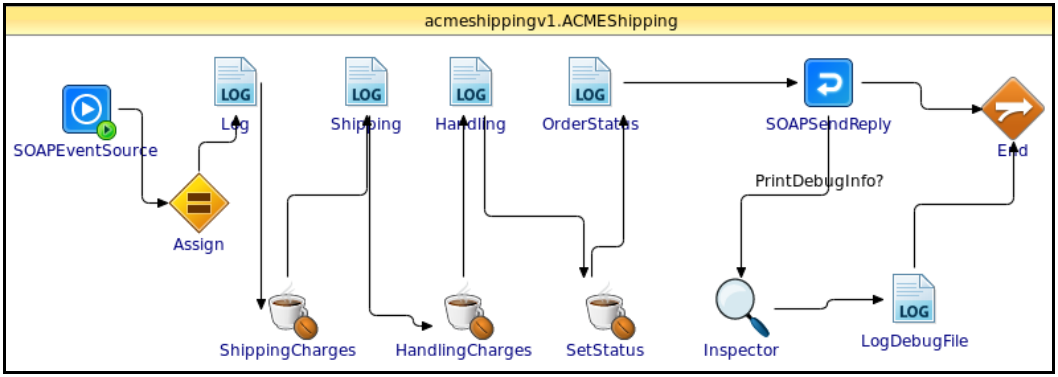
8. Update the Shipping Reply activity with the Total Charges and Status.

- Update the SOAPSendReply as shown below:
- TotalCharges:

  `$ShippingCharges/MethodReturnValue + $HandlingCharges/MethodReturnValue`
- Status: **`$SetStatus/MethodReturnValue`**

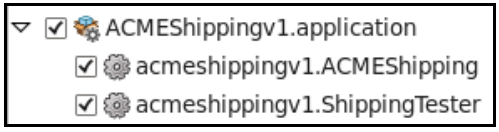
ShippingResult	
CustID	$SOAPEventSource/parameters/ns1:ShippingRequest/ns1:CustID
TotalCharges	$ShippingCharges/MethodReturnValue + $HandlingCharges/MethodReturnValue
DeliveryDate	if (starts-with($SOAPEventSource/parameters/ns1:ShippingRequest/ns1:CustID, $
Status	$SetStatus/MethodReturnValue

- Confirm there are no errors in your process
- **Save** all changes
- Verify your process configuration



9. Run the ACMEShipping application.

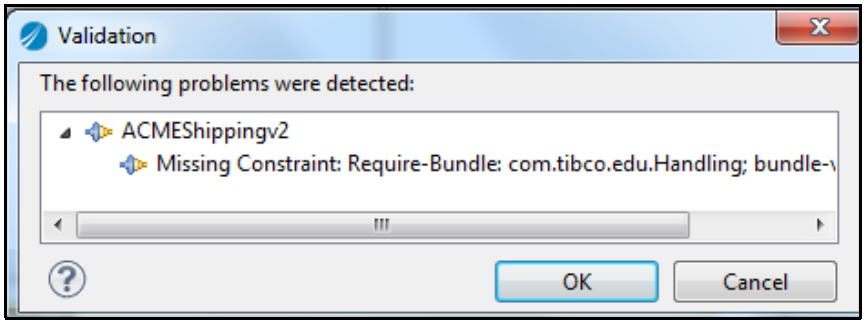
- Run the ACMEShipping configuration



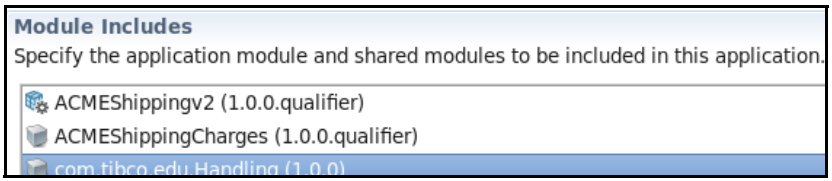
- Verify the application runs and charges are calculated

```
300006: Started BW Application [ACMEShippingv1.application:1.0]
ppingv1.Log - Received Shipping  Request for CustiID: 8899. JobId [bw0a101], Proc
ingv1.Shipping -  Shipping Charges Calculated at : 40
ingv1.Handling - Handling Charges: 20
atus -  Status Set at: AIR_FREIGHT
gFile - Detailed Debug  Info written to /home/tibco/bw6edu/dev/Files/mydebug.txt.
```

- If an error about a missing bundle appears:



- To resolve:
  - Stop running the Application
  - Browse to and open *ACMEShippingv1.application* > *Package Unit* > ***Includes***
  - Click **Find Missing**, and *com.tibco.edu.Handling(1.0.0)* is added
  - Save



- Close this view

> ***Tip:*** Note: This Java program calculates Handling Fees based on customer level (GOLD, SILVER or PLATINUM and Amount < 1000 or more than 1000. You can examine the code in the same location where you obtained Handling.jar.

- Stop the Debugger.

10. Export your solution as a ZIP file and shut down Business Studio.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File: `/home/tibco/bw6edu/dev/exports/`**`UsingOSGIJava.zip`**
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created
- **Exit** Business Studio

Notes

Notes

Unit 12 Invoking Services

This unit describes process details and service invocation.

Topics

- *Objectives,* page 364
- *BusinessWorks 6: Process Types,* page 365
- *Sub-Process Types,* page 366
- *Invoking WSDL Based Sub-Process,* page 367
- *Dynamic Process Invocation,* page 368
- *Dynamic Process: Name Generation,* page 369
- *Service Invocation: Internal,* page 370
- *Invoking an External Service,* page 371
- *Reference Binding: Review,* page 372
- *Business Case Implementation,* page 373
- *Process-Pattern Mapping to Test Invocation Options,* page 374
- *Review,* page 375
- *Lab Overview,* page 376

Objectives

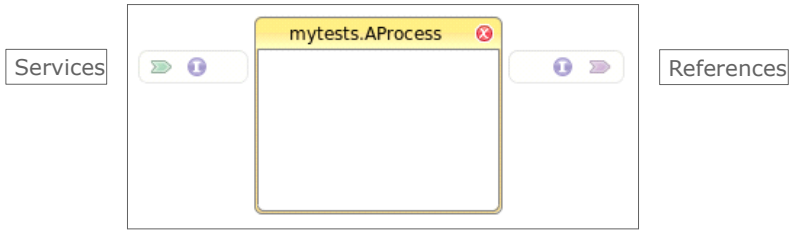
- Examine process details
- Compare process and service invocation
- Explain process invocation options
- Describe dynamic process invocation
- Explore how to invoke services

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Objectives

BusinessWorks 6: Process Types

- BusinessWorks 6: Service based from the start
- Process types
  - Classic starter process
    - Timer, JMS Receiver, Rendezvous Receiver etc
  - Services
    - REST or SOAP based



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

BusinessWorks 6: Process Types

BusinessWorks 6 process definitions are designed to be easily converted to services without a lot of configuration.

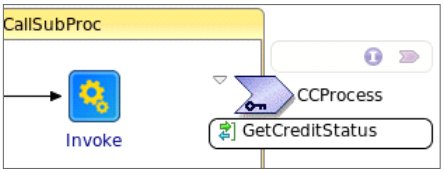
Classic process starters from version 5 are supported. These include starter activities like Timer, JMS Receiver, Rendezvous subscriber and others.

BusinessWorks 6 applications are designed to be modular from the start. Instead of one long process, image several small services that can be called as needed. A process and service are equivalent from that perspective. We will see this in the upcoming slides.

A stateless service is a RESTful service or SOAP service, and a state-ful service can only be a SOAP based service.

Sub-Process Types

- Option 1: Non WSDL Based
  - Provide input/output schema
  - Spawn: separate thread
  - Green process boundary
- Option 2: WSDL Based
  - WSDL used as contract
    - Operation + Input/Output
    - No more free form schemas
  - Reference added

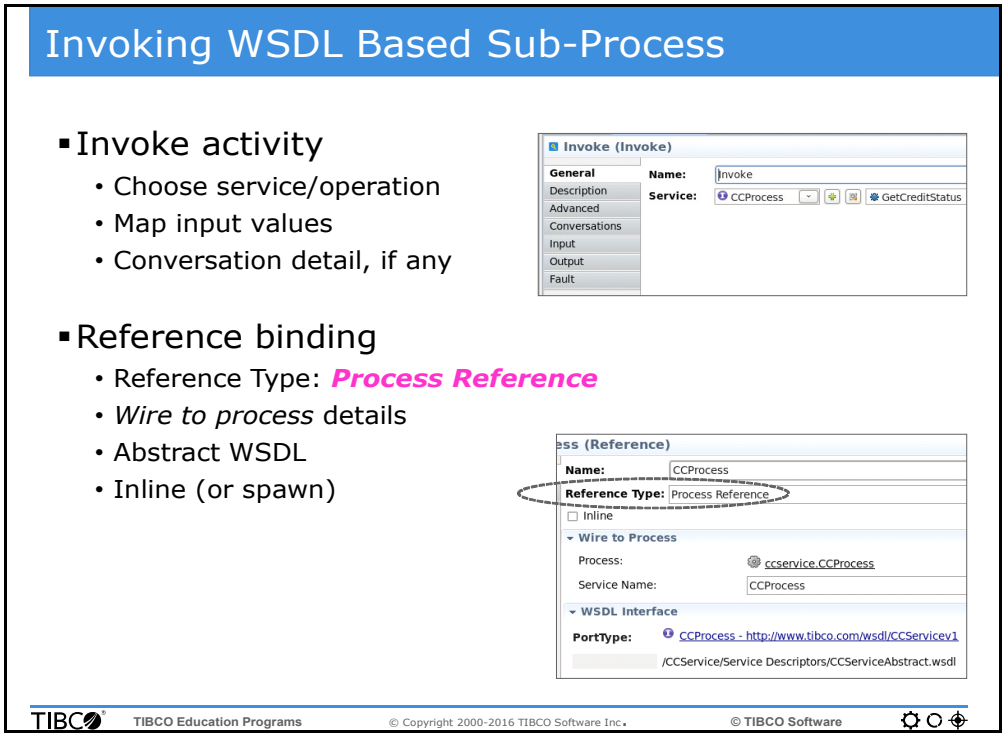


TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Sub-Process Types

BusinessWorks 5 supports invocation of sub-process through the call sub process activity. In this activity, you can also override the process by using the dynamic process override, and provide an XPath expression which resolves to a process name to be called. The input and output schema of the calling process and called process have to be in sync, and these can be free-form, defined using the input/output editor.

BusinessWorks 6 has a few notable differences. When you add an invoke activity to a process, this indicates your intention to invoke a sub-process. This also adds a reference binding to the process boundary as shown. The invoke activity uses a WSDL as a contract to define the message structure for input and output. There is no input/output editor to define free form schemas. An abstract WSDL is always required for enforcing the contract, and if it does not exist, one is created for you.

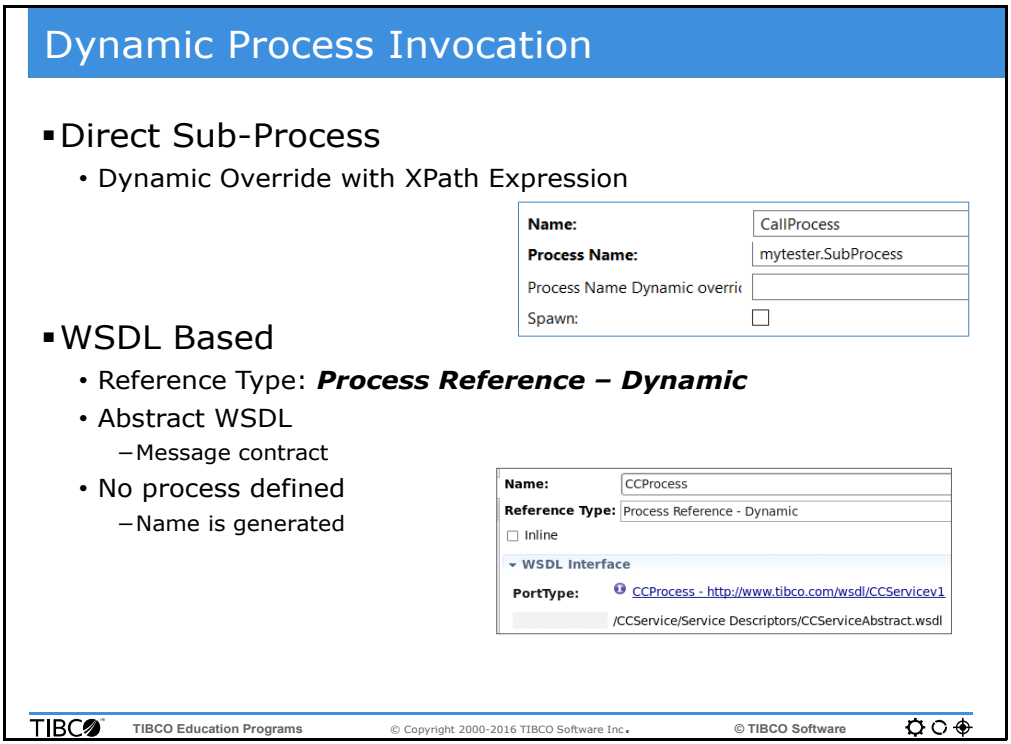


Invoking WSDL Based Sub-Process

A normal sub-process invocation involves configuring the invoke activity and reference binding:

Invoke activity: Here, choose the WSDL operation which will be used for messaging details. The input and output tabs are populated according to this WSDL operation. If this process is part of a conversation, you can enter conversation details as well.

Reference binding: Confirm that reference type is process reference. This provides the option to invoke the sub-process as an inline process. It also provides the wire-to-process section, where you can select the actual process to be called. The called process must provide an implementation of the WSDL operation used by the invoke activity.



Dynamic Process Invocation

In version 5, dynamic process binding is provided by the dynamic override with XPath mapping as shown above.

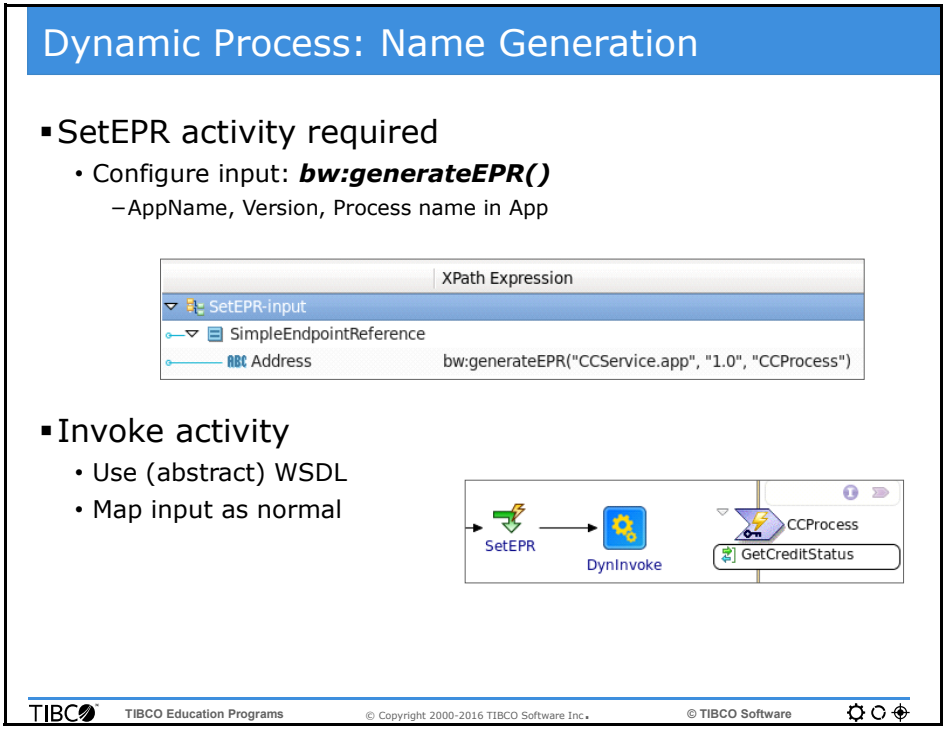
In version 6, there are a few changes:

In reference binding, change the reference type to Process Reference-Dynamic. Doing this causes a lightning bolt to appear through the binding, indicating its dynamic nature.

There is an abstract WSDL for the message contract.

There is no wire-to-process option, since the process name must be generated. The invoke activity remains unchanged, since it still uses the abstract WSDL and maps the input values.

Generating of the process name involves additional configuration, as seen next.



Dynamic Process: Name Generation

SetEPR: Set End Point Reference. To generate the process name in dynamic process invocation, you must add a SetEPR activity to the process.

The SetEPR activity must be added before the invoke activity, since the result of this activity is used to call the process.

The SetEPR activity itself has a single input field named Address, where you must add the bw:generateEPR() function.

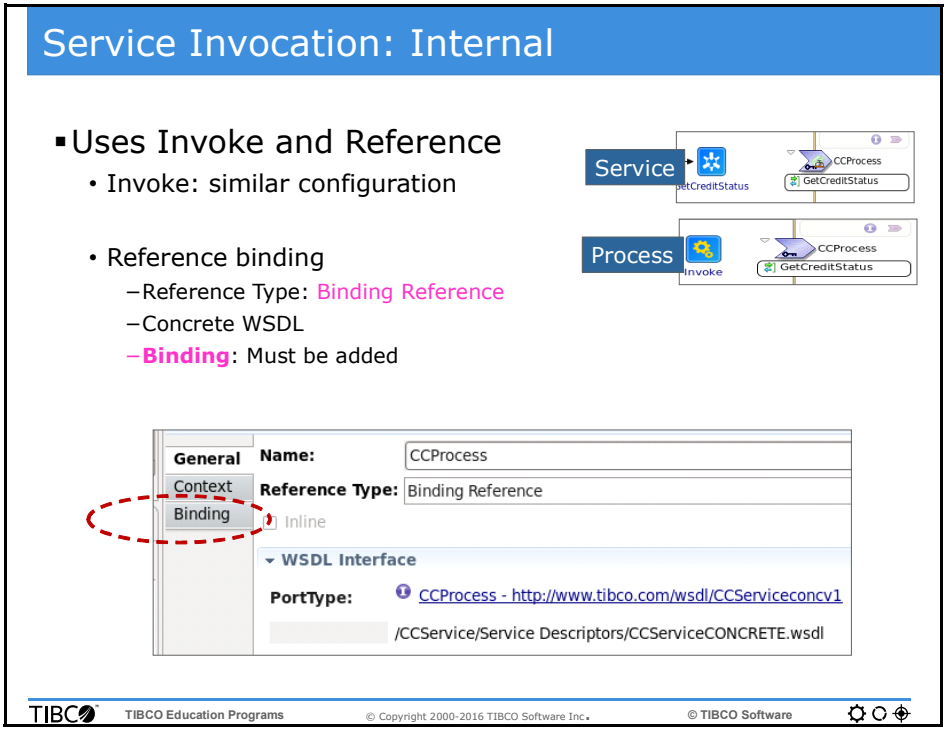
The BW function uses three arguments to dynamically generate an endpoint that can be invoked.

The first argument specifies the application which is running the sub-process.

The second argument is the version number of the application running the sub-process.

The third argument is the name of the sub-process inside the application.

At runtime, these three arguments are used to dynamically generate an End Point Reference for a sub-process, and the sub-process is invoked.

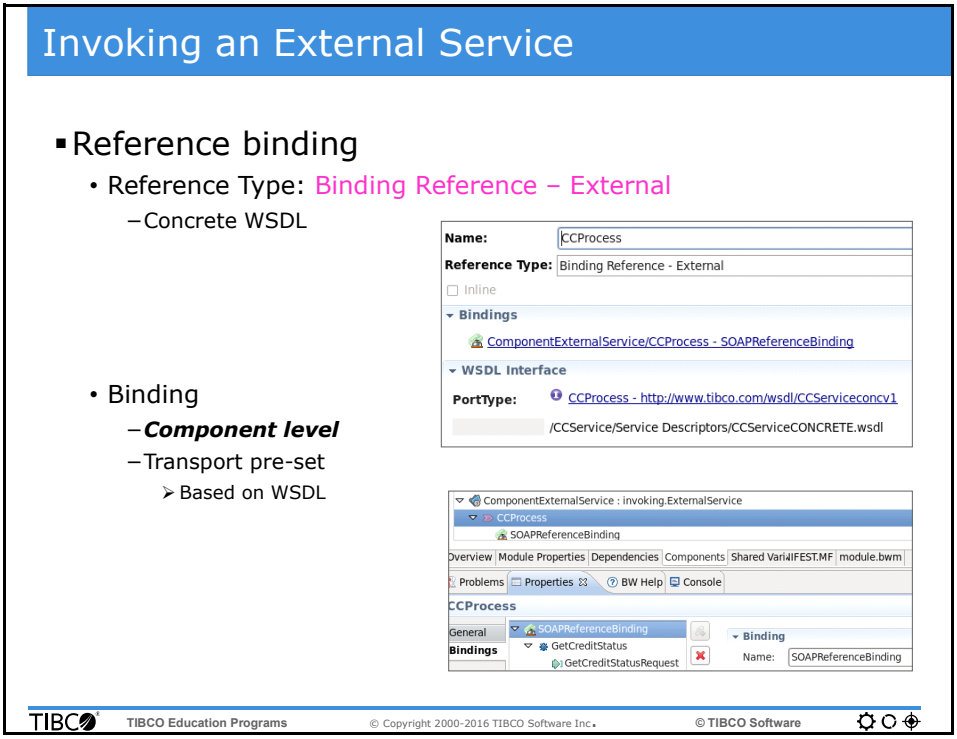


Service Invocation: Internal

Service invocation is similar to process invocation. It uses the same invoke activity and reference binding, with different settings.

The invoke activity uses a concrete WSDL for the message contract, and provides the input mappings. The Reference Binding uses a Reference Type of Binding Reference to invoke an internal service. An internal service is a service that is available within the same application.

When the reference type of binding reference is chosen, an extra tab appears, named binding. Here, you must define the HTTP client configuration that is used to call the internal service.



Invoking an External Service

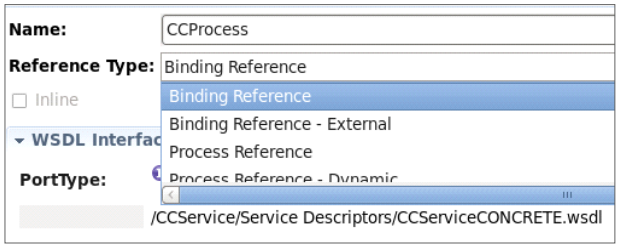
Invoking an external service takes the discussion of invocation to its final capability.

In Reference Binding, if you choose Reference Binding- External, it implies that you want to invoke an external service using a concrete WSDL provided by the service provider.

In this case, binding is configured at the component level. This means that you can change the values at deployment time. The transport type and details are read from the concrete WSDL and are filled up automatically.

Reference Binding: Review

- Reference binding
  - Decides process or service invocation
  - Properties change based on selection
    - Wire to process
    - Abstract or Concrete WSDL
    - Binding tab



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Reference Binding: Review

By now, it is obvious that the configuration of the Reference Binding determines whether a process or a service is to be invoked.

The invoke activity does not have these details, it simply uses the WSDL for message input/out details, and that's it.

There are four kinds of reference bindings:

Process reference: Hard wired to a sub-process.

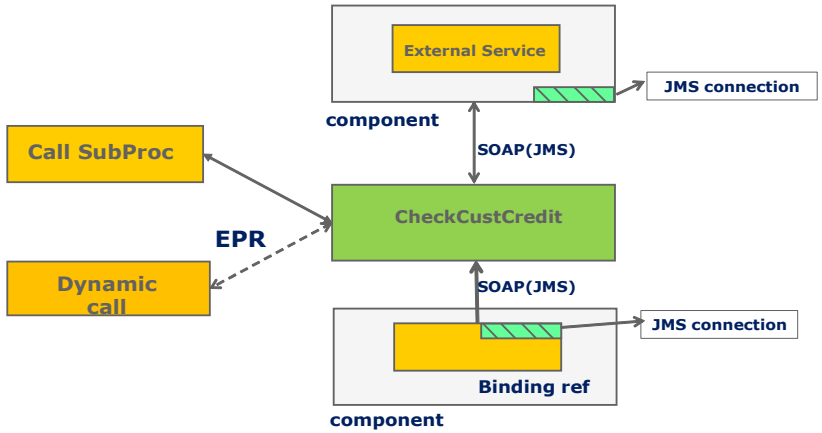
Process reference – dynamic: determine a process name at runtime, using the SetEPR activity.

Binding Reference: Invoke an internal service.

Binding Reference – external: Invoke an external service.

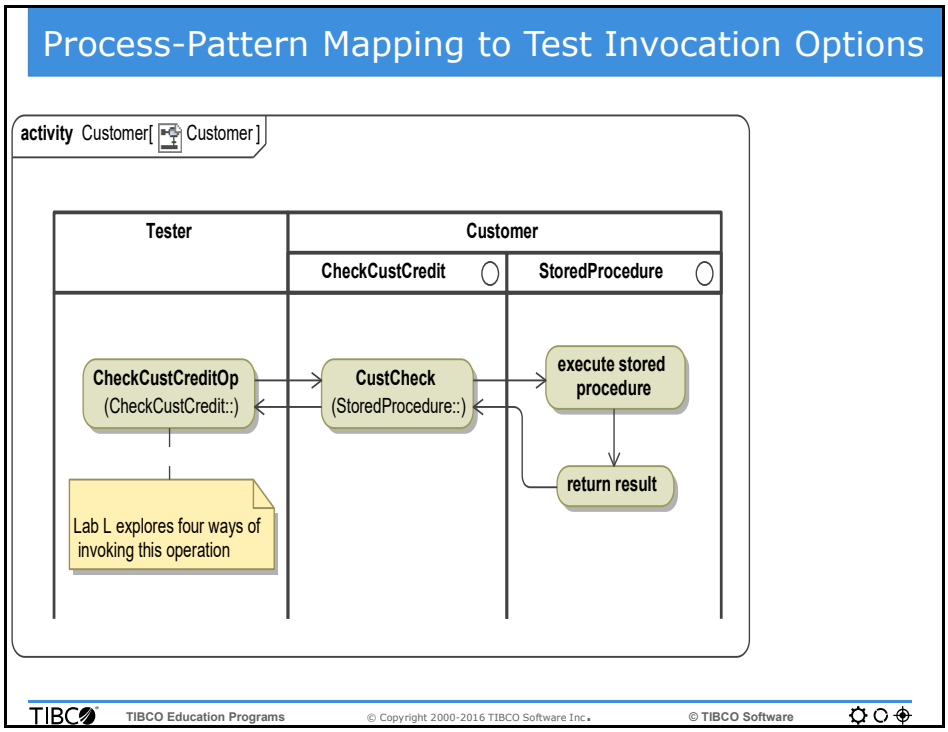
Business Case Implementation

- Reference bindings to invoke CheckCustCredit
  - As a process: 2 options
  - As service: 2 options



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Business Case Implementation



Process-Pattern Mapping to Test Invocation Options

Review

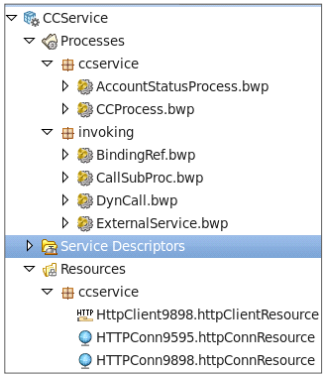
- Process types: Classic and service-based
- Process and service invocation are similar
- Invoke activity: WSDL as a contract for message
- Dynamic process call uses SetEPR activity
- Reference binding: Choose process or service call

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Review

Lab Overview

- Import starter service project
- Implement process invocation
  - Simple and dynamic
- Implement service invocation
  - Internal and external service



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Lab Overview

Notes

Notes

Lab L Process Invocation Techniques

Overview

In this lab you will examine the details for process invocation. You will differentiate between process reference and binding reference, and build logic to dynamically invoke a sub-process.

Objectives

- *Sub Process Invocation, page 380*
- *Dynamic Sub Process,* page 384
- *Using Binding Reference,* page 388
- *Calling External Service,* page 393

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO BusinessWorks 6.3
- Previous lab has been completed

Directions

- Complete the exercises that follow

Lab L - Exercise 1: Sub Process Invocation

Overview

In this exercise implement the two ways of invoking a process.

Steps

1. Using Business Studio, add a log activity to the CheckCustCredit process.

- Open Business Studio with the Workspace `/home/tibco/bw6edu/workspace`
- Open ACMECustomerChk > Processes > acmecustomerchk >CheckCustCredit.bwp ~~*ACMECustomerChk > Processes > **CheckCustCredit***~~
- Add a log activity between Start and JDBC-CallProcedure
- Name: **`LogCustInfo`**
- Input Message: **`concat(" Received Request for AccountID : ", $Start/CustID)`**

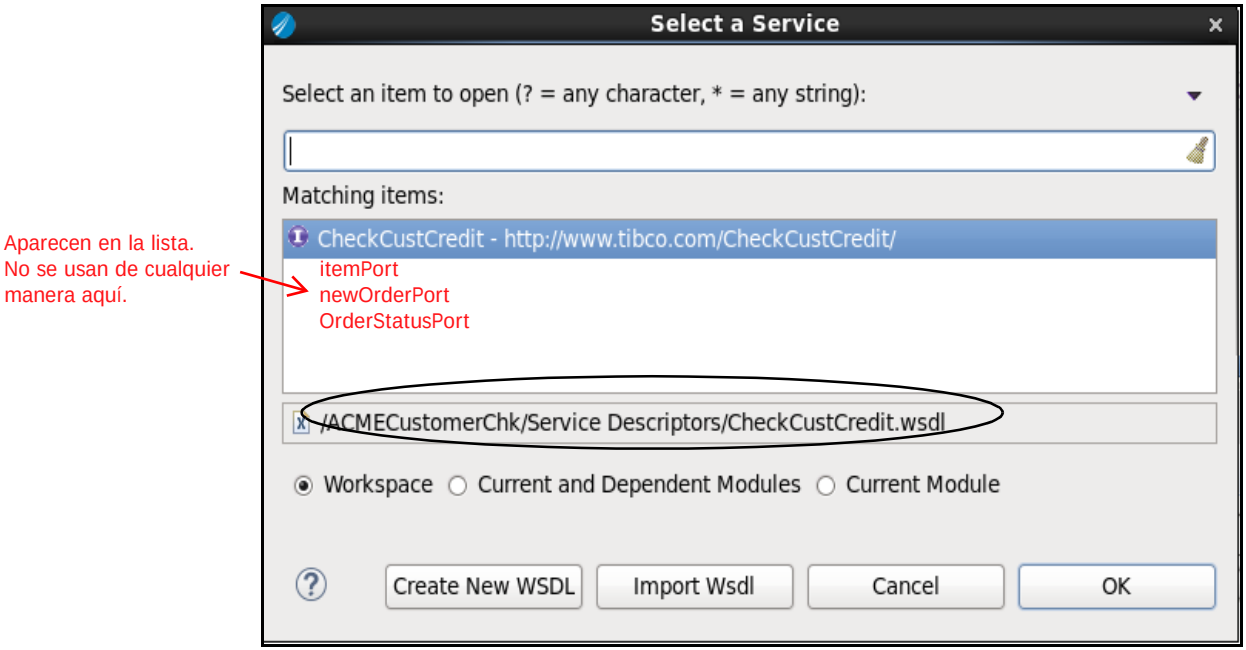
2. Define a new package and a process for this application.

- In *ACMECustomerChk* > *Processes*, add *New* > ***BusinessWorks process***
  - Package Name: **`Invoking`**
  - Process Name: **CallSubProc**
  - Patterns: **Empty process,** click **Finish**
- The process opens
  - Add a timer activity, and accept all defaults for this activity

- **Save** changes

3. Add an invoke activity to call CreditService and configure it.

- Add an **Invoke** activity (from Basic Activities)
  - Connect Timer to Invoke
- Configure invoke activity: *Properties* > ***General*** tab:
  - Name: **`SubProcInvoke`**
  - Service: Click + sign and select **CheckCustCredit** from **CheckCustCredit.wsdl**



  - Click **OK** and a Reference Chevron is added to the process
- Properties Input tab:
  - CustID: **"SUB-PROC-TEST"**
  - CreditAmt: **1000**

4. Update the Reference binding to call the Service as a sub process.

- The Reference chevron has been added to the process boundary, click on it
- In *Properties* > ***General***
  - Reference Type: **Process Reference**
  - Process: **CheckCustCreditService** (browse and select)
  - Service Name: **CheckCustCredit** (select from list)
- **Save** changes and there should not be any errors

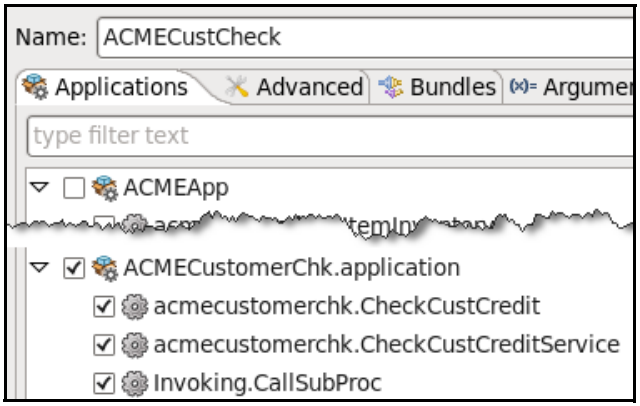
Analysis: When invoking a sub-process, a WSDL is used as a contract for invocation. Abstract WSDLs are used here, since only the Operation details are needed (input and output values).

Further more, an abstract WSDL implies that a process is internally invoked, instead of invoking it as a service (through HTTP or JMS).

An abstract WSDL can be implemented by more than one BusinessWorks process, so the required sub-process is identified in the Reference Binding section.

5. Test the process invocation and confirm that it works.

- Launch the debugger with a **New Configuration** to run ACMECustomerChk



  - Confirm that the Timer fires and the Service is invoked as a sub-process

- Observe console for log message:

```
FRWK-300021: All Application dependencies are resolved for Application [ACMECust
omerChk.application:1.0]
-300006: Started BW Application [ACMECustomerChk.application:1.0]
tInfo -   Received Request for AccountID : SUB-PROC-TEST
```

- **Stop** the debugger when done
- Switch back to Design perspective

Lab L - Exercise 2: Dynamic Sub Process

Overview

In this exercise you will test your Web service and confirm that both operations can be invoked successfully.

Steps

1. Add a new BusinessWorks process named DynCall.

- Right-click on package *Invoking,* and add a **BusinessWorks Process**
- Process Name: **`DynCall`** **(**Patterns: Empty Process)
- Click **Finish**

2. Add an invoke activity to call CreditService dynamically and configure it.

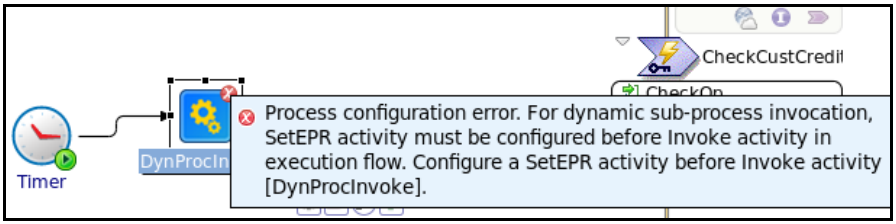
- Add an **Invoke** activity (from Basic Activities)
- Configure it as follows:
- *Properties* > ***General*** tab:
  - Name: **DynProcInvoke**
  - Service: Click + sign and select **CheckCustCredit** from **CheckCustCredit.wsdl**
  - Click **OK** and a Reference Chevron is added to the process
- Properties Input tab:
  - CustID: **"DYN-SUB-PROC"**
  - CreditAmt: **1000**

3. Add a Timer activity to start the process.

- Add a Timer activity with default values
  - Connect Timer activity to Invoke activity

4. Update the Reference binding to call the Service as a dynamic sub process.

- The Reference chevron> *Properties* **> *General***
  - Reference Type: **Process Reference - Dynamic**
- **Save** changes, and a lightning bolt appears through the reference activity
- Click on DynProcInvoke activity and see the need to add a SetEPR activity



Analysis: Dynamically invoking a sub-process, also uses an abstract WSDL. The name and location of the sub-process are decided by the SetEPR activity.

5. Add a set end point reference activity to the process and configure it.

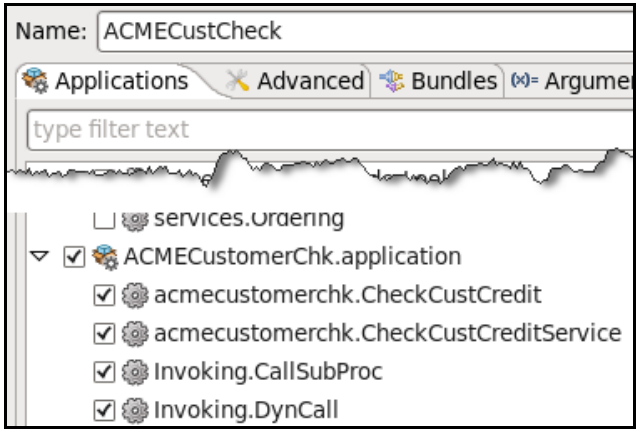
Basic Activities > Set EPR

DynProcInvoke

- Add a **SetEPR** activity between **Timer** and ~~**SubProcInvoke**~~
- *Properties* > *General* > ***Dynamic Reference***: **CheckCustCredit** (select from list)
- *Properties* > ***Input*** (Replace values in the address mapping):
  - Address =
    `bw:generateEPR("ACMECustomerChk.application","1.0", "CheckCustCredit")`

6. Test the process invocation and observe the error returned by the runtime.

- Launch the debugger and run the ACMECustomerChk applications



- Observe that the dynamic sub-process fails with this error

```
: The BW process [invoking.DynCall] instance faulted, JobId [bw0a101], ProcessInstanceId [bw0a101],
ParentProcessInstanceId [-], Module [ACMECustomerChk:1.0.0.qualifier], Application [ACMECustomerChk.
application:1.0].
   <CausedBy> TIBCO-BW-CORE-500052: Invoke activity [SubProcInvoke] fault.
   <CausedBy> com.tibco.bx.core.faults.BXException: BX-600006: Invoke activity, [SubProcInvoke] was
unable to send a message using partner link [CheckCustCredit-PortType], operation [CheckCustCredit-O
peration]. Cause: BX-905007: Dynamic service [bw:///ACMECustomerChk.application_1.0/ACMECustomerChk]
 not found, cause [No matching services registered]
```

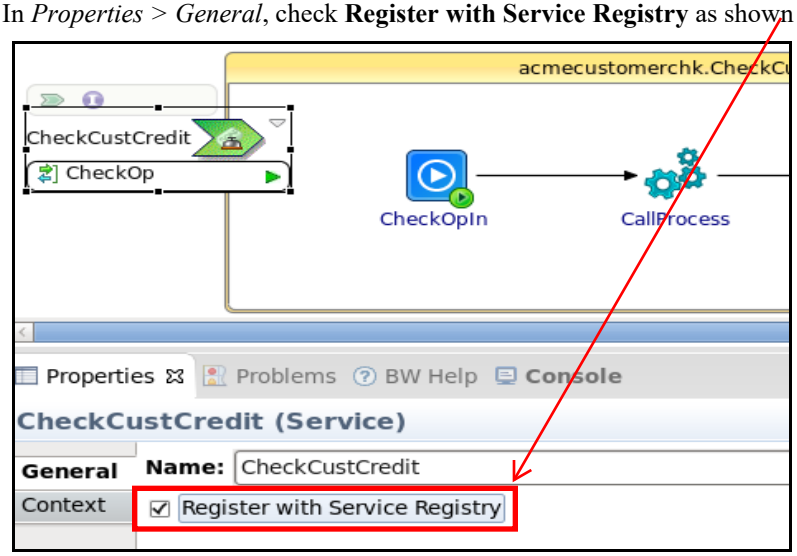
DynProcInvoke

7. Resolve the error by registering your BusinessWorks service with the Registry.

- **Stop** the debugger and switch back to design perspective

ACMECustomerChk > Processes > acmecustomerchk >CheckCustCreditService.bwp

- Open ~~*Processes* > **CheckCustCreditService.bwp**~~
  - Click on the service chevron (on Process boundary
  - In *Properties* > *General*, check **Register with Service Registry** as shown



Tip: Notice the service name: CheckCustCredit, the same defined in EPR activity

  - **Save** changes and close the process editor for this process

8. Restart debugger and confirm that dynamic invocation works.

- Restart the debugger
- Confirm that both process invocations are executed by looking at the console view

```
K-300019: BW Application [CCService.app:1.0] is impaired.
 - Started by BusinessStudio, ignoring .enabled settings.
-FRWK-300021: All Application dependencies are resolved for Application [CCService.app:1.0]
K-300006: Started BW Application [CCService.app:1.0]
vice.Log -  **** Received Request for AccountID : SUB-PROC ****
vice.Log -  **** Received Request for AccountID : DYN-SUB-PROC ****
```

- **Stop** the debugger and switch back to Design perspective

Sin asteriscos, porque no se agregaron cuando se definieron estos mensajes.

Lab L - Exercise 3: Using Binding Reference

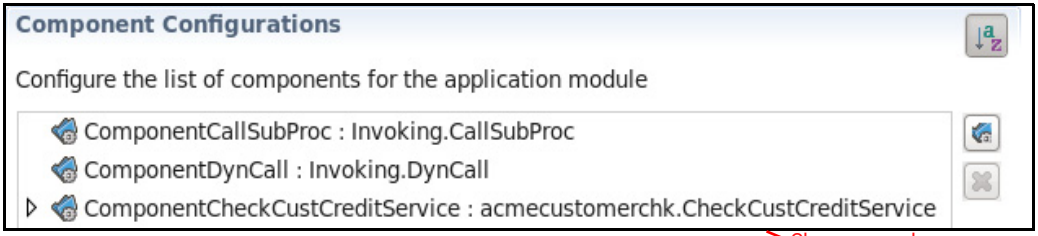
Overview

In this exercise you will use a binding reference to invoke your service.

Steps

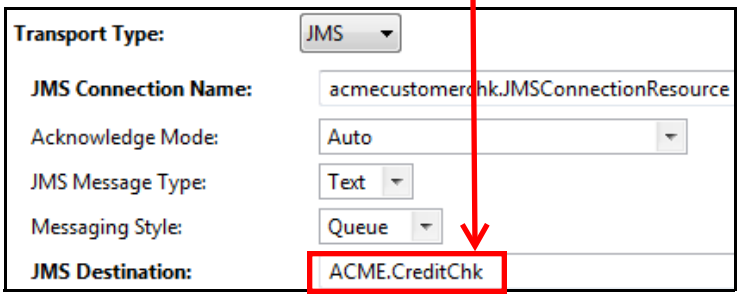
1. Generate the concrete WSDL for the Credit Check service.

- Browse to and open *ACMECustomerChk* > *Module Descriptors* > ***Components***



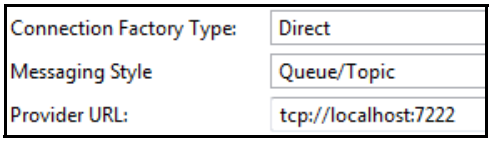
Chevron verde

- Expand *CheckCustCreditService* and click on its port type (*CheckCustCredit*)
- Now in Properties, click on **Bindings** and add a **SOAP Binding** Click en Finish
- Choose **SOAPServiceBinding1** in component view
  - For Transport, choose **JMS**
  - In the pop up window, choose **Create Shared Resource**
    - Accept default values and click **Finish**
  - Change JMS Message Type: **Text** Difícil seleccionar el valor de la DropDown List. Usar rueda del mouse.
  - JMS Destination: **`ACME.CreditChk`** Escribirlo



2. Fix the error with JMS Connection.

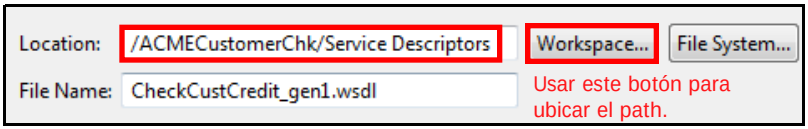
- Open *ACMECustomerChk* > *Resources* > *acmecustomerchk* > ***JMSConnectionResource***
  - Connection Factory Type: **Direct**
  - Messaging Style: **Queue/Topic**
  - Provider URL: **tcp://localhost:7222**



- Click **Test Connection** to confirm that it succeeds

3. Save the concrete WSDL for this service.

- Back in *Module Descriptors* > ***Components***, select ..../ *SOAPServiceBinding1*
- In **Bindings**, click on **Generate Concrete WSDL** and enter these details



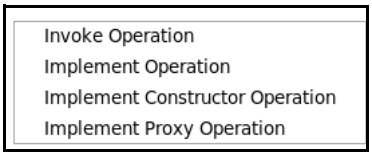
- Click **Next**
- Confirm Embed Abstract WSDL and Embed Schema are checked
- Click Advanced: Set Namespace URI = **`http://www.tibco.com/edu/CreditCheck`**
- Click **Finish**
- The WSDL opens, review it and close it

4. Add a new BusinessWorks process named BindingRef.

- Right-click on package *Invoking,* and add a **BusinessWorks Process**
- Process Name: **BindingRef** (Patterns: Empty Process), click **Finish**

5. Use the Concrete WSDL to define the invoke activity.

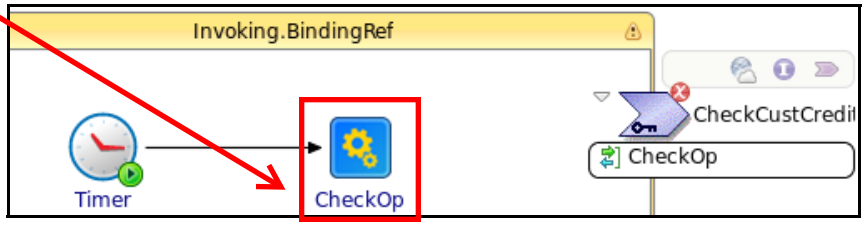
- Drag *Service Descriptors* > ***CheckCustCredit_gen1.wsdl*** onto the process
- In the wizard that pops up, choose **Invoke Operation**



  - The invoke activity and the reference is added
- *CheckOp* > *Input* tab

Click en la Actividad

  - CustID = **`"BINDING-REF"`**
  - CreditAmt: **`1000`**



6. Add a Timer activity to start the process.

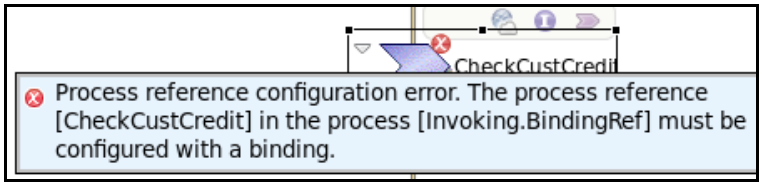
- Add a Timer activity with default values
  - Connect Timer activity to Invoke activity

7. Change the reference binding to use a binding reference.

- The Reference chevron has been added to the process boundary, click on it
- In ***Properties > General***
  - Reference Type: **Binding Reference**
  - **Save** changes

8. Add a binding to be used with the reference.

- An error appears indicating that a binding is needed

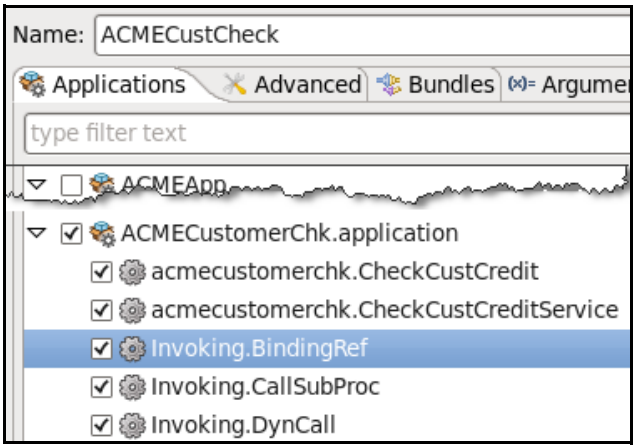


Analysis: Binding reference needs concrete WSDL, and client connection.

- A new tab, named **Binding** is added to the reference chevron properties
- On Binding tab, click *Add Binding*, then add **SOAP binding**
  - **Uncheck** Select/Create Required Resources
  - Click **Finish**
- Now in Transport Configuration
  - For *JMS Connection Name*, click **Browse** and select *JMSConnectionResource*
- **Save** changes and verify there are no errors.

9. Test the process invocation and confirm that it works.

- Launch the debugger



  - Examine the console for debug messages indicating that the process was run

```
Started by BusinessStudio, ignoring .enabled settings.
WK-300021: All Application dependencies are resolved for Application
00006: Started BW Application [ACMECustomerChk.application:1.0]
nfo -  Received Request for AccountID : BINDING-REF
nfo -  Received Request for AccountID : SUB-PROC-TEST
nfo -  Received Request for AccountID : DYN-SUB-PROC
```

- Stop the debugger
- Switch back to design perspective

Lab L - Exercise 4: Calling External Service

Overview

In this exercise, use an external binding reference to invoke your service as an external service.

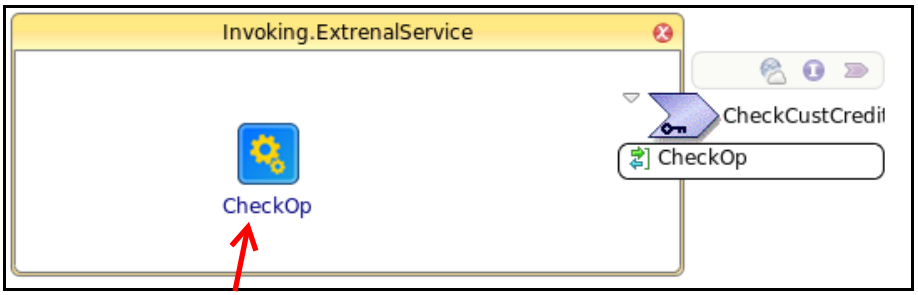
Steps

1. Add a new BusinessWorks process named ExternalService.

- Right-click on package *Invoking,* and add a **BusinessWorks Process**
- Process Name: **ExternalService** (Patterns: Empty Process), click **Finish**

2. Use the Concrete WSDL to define the invoke activity.

- Drag *Service Descriptors* > ***CheckCustCredit_gen1.wsdl*** onto the process
- In the wizard that pops up, choose **Invoke Operation**
  - The invoke activity and the reference is added as shown



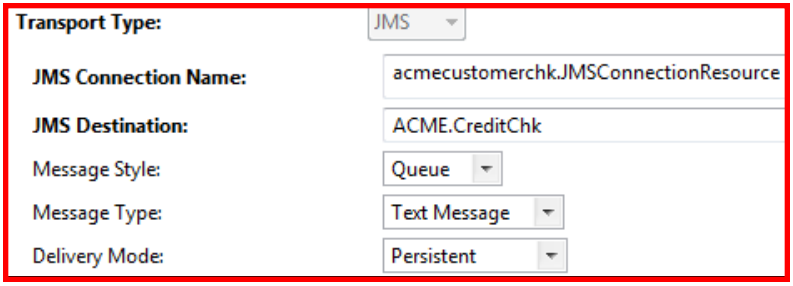
- Input tab de la Activity CheckOp
  - CustID = **`"BINDING-EXTERNAL"`**
  - CreditAmt: **1000**
- Add a Timer activity with default values
  - Connect Timer activity to Invoke activity

3. Change the reference binding to use an external binding reference

- The Reference chevron has been added to the process boundary, click on it
- In *Properties* > ***General***
  - Reference Type: **Binding Reference - External**
  - **Save** changes
- An error appears *at the module level* indicating that a binding is needed

4. Add a binding to be used with the component.

- In Project Explorer, browse to *Module Descriptors* > *Components*
- Expand *invoking.ExternalService* > ***CheckCustCredit~~PortType~~***
  - Click **Bindings**
  - Add a SOAP Binding and **Uncheck** Select/Create Required resources
  - Click **Finish**, click OK
  - For *JMS Connection*, Select the one created earlier .



Ya están definidos estos valores, sólo verificar.

5. Test the process invocation and confirm that it works.

- Launch the debugger
  - Examine the console for debug messages indicating that the process was run

```
- Started by BusinessStudio, ignoring .enabled settings.
FRWK-300021: All Application dependencies are resolved for Application
-300006: Started BW Application [ACMECustomerChk.application:1.0]
tInfo -  Received Request for AccountID : BINDING-REF
tInfo -  Received Request for AccountID : BINDING-EXTERNAL
tInfo -  Received Request for AccountID : SUB-PROC-TEST
tInfo -  Received Request for AccountID : DYN-SUB-PROC
```

- Stop the debugger
- Switch back to design perspective

Tip: Note the difference between *Binding Reference* and *Binding Reference - External.*

- For Binding Reference, the Client connection is defined at the process level.

- For Binding Reference - External, the connection is defined at the component level.

6. Export your solution as a ZIP file and shut down Business Studio.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File: `/home/tibco/bw6edu/dev/exports/`**`SubProcesses.zip`**
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created
- **Exit** Business Studio

Notes

Unit 13 Using Conversation

This unit explores the conversation and the activities that support conversation.

Topics

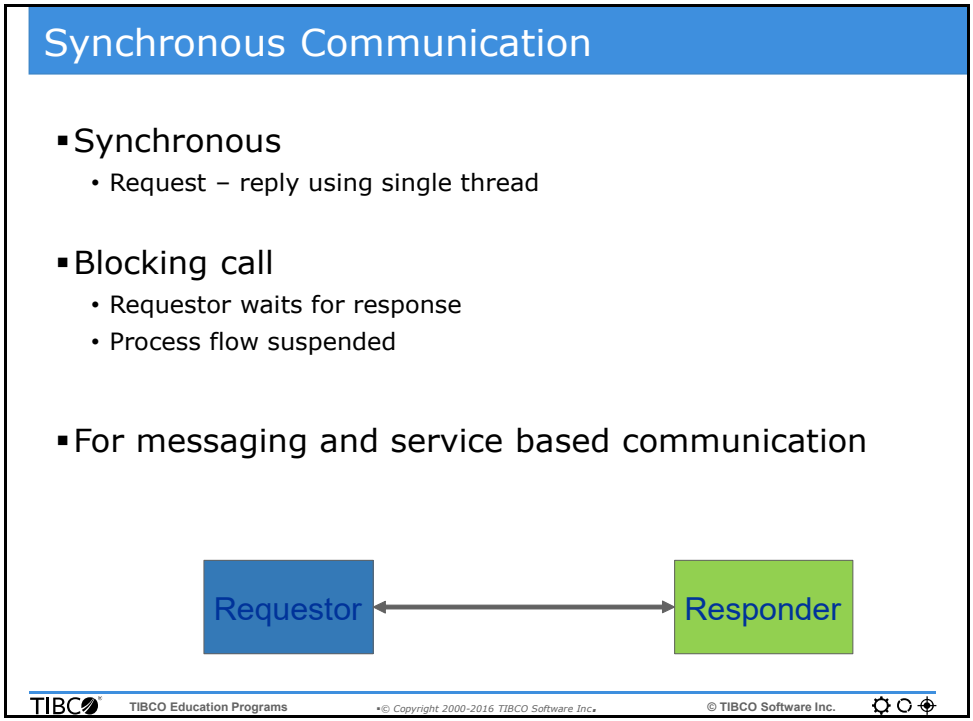
- *Objectives,* page 398
- *Synchronous Communication,* page 399
- *Asynchrounous Communication,* page 400
- *Asynchrounous Communication in Messaging,* page 401
- *Asynchronous Communication in Services,* page 402
- *Setting Up a Conversation,* page 403
- *Joining a Conversation,* page 404
- *The Conversation Tab,* page 405
- *Activities Supporting Conversations,* page 406
- *Conversation Details,* page 407
- *Interface Specification,* page 408
- *Process-Pattern Mapping to Test Conversations,* page 409
- *Review,* page 410
- *Lab Overview,* page 411

Objectives

- Knowledge
  - Compare synchronous and asynchronous communication
  - Examine details for a linking services
  - List components for a conversation
  - Identify activities that can support conversations
- Skills
  - Use conversation to process orders and handle billing requests

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Objectives



Synchronous Communication

When a BusinessWorks process communicates with another process or system using messaging (Rendezvous, FTL or EMS) or services (Web services), it supports two kinds of communication.

Synchronous : In this type of communication, a request is made and the process waits until a response is received from the responder. Process execution on the requestor process is suspended until the response is received or a time-out occurs.

This kind of processing is important when the response data is used to determine further processing.

For example: This processing is used to determine account balance for a customer. If the account balance is less than a threshold, one path may be followed, and if it more than the threshold, a different path may be taken.

This type of communication slows down overall processing, since external dependencies are now added. Processing is done on a single (logical) thread.

Asynchronous Communication

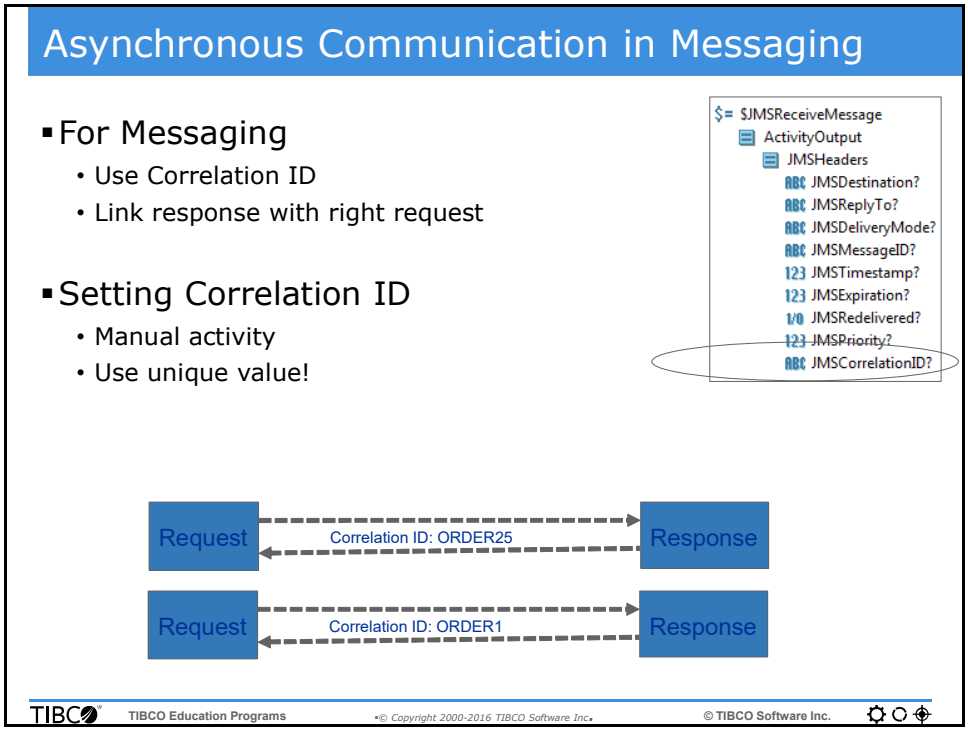
- Asynchronous Calls
  - Uses separate requestor and responder threads
- Non-blocking
  - Requestor continues processing
- Supported for messaging and service based

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Asynchrounous Communication

In this form of communication, the requester process makes a request and continues its own processing. The responder sends a reply at a later time, and this response is included in processing when received.

This is a non-blocking call, since the requester process can continue processing. An example of this kind of processing would be the request to log a customer request in a back end database. Processing is done on two logical threads: One thread makes the request and another thread processes the reply when is received.



Asynchrounous Communication in Messaging

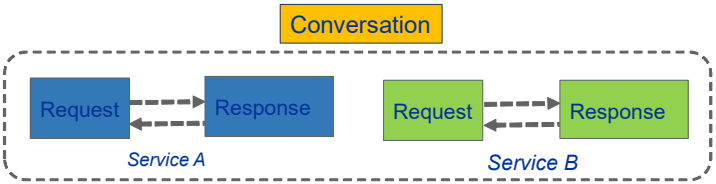
In asynchronous communication using messaging, a unique identifier called the correlation ID must be used.

When the process creates a request message, it assigns a unique identifier to the request that is different from those for all other currently outstanding requests (for example, requests do not yet have replies). When the receiver processes the request, it saves the identifier and adds the request's identifier to the reply.

When the caller processes the reply, it uses the request identifier to correlate the request with the reply. This is called a correlation id because of the way the caller uses the identifier to correlate each reply with the request. A correlation ID is usually put in the header of a message. The ID is not a part of the command or data the caller is trying to communicate to the receiver.

Asynchronous Communication in Services

- Multiple services are linked together
  - Service (request-response) is synchronous
- Linking services together
  - Use a Conversation
- Conversation ID and Key must be set

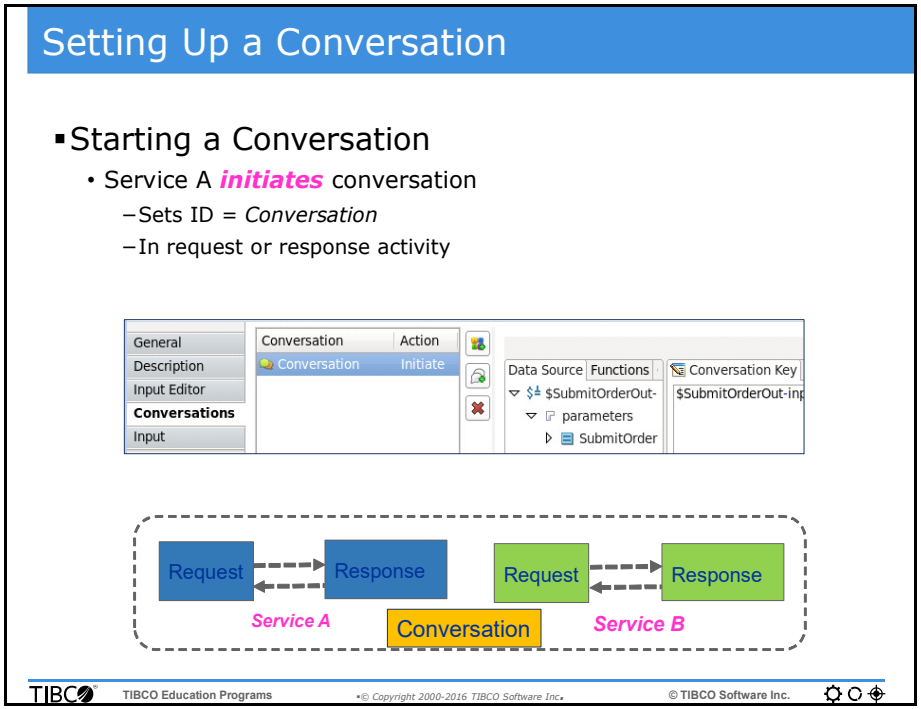


TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Asynchronous Communication in Services

Conversations are used for stateful business processes, which means that for completion, processes require correlation of message exchanges. Such processes can be reentrant and so the previous process context is maintained for continuity.

Conversations are always initiated by one activity and joined by other activities. All operations that are part of the stateful process must generate a conversation ID and reply to the original client that contains the conversation ID.

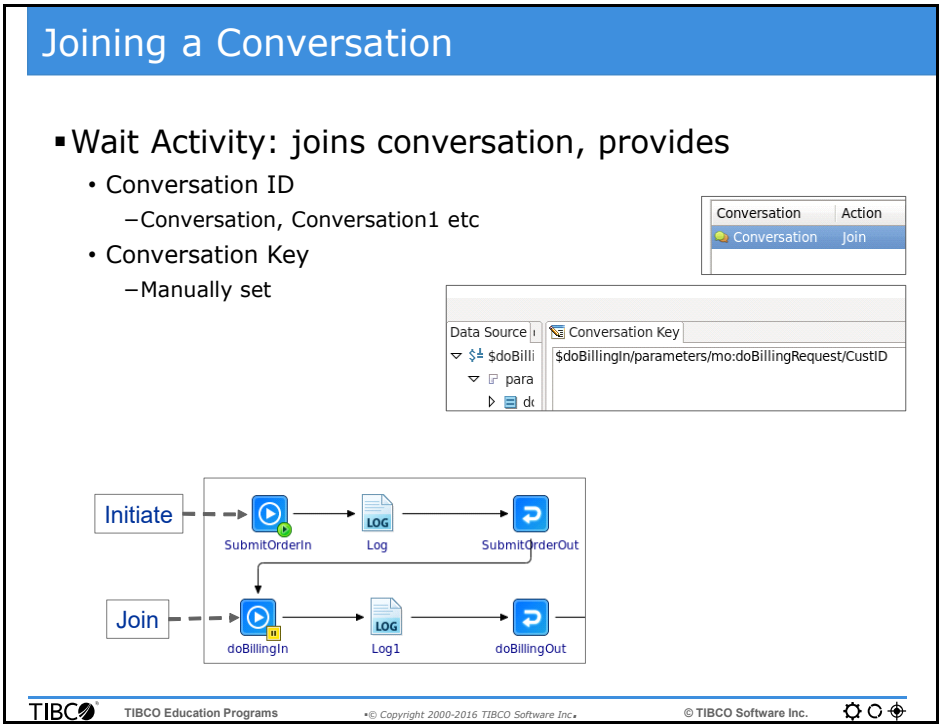


Setting Up a Conversation

When using asynchronous communication with services, the similar functionality is accomplished by using conversations.

Note however that a single conversation can contain multiple, separate request-reply communications as shown above. To implement a conversation, an activity (request or reply) initiates a conversation. A conversation can be started at any time (unlike in messaging) and any number of services (request-replies) can join the conversation.

The conversation is complete when the process (job) execution is complete.



Joining a Conversation

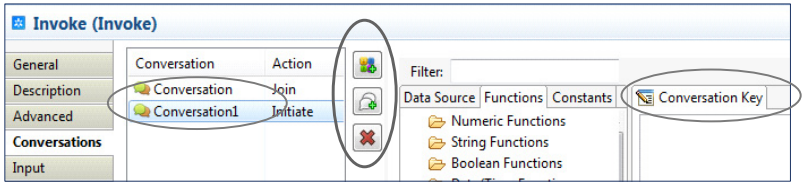
One more aspect of a conversation is how to join a conversation. A conversation is started and completed within a process execution path.

One service initiates a conversation. Of another service wants to join a conversation, it must use a wait activity to join the conversation. The second service cannot use a request activity, since it is a process starter, and only one process starter is allowed per process.

Thus, if multiple services join a conversation, they must all join with Wait activities.

The Conversation Tab

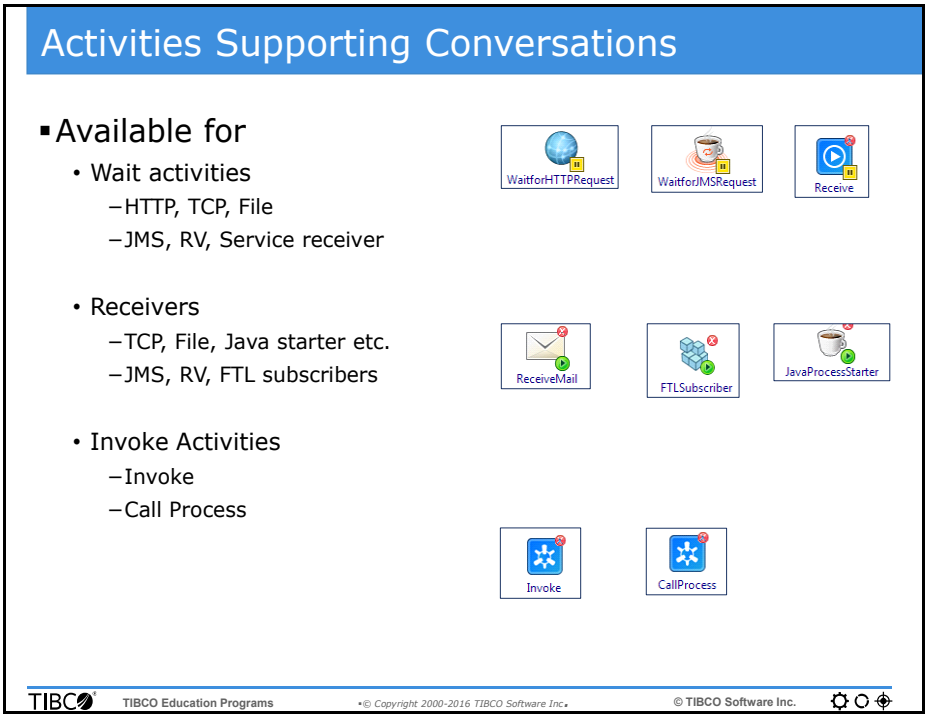
- Conversation details
  - Initiate or Join
    - One activity can have multiple conversations
  - Conversation ID
    - ***Conversation***, ***Conversation1*** and more
  - Conversation Key
    - Set by you



TIBCO Education Programs • © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

The Conversation Tab

The conversation Tab is used to define a unique value for a conversation.
By default conversations are named Conversation, Conversation1 and so on.
If you add a new conversation, and the name displayed is Conversation2, it implies that there are two more conversations (Conversationb and Conversation1) which have been defined for this process.
To define a unique key for a conversation, a combination of process id and time stamp is deemed unique enough for most scenarios. Use the xpath functions to define a conversation key.

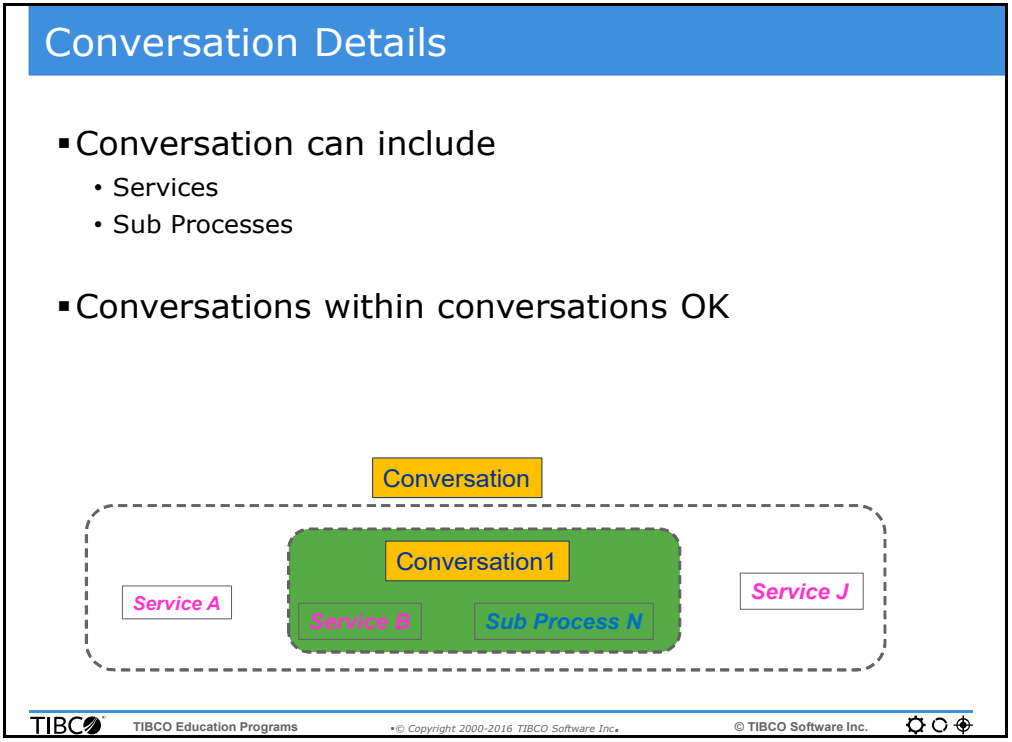


Activities Supporting Conversations

When designing an application, better support for conversations can be achieved by defining the following:

Number of conversations that a process participates in

- Definition of the activities that are part of the same conversation
- Sequence of messages for each conversation.

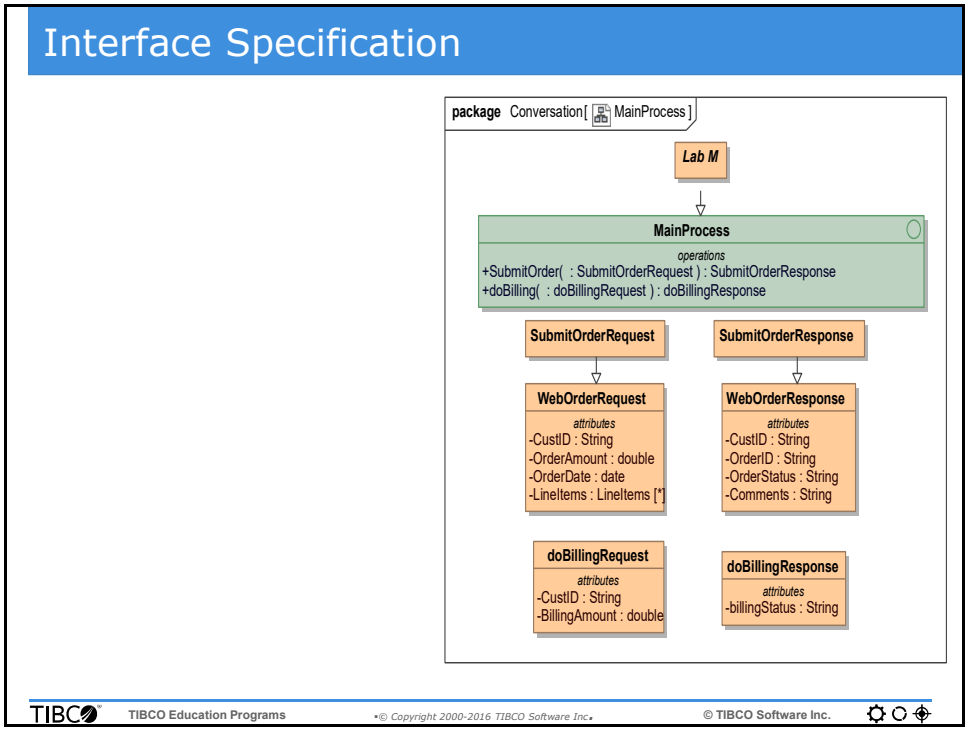


Conversation Details

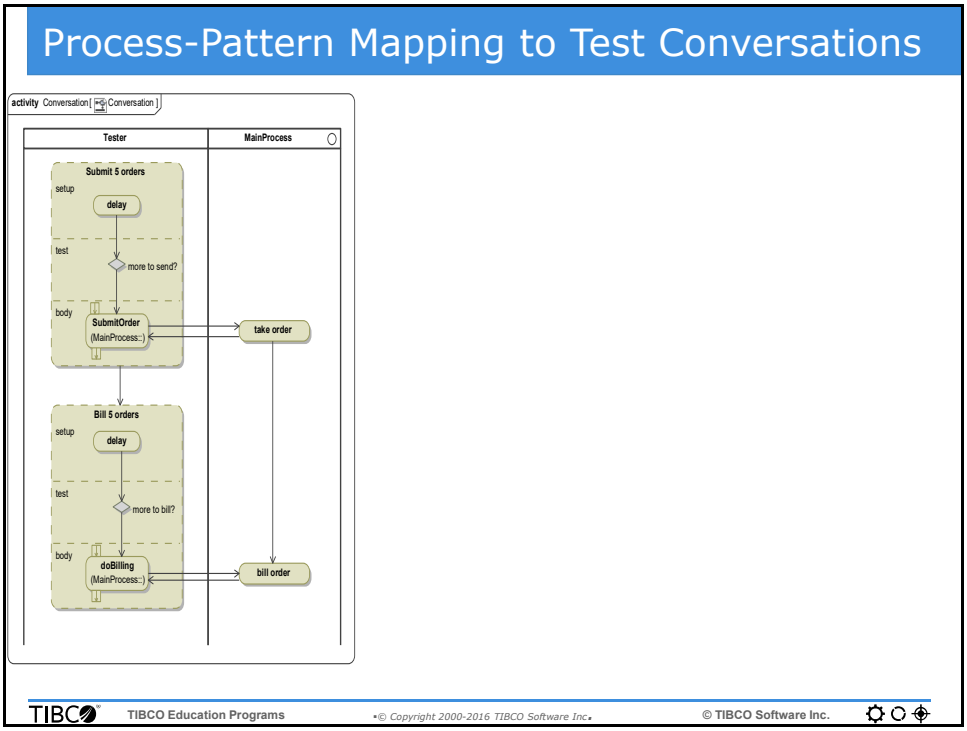
A conversation can span the duration of a complete process. This includes any sub-processes and services that are invoked by the process.

Each conversation operates within its own boundaries. As such, conversations within conversations are allowed, since they do not know of each other. These are not hierarchical, they are simply independent conversations which run in parallel for some duration of execution.

When designing conversations within conversation, be careful not to mix up the conversation names or key values used by them.



Interface Specification



Process-Pattern Mapping to Test Conversations

Review

- Synchronous communication uses blocking calls
- Asynchronous communication uses two threads
- Messaging uses correlation IDs for async calls
- Conversation: Links services
- Conversation has ID and Key

TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Define Web Order Process
  - Add order processing
  - Define operation for billing
  - Create conversation to link services
- Create tester process
  - Submitting billing requests in reverse order
  - Test conversations

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab M Use Conversation

Overview

In this lab you will update the existing application module to implement Web Order processing . You will also configure it to be part of a conversation.

Objectives

- *Define a Conversation,* page 414
- *Test Conversation,* page 420

Prerequisites

- The following TIBCO Software is installed and configured on your educlt machine:
  - BusinessWorks 6.3.1
- Completion of previous labs

Directions

Complete the exercises that follow.

Lab M - Exercise 1: Define a Conversation

Overview

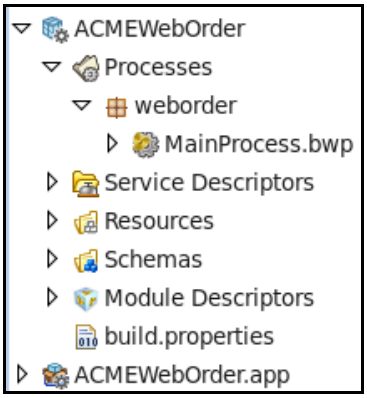
In this exercise you import an existing service and build a conversation using this service.

Steps

1. Open your workspace and import the ACMEWebOrderV1 project.

- In TIBCO Business Studio, select *Applications Menu* > *TIBCO* > *BW6* > ***BusinessWorks Studio***
- Open workspace: **`/home/tibco/bw6edu/workspace`**
- Close Welcome tab
- Choose *File* > ***Import***
  - Select *General* > *Existing Studio Projects into Workspace,* click **Next**
  - Choose **Select Archive File**
  - Browse and select `/home/tibco/bw6edu/dev/Others/ACMEWebOrderV1.zip/`
  - Confirm all projects are selected and click **Finish**
- Confirm both Application module and Application are imported

- Expand *ACMEWebOrder* > *Processes* > ***weborder***



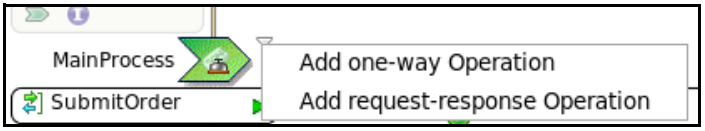
2. Examine the SubmitOrder process.

- Open ~~**MainProcess.bwp**~~**,** a starter service with a single log activity

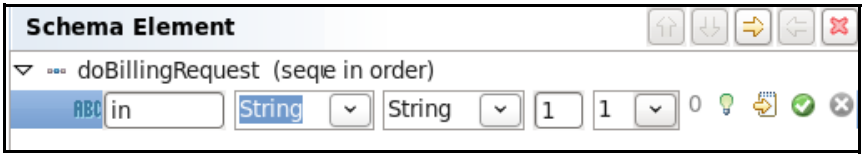
MainProcess.bwp

3. Add a new operation named doBilling in ~~NewOrderProcess~~, and set its input.

- Locate the *Service* icon for the process (large green chevron) and click on the **dropdown arrow**
  - Choose **Add request-response Operation**



- Verify a new operation named *operation* is added, click on **operation**
- Configure operation Properties >
  - **General tab:** Click the Light Bulb, rename to **`doBilling`** El nombre podría no cambiar. Salir y regresar al campo para ver el cambio.
  - **Input Tab**
    - Click **doBillingRequest** under Operation Input (left side of properties tab)
    - Double click **in** element in Schema Element (right side window) to edit

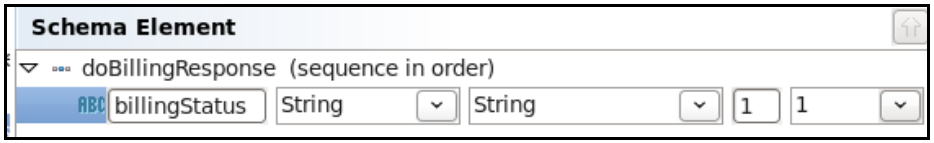


    - Change name from *in* to **`CustID`** and hit <enter>
    - Now select doBillingRequest and then click **Add Primitive Element** icon (circled)

  - Element Name: **`BillingAmount`**, Type: Decimal (Double) and hit enter
- **Confirm** that there are two elements in the Input

4. Set the output for doBilling to return BillingStatus.

- Choose the **Output** tab
  - Rename the **out** element to **`billingStatus`**

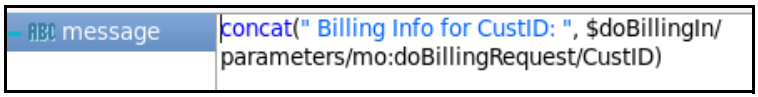


  - **Save** changes

5. Implement the doBilling operation.

- Under the Service chevron, right-click **doBilling** then Operation > Implement Operation
  - Create **transition** from *SubmitOrderOut* to *doBillingIn*
  - Add **Log Activity** between *doBillingIn* and *doBillingOut*

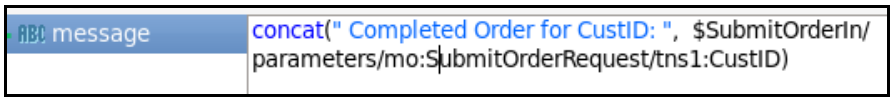
    (Select Log activity from palette and drop it on the connection)
- Configure and Map the **Log1** activity input as shown



- Configure **doBillingOut** properties input tab
  - billingStatus: **`"Success"`**

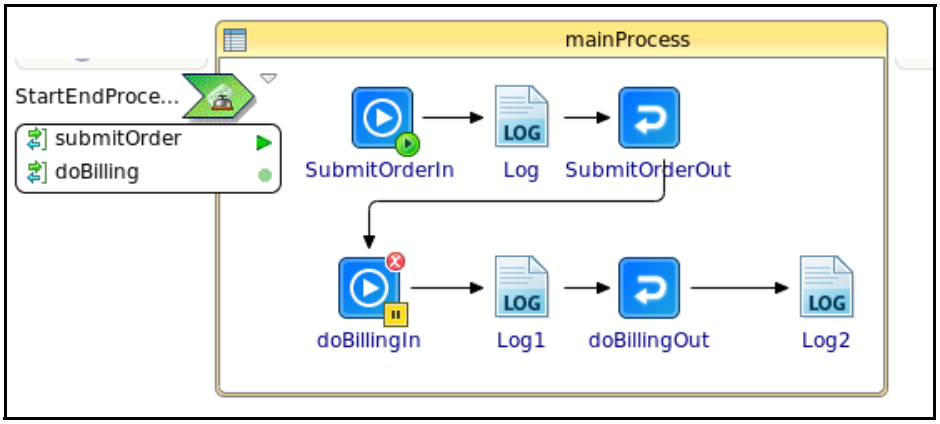
6. Add a log activity at the after billing response.

- Add **Log** activity after doBillingout
- Configure input:



7. Resolve the error with billing service by defining a conversation.

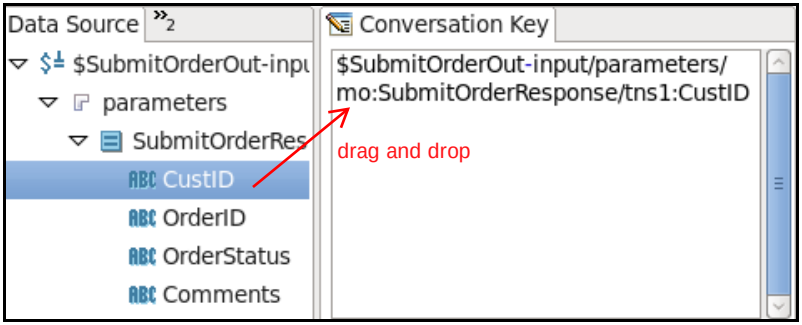
- **Save** changes and **verify** that there is one error with *doBillingIn*



- **Hover** over the red 'X' error symbol and confirm the Conversation configuration error

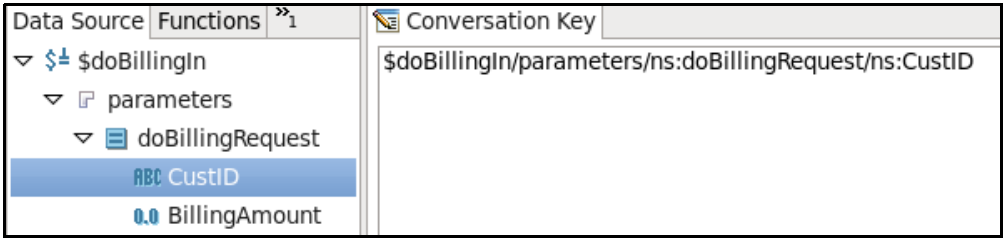
8. Update SubmitOrderOut to initiate a conversation.

- Select ***SubmitOrderOut*** > *Properties* > *Conversations*
- Click on **Add New Conversation** icon
- Select new conversation called **Conversation**
- Map **CustID** to Conversation key



9. Update doBillingIn to join the conversation with the same key.

- Select **doBillingIn** activity
- Under Properties, select **Conversations** tab
- Click on **Join Existing Conversation** icon to automatically add conversation
- Select conversation called **Conversation**
- Map **CustID** to Conversation key



- **Save changes** and confirm there are no more errors

Lab M - Exercise 2: Test Conversation

Overview

In this exercise you run the application and confirm that the conversation behaves as expected.

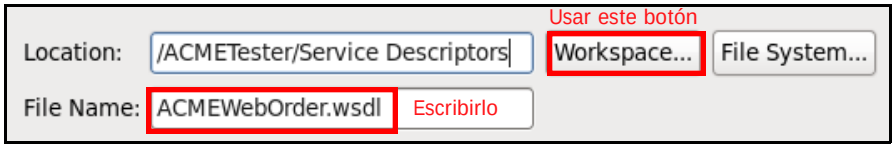
Steps

1. Copy schema files to ACMETester.

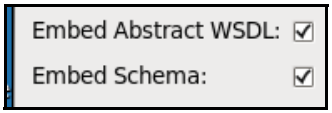
- Browse to *ACMEWebOrder* > ***Schemas***
  - **Copy** *MainProcessSchema.xsd* and *WebOrderSchema.xsd*
  - Now browse to *ACMETester* > ***Schemas***
  - Right-click and choose **Paste BW**
  - Confirm both files are copied

2. Generate the concrete WSDL for the ACMEWebOrder service

- In Project Explorer, open *ACMEWebOrder* > *Module Descriptors* > ***Components***
  - Expand and select *ComponentMainProcess* > *MainProcess* **> *SOAPServiceBinding***
  - In *Properties tab* > ***Binding tab***, click **Generate Concrete WSDL**
  - Enter details as shown below:



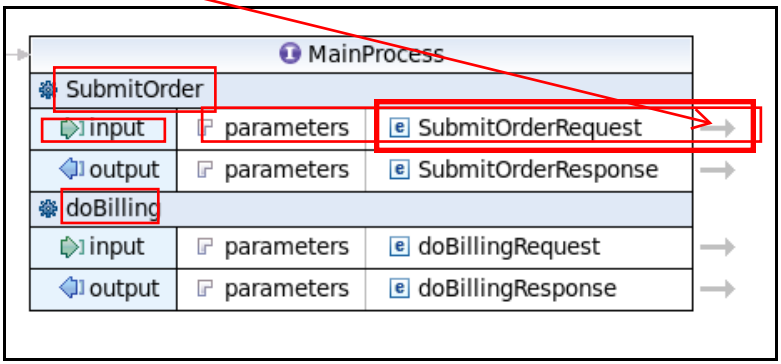
  - Click **Next**
  - **Checkmark** Embed Abstract WSDL and Embed Schema



  - Click **Finish**
- **Verify** the ACMEWebOrder.wsdl file is generated and opened

3. Correct the error in the WSDL file.

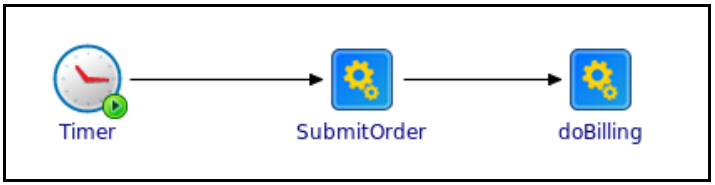
- Click the grey arrow to the right of the SubmitOrderRequest



- Click the **Show schema index view** button in the upper left of the Inline schema view
- In Directives click to highlight **No Location Specified**
- In *Properties* > ***General*** browse to set the Schema Location
  - Select **Workbench projects** and click **Next**
  - Select *ACMEWebOrder* > *Schemas* > ***WebOrderSchema.xsd***
  - Click **Finish**
- **Save** Projects
  - Select *Project* > ***Clean***
  - Select **Clean all projects**
  - Click **OK**

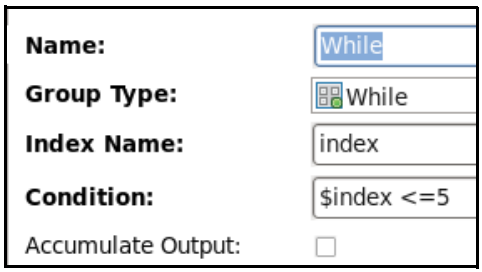
4. Define a tester process in ACMETester module.

- In Project Explorer, right click *ACMETester* > ***Processes*** and select *New* > ***BusinessWorks Process***
- Name the new process **`WebTester`** and click **Finish**
- From *ACMETester* > ***Service Descriptors***
  - Drag **ACMEWebOrder.wsdl** to the WebTester process
  - Select **Invoke Operation**, then select click **SubmitOrder** Click OK
- From the Basic Activities palette select **Invoke** activity
  - Click to add **Invoke** activity to the process
  - In *Properties* > *General*
    - Name: **`doBilling`**
    - Choose Service: **MainProcess**
    - Operation: **doBilling (**select from the drop-down list**)**
  - Add a **Timer** activity to trigger the process and **Connect** the activities as shown:



5. Define a loop around SubmitOrder to submit multiple orders.

- Right-click SubmitOrder and choose **Create Group > While**
  - The group is added to surround the activity
  - Configure the group as shown below:



  - **Save** changes

6. Update the inputs to be provided to the submitOrder.

- For *SubmitOrder*, configure the input as shown:

SubmitOrder-input	
SubmitOrderRequest	
parameters	
SubmitOrderRequest	
CustID	concat("CUST", xsd:string($index))
OrderAmount	100
OrderDate	current-date()
LineItems	
ItemSKU	"TEST"
Qty	1

Tip: We will submit 5 orders with different customer IDs, to see a sequence of incoming messages

If you do not see the schema elements, it means that the WSDL cannot find the imported schemas. Open MainProcess.WSDL and fix the import location for the Schemas

- **Save** changes

7. Configure a group activity for doBilling using a repeat group

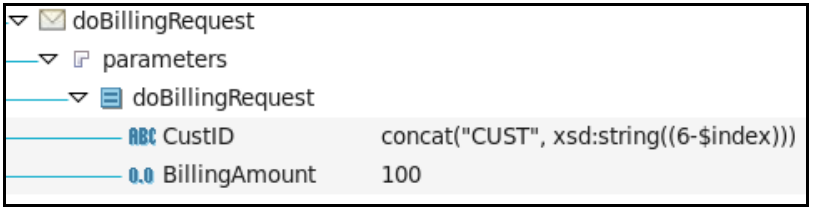
- Right-click on *doBilling* then choose *Create Group* > ***Repeat***
- Configure the group as shown below

Name:	Repeat
Group Type:	Repeat
Index Name:	index
Condition:	$index>=5
Accumulate Output:	☐

- **Save** changes

8. Update the callWebOrder activity to invoke billing requests.

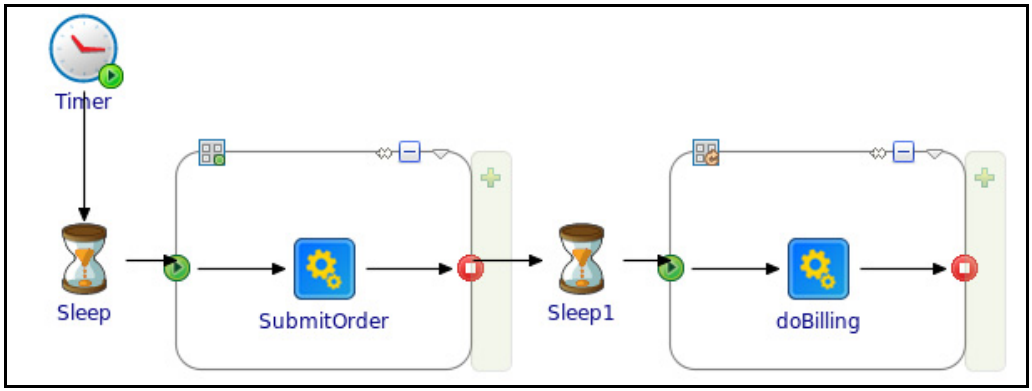
- For *doBilling* > *Input*, configure as follows :



Tip: We will complete the billing requests in *reverse order* to demonstrate that the process can be completed out of sequence

9. Add sleep activities to delay the two operations.

- Add a **Sleep** activity between Timer and submitOrder
  - For input, Configure intervalMillisec: `8000`
- Add a **Sleep** activity between the two Invoke Service calls
  - For input, Configure intervalMillisec: `8000`
- **Verify** that the final process definition looks as shown below



- **Verify** there is an error with the reference chevron (on the right side of the process)

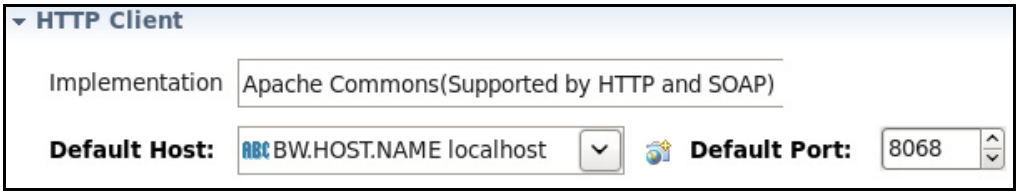
10. Configure the reference details to invoke the external services.

- Click on the **reference chevron** containing the two operations (*submitOrder* and *doBilling*)
- In *Properties* > ***General***, set Reference Type: **Binding Reference - External**
  - **Save** Changes and **verify** an error appears in ~~*Project Explorer > Module Descriptors*~~ Project Explorer > ACME Tester > Module Descriptors
- Open *ACMETester* > *Module Descriptors* > ***Components***
  - Expand and select *ComponentWebTester* > ***MainProcess***
  - In *Properties* > *Bindings*, click **Add Binding**
  - Select **SOAP binding** and click **Finish**
- Expand Main Process and select the new **SOAPReferenceBinding**
- In Properties, scroll to *Transport Configuration*
  - Set Transport Type: HTTP Ya está este valor (grayed out)
  - Client Name: click *browse* (...)
  - Click ***Create Shared Resource*** Button,
    - Resource Name: **WebOrderClient** Click Finish

  - **Save** changes

11. Confirm settings for the newly created HTTP Client.

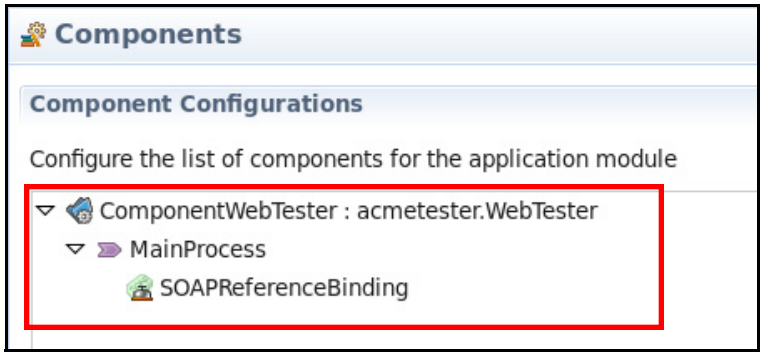
- Browse to and open *ACMETester* > *Resources* > *acmetester* > ***WebOrderClient.httpClientResource***
- For Default Host select the **Choose field value type** icon
  - Select **Module property**
- Set the following values
  - Default Host: **BW.HOST.NAME** (module property)
  - Default Port: `8068`



- **Save** changes and close this view

12. Update the ACMETester application to invoke new web order processing only.

- Browse to *ACMETester* > *Module Descriptors* > ***Components***
- **Verify** only the following component is selected:

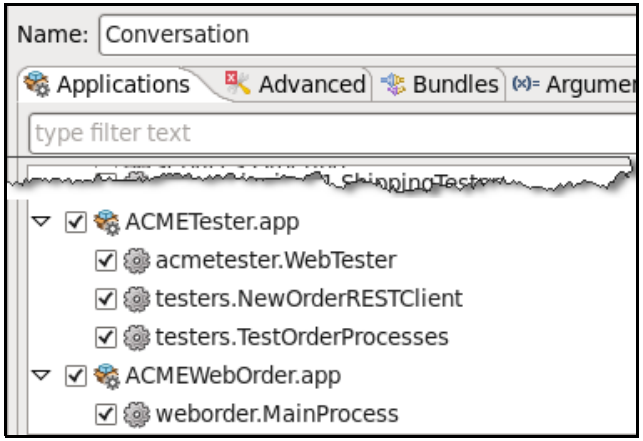


Dejar sólo este Componente. Eliminar adicionales con el botón "X" a la derecha.

- Delete any additional components that may reside

13. Define a new run configuration to run ACMETEster and ACMEWebOrder only.

- Select *Run* > ***Run Configurations..***
- Select *BusinessWorks Application* > ***BWApplication***
- Rename the Application **`Conversation`** and define a new run configuration as shown below:



- **Apply** changes
- Click **Run** to start the applications

14. Observe processing and confirm that conversations work as expected.

- **Observe** the console for messages.

```
6: Started BW Application [ACMETester.app:1.0]
.Log -  *** Submit Order for CustID: CUST1***
.Log -  *** Submit Order for CustID: CUST2***
.Log -  *** Submit Order for CustID: CUST3***
.Log -  *** Submit Order for CustID: CUST4***
.Log -  *** Submit Order for CustID: CUST5***
.Log1 -  Billing Infor for CustID: CUST5
.Log2 -  Completed Order for CustID: CUST5
.Log1 -  Billing Infor for CustID: CUST4
.Log2 -  Completed Order for CustID: CUST4
.Log1 -  Billing Infor for CustID: CUST3
.Log2 -  Completed Order for CustID: CUST3
.Log1 -  Billing Infor for CustID: CUST2
.Log2 -  Completed Order for CustID: CUST2
.Log1 -  Billing Infor for CustID: CUST1
.Log2 -  Completed Order for CustID: CUST1
```

Analysis: Execution flow is as follows:

- Orders are submitted in Sequence: Cust1, Cust2 etc
- There is a brief pause
- Billing information is submitted and completed in reverse sequence

- **Stop** the Tester when done
- Shut down Business Studio

15. Export your solution as a ZIP file and shut down Business Studio.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File: `/home/tibco/bw6edu/dev/exports/`**`Conversations.zip`**
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created
- **Exit** Business Studio

Notes

Notes

Unit 14 Implementing Event Handlers

This unit describes the purpose of event handlers and the usage of multiple handlers.

Topics

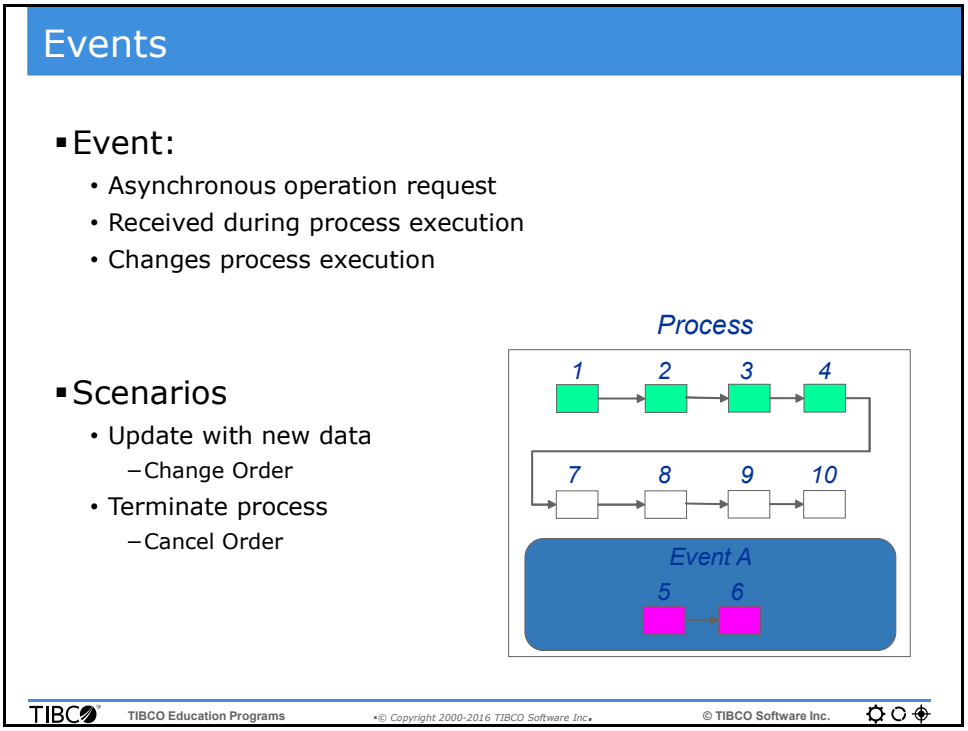
- *Objectives,* page 432
- *Events,* page 433
- *Event Handlers,* page 434
- *Changing Event Handler Scope,* page 435
- *Multiple Event Handlers,* page 436
- *Defining an Event Handler,* page 437
- *Adding Multiple Event Handlers,* page 438
- *Event Handler Business Logic,* page 439
- *Interface Specification - Lab N,* page 440
- *Process-Pattern Mapping to Test Event Handler,* page 441
- *Review,* page 442
- *Lab Overview,* page 443

Objectives

- Knowledge
  - Explain purpose of event handlers
  - Discuss scope of an event handler
  - Examine scenarios for multiple handlers
  - Describe use of an exit activity
- Skills
  - Use Event handler to implement order cancellation

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives

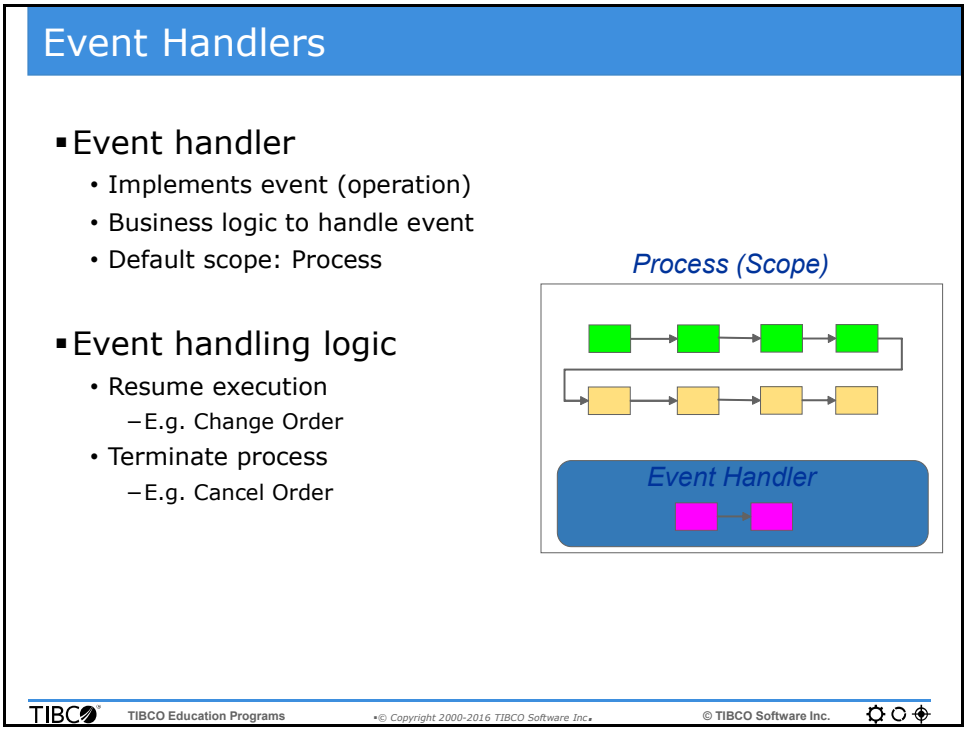


Events

Recall that asynchronous processing is like running two logical threads - one for request and one for handling the response.

In the earlier unit, you examined the usage of the wait activity, which is used in conversations. Even though conversations and wait activity are used in asynchronous processing, this is not 2-threaded processing. The wait activity causes the process to wait until the activity times out or the response.

To accomplish true 2-threaded processing, you must use an event handler. An event arrives anytime during the process execution and must be handled appropriately. An event handler can be thought of as a separate thread which simply waits for the appropriate event. Once the event is received, the processing moves to the activities defined by the event handler. After the event handler completes execution, processing shifts back to the main process.

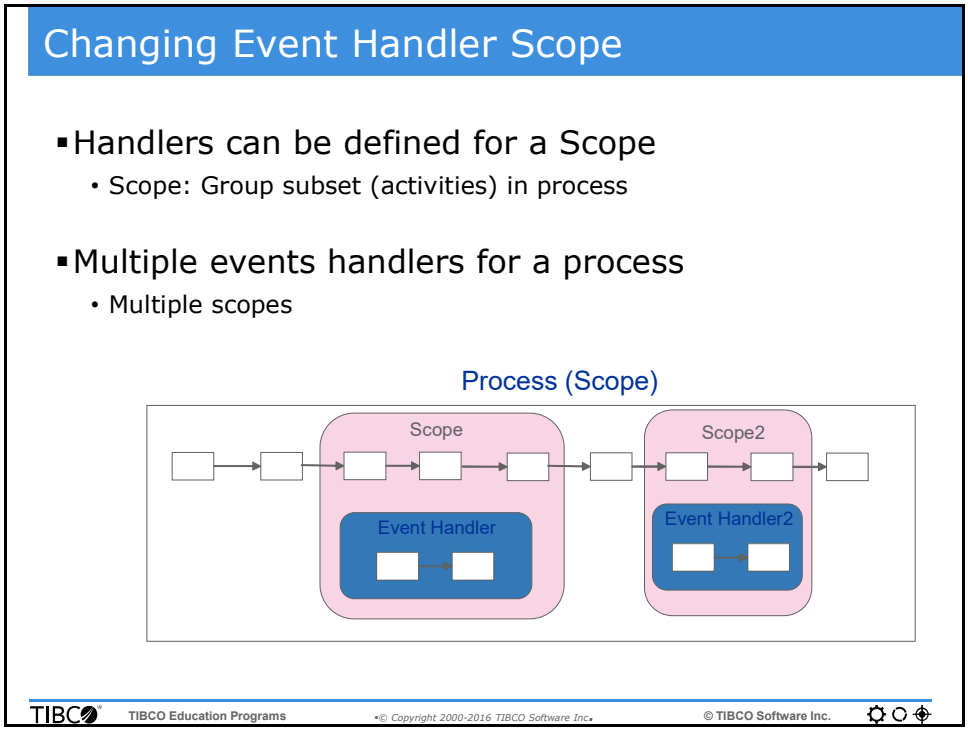


Event Handlers

Event handlers allow asynchronous event processing. They are always attached to a scope and execute parallel to the main business logic of the process; so they are associated with an operation that is a part of the process.

Each event handler is associated with one process and has access to that process. An event handler can be executed multiple times during the process execution.

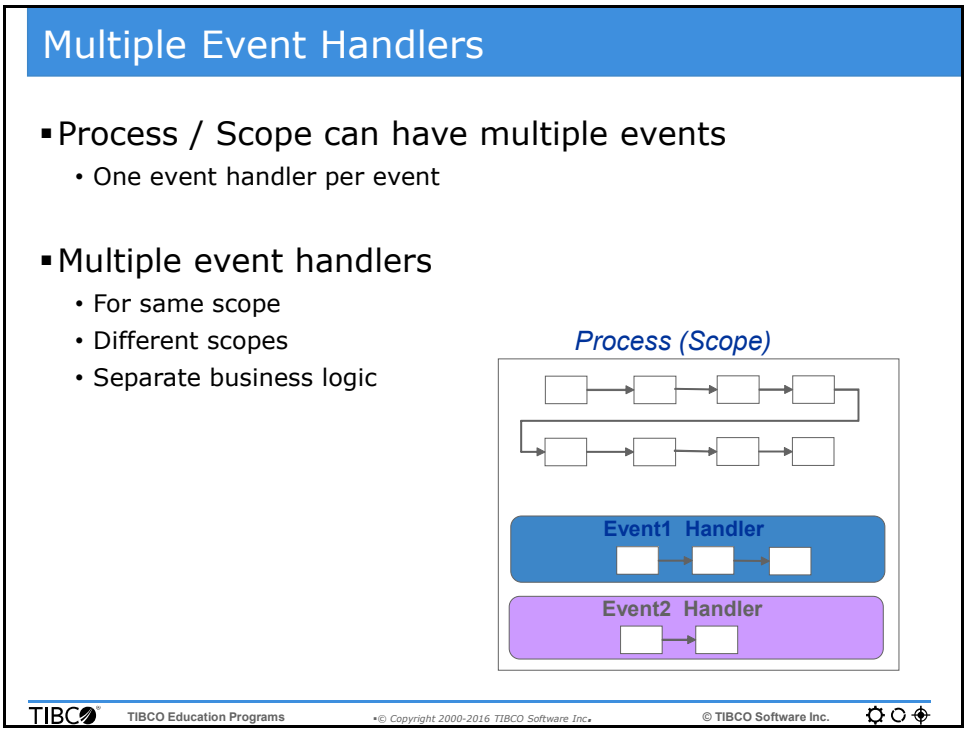
The event handler must specify the scope (i.e when the event handler is active) and the event it listen for during that scope.



Changing Event Handler Scope

Event handlers can be defined on two different levels:

- Process level - When defined on the process level, it cancels the job.
- Scope level - When defined on the scope level, it typically cancels the scope. Instead, it can be configured to timeout.



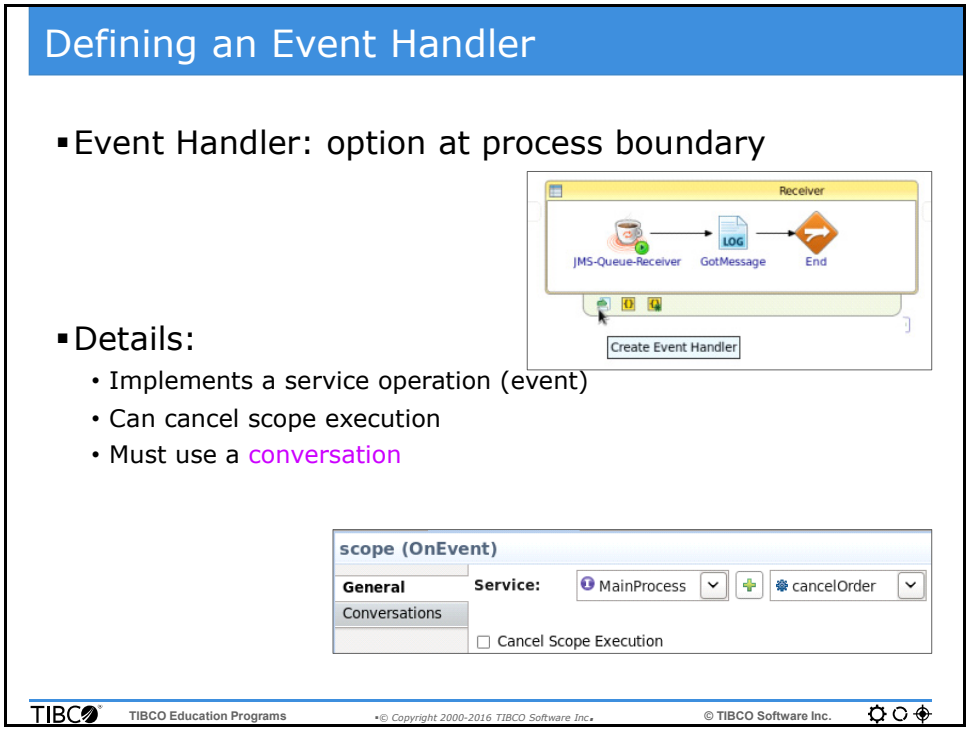
Multiple Event Handlers

Multiple event handlers can be defined for the same scope and event combination.

This may be necessary of you need to execute different business sets of business logic when an event arrives.

For example: If an order cancellation event arrives, you may want to update the order database. You may also want to do some processing with a third-party system.

All event handlers are activated upon arrival of the event, and executed in sequence. The event handler defined first is usually executed first, followed by the second one and so one.



Defining an Event Handler

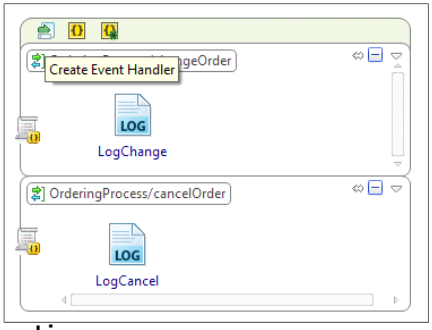
An event handler is defined for a scope. For an event handler with a process scope, simply click the bottom of the process boundary to see the floating menu as shown. There are three icons in the floating menu - the event handler, exception handler and the catch-all exception handler. Chose the first icon to define an event handler for the process.

The second way to define an event handler is to create a scope by defining a group around a set of activities. After that, define the event handler in the same manner as shown above.

An event handler is a service operation invocation. So, in the event handler, define the service and operation is associated with this event.

Adding Multiple Event Handlers

- Adding to an existing handler
  - Choose link on existing
- Add a new scope
  - Add handler to scope



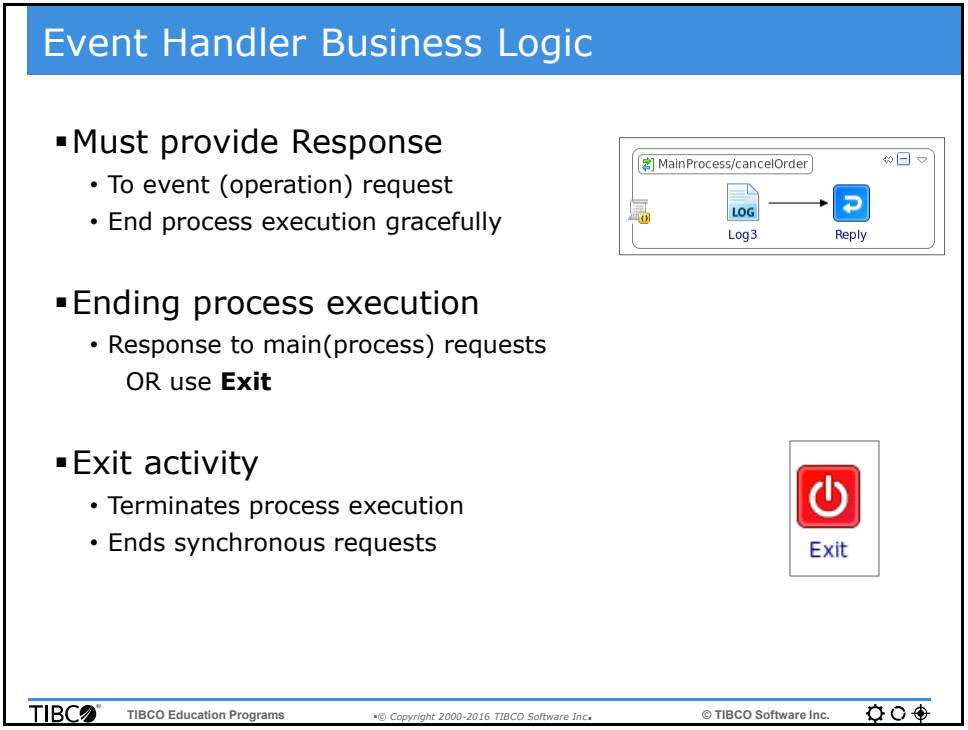
- 1:1 = event handler to operation
- 1:many = scope to event handlers

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Adding Multiple Event Handlers

Multiple event handlers can be defined for the same scope.

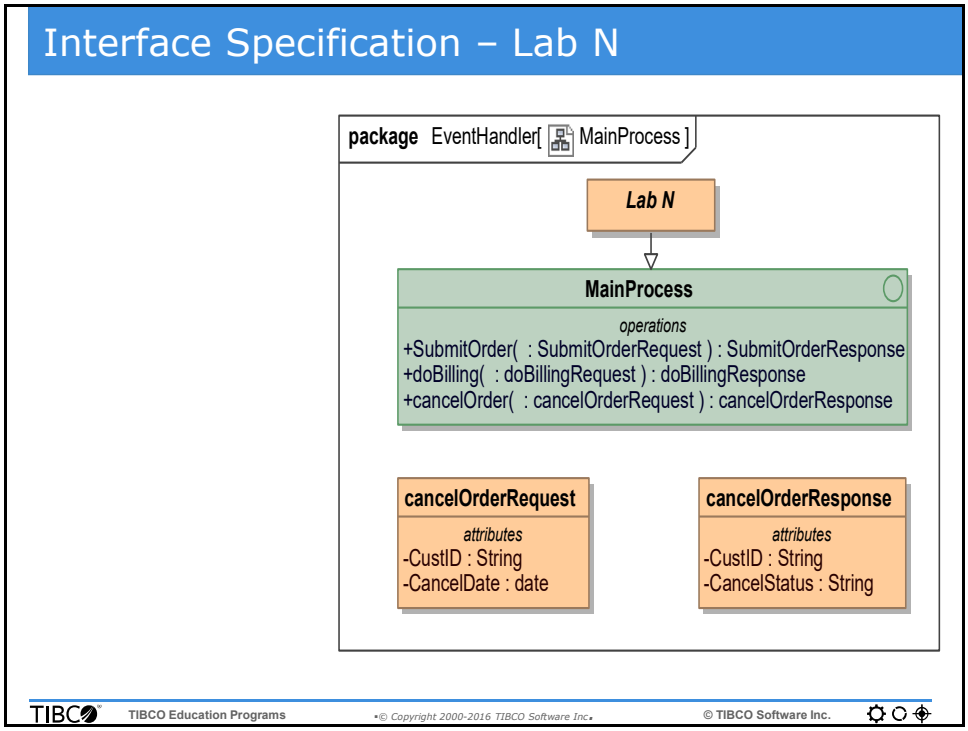
Remember that each event handler maps to exactly one service operation, and each service operation must map to exactly one event handler. So to define multiple event handlers, you must have multiple service operations available. If you violate these rules, an error will be returned by BusinessStudio.



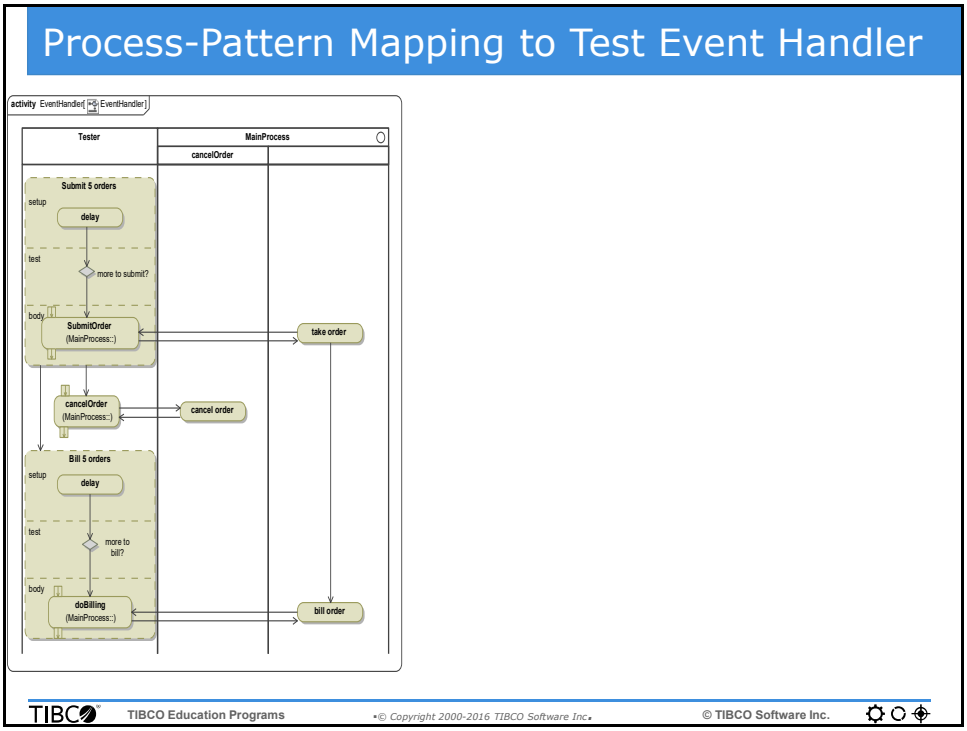
Event Handler Business Logic

An event handler is a SOAP request-response service. So the event handler implementation must provide a response to the invocation request, in addition to the business logic to be executed. Once an event handler provides a response to its invocation, process execution returns to the main process. If you want to stop executing the main process at this time, you can provide a response to the main process request as well.

You can also use the exit activity to end process execution. The exit activity is equivalent to the System.exit() call in a Java program. Process execution is stopped and any running conversations are also terminated. Due to this, be careful when you use the exit activity.



Interface Specification - Lab N



Process-Pattern Mapping to Test Event Handler

Review

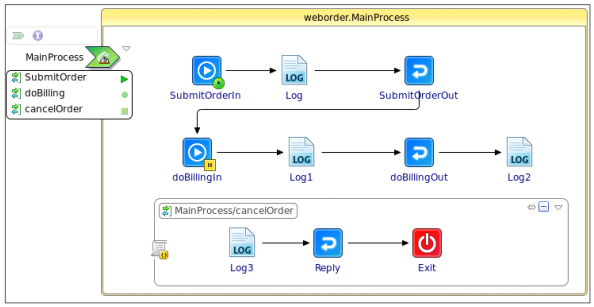
- Event: Operation that interrupts process flow
- Event handler: Logic executed on event
- Multiple event handlers allowed for a scope
- 1:1 event (operation) to event handler
- Event handler must use conversation

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Implement order cancellation logic
  - Define operation for cancellation
  - Add event handler
  - Test



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab N Define Event Handlers

Overview

In this lab you will update the conversation example and add event handlers to it.

Objectives

- *Add Event Handler,* page 446

Prerequisites

- The following TIBCO Software is installed and configured on your educlt machine:
  - BusinessWorks 6.3.1
- Completion of previous labs

Directions

Complete the exercises that follow.

Lab N - Exercise 1: Add Event Handler

Overview

In this exercise, you will define another event handler to deal with order cancellations.

Steps

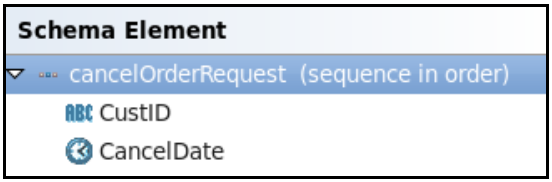
1. Open your ACMEWebOrder application module and add new operation.

- In *Business Studio*, open workspace: **`/home/tibco/bw6edu/workspace`**
- Open *ACMEWebOrder* > *Processes* > *weborder* > ***MainProcess.bwp***
- Locate the Service icon for the process and click on the down arrow (top left corner)
  - Choose **Add request-response Operation**

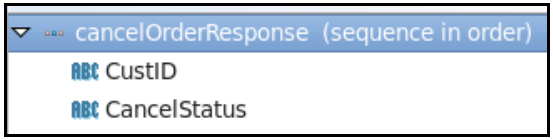
  Note: New operation named *operation* is added .

2. Configure cancelOrder operation

- Choose operation, then Properties >
  - *General* tab: Click the **Light Bulb**, rename to **`cancelOrder`**
  - *Input* Tab
    - Locate and select **cancelOrderRequest** under parameters (left side of input window)
      - Double-click *in* parameter and rename to **`CustID`** on the right side
      - Select **cancelOrderRequest** on the left side and click **Add a primitive element** on the right side
      - Rename primitive element to **`CancelDate`**
      - Set type to **Date&Time** and subtype to **DateTime**



  - *Output* Tab*:*
    - Similarly configure the output fields as shown (**`CustID, CancelStatus`** of type **String**)

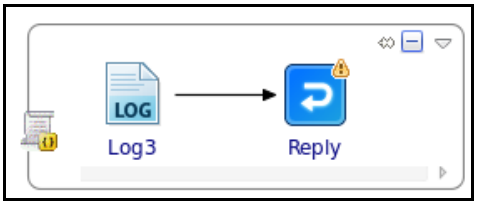


- **Save** changes

3. Implement the cancelOrder operation by adding an event handler.

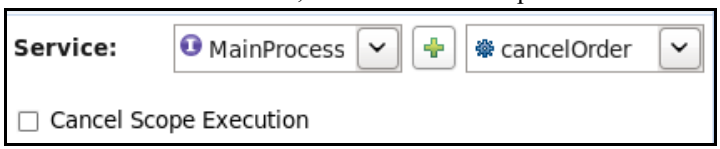
- **Hover** on lower boundary of process a floating menu appears

  - Click on **Create Event Handler** icon
- Add new **Log** Activity and a **Reply** activity and create a transition between them

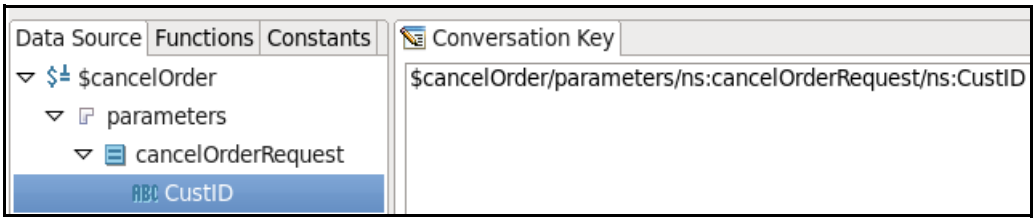


4. Configure the event handler with the correct operation and conversation.

- Click on the **Event Handler** icon on the left side of the Event Handler area
- General Tab:
  - Select the **MainProcess** service, and **cancelOrder** operation

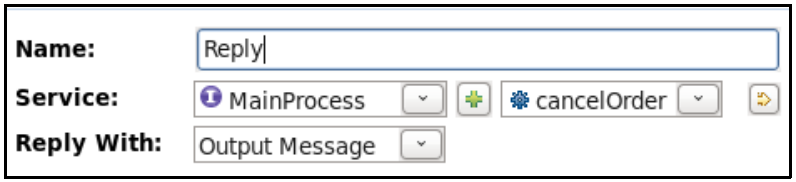


- Conversations tab:
  - Click **Join Existing Conversation**
  - Map **CustID** to Conversation Key area as shown:

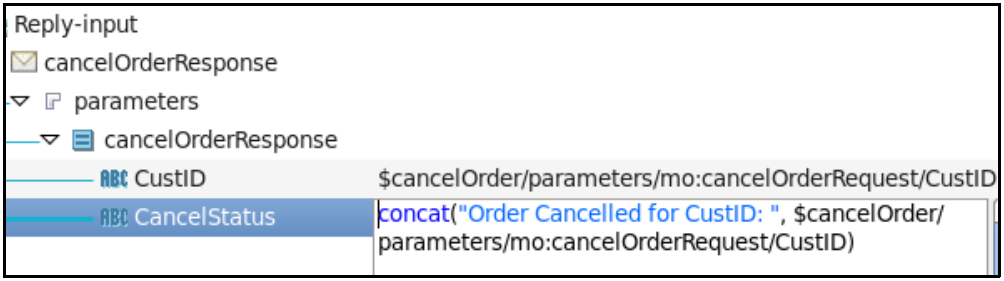


5. Configure the Reply activity.

- Select the **Reply** activity
- *General* Tab:
  - Configure *Service* and *operation*



- *Input* tab:



6. Complete input for the log activity to note incoming customer ID.

- Select the new **Log** activity
- Configure the new Log activity input tab as follows:

- **Save** changes

7. Verify Changes to MainProcessSchema.xsd.

- Open *ACMEWebOrder* > *Schemas* > ***MainProcessSchema.xsd***
- **Verify** the presence of the *cancelOrderRequest* and *cancelOrderResponse* Elements
- In *Directives* click to highlight **No Location Specified**
- In *Properties* > ***General*** browse to set the Schema Location
  - Select **Workbench projects** and click **Next**
  - Select *ACMEWebOrder* > *Schemas* > ***WebOrderSchema.xsd***
  - Click **Finish**
- **Save** Changes

8. Recopy schema file to ACMETester.

- Browse to *ACMEWebOrder* > ***Schemas***
  - **Copy** *MainProcessSchema.xsd*
  - Now browse to *ACMETester* > ***Schemas***
  - Right-click and choose **Paste BW**
  - Set the file name to **`MainProcessSchema`** in the popup window

9. Regenerate the ConcreteWSDL for the ACMEWebOrder service.

- Open *ACMEWebOrder* > *Module Descriptors* > ***Components***
  - Browse to *ComponentMainProcess* > ***MainProcess***
  - In *Properties* > ***Binding tab***
- Click on **Generate Concrete WSDL** link
  - Location: `ACMETester/Service Descriptors`
  - Name: **`ACMEWebOrder.wsdl`**
  - **Verify** the warning about overwriting the existing WSDL at the top of the Dialog box
  - Click **Finish**

10. **Verify** the newly generated WSDL.

- **Verify** the newly generated WSDL is open
- Click the **grey arrow** to the right of the *SubmitOrderRequest*
  - Click the **Show schema index view** button in the upper left of the Inline schema view
  - In Directives click to highlight **No Location Specified**
  - In *Properties* > ***General*** browse to set the Schema Location
  - Select **Workbench projects** and click **Next**
  - Select *ACMETester*> *Schemas* > ***WebOrderSchema.xsd***
  - Click **Finish**

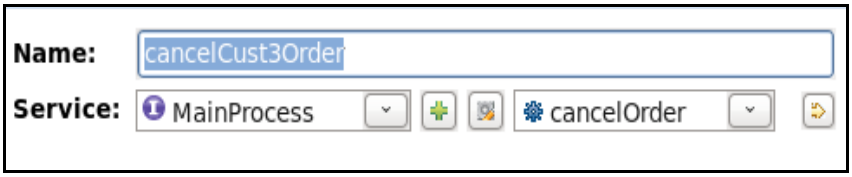
11. Add Invoke cancelOrder.

- Open *ACMETester* > *Processes* > ***WebTester.bwp***
  - Locate the Reference chevron in the upper right and **verify** *cancelOrder* is listed
  - Select and right click **cancelOrder** and select *Operation* > ***Invoke Operation***
  - Define a **transition** from the first group containing submitOrder to the **cancelOrder** activity

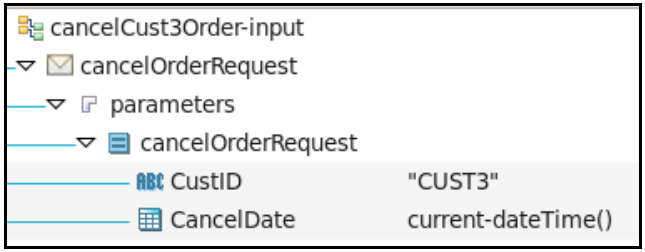
Tip: Click in the green area (not the plus sign) and start the transition from there.

12. Configure the Invoke cancelOrder activity.

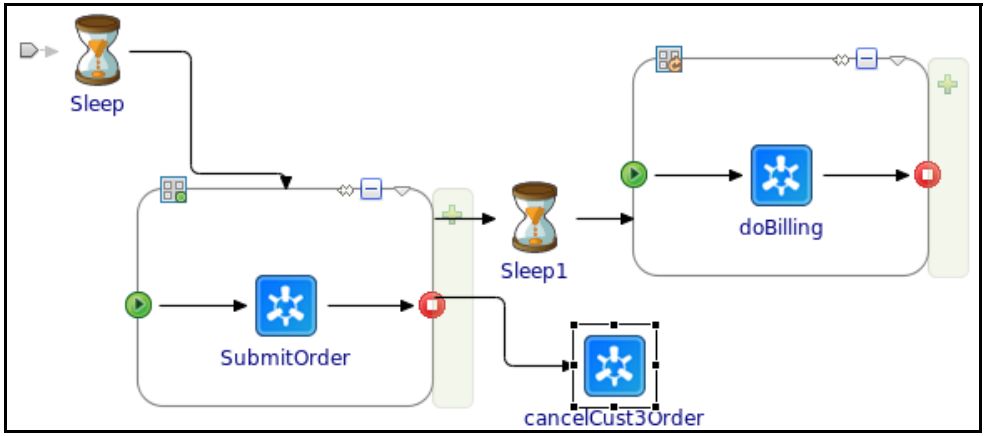
- Select the Invoke CancelOrder activity
  - *General Tab:* Name: **`cancelCust3Order`**



  - *Input Tab:*

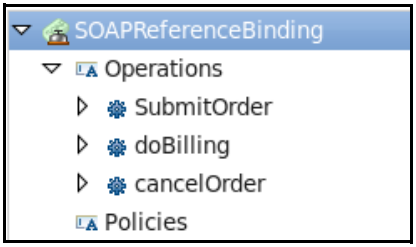


- **Save** changes
  - **Verify** process diagram appears as follows:

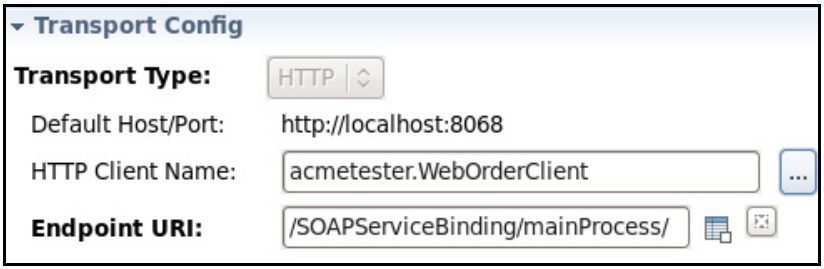


13. Reset ACMETester Components.

- Open *ACMETester> Module Descriptors* > ***Components***
  - **Delete** *ComponentWebTester*
  - **Re-add** *acmetester.WebTester*
  - Expand and select *ComponentWebTester* > *MainProcess* > *Properties* > ***Binding tab***
  - Click **Add Binding** button
  - Select **SOAP Binding**
  - Click **Finish**
  - Expand *SOAPReferenceBinding* > *Operations* and **verify** *cancelOrder* appears



- Expand *ComponentWebTester* > *MainProcess* and select the new **SOAPReferenceBinding**
- In Properties, scroll to *Transport Configuration*
  - Set Transport Type: **HTTP**
  - Client Name: browse and select **acmetester.WebOrderClient**



  - **Save** changes

14. Reset ACMEWebOrder Components.

- Open *ACMEWebOrder* > *Module Descriptors* > ***Components***
  - **Delete** *ComponentMainProcess*
  - **Re-add** *weborder.ComponentMainProcess*
  - Select *ComponentMainProcess* > *MainProcess* > *Properties* > ***Binding tab***
  - Click **Add Binding** button
  - Select **SOAP Binding**
  - Click **Finish**
  - Scroll down to *Transport Configuration*
  - Transport Type: **HTTP**
  - HTTP Connector name: **`weborder.HTTPConn8068`**
  - Click **OK**
- **Save** Projects

15. Launch the tester in run mode.

- Launch the Tester using the *Conversation* configuration
  - **Confirm** that both ACMETester and ACMEWebOrder applications are started
  - **Observe** messages in the console

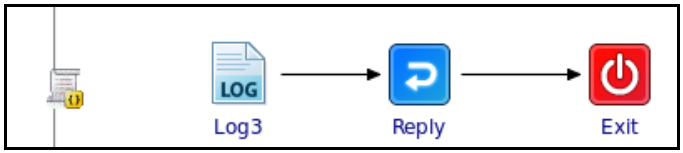
```
: Started BW Application [ACMEWebOrder.app:1.0]
Log -  *** Submit Order for CustID: CUST1***
Log -  *** Submit Order for CustID: CUST2***
Log -  *** Submit Order for CustID: CUST3***
Log -  *** Submit Order for CustID: CUST4***
Log -  *** Submit Order for CustID: CUST5***
Log3 - Order CANCELLED for CustID: CUST3
Log1 - Billing Info for CustID: CUST5
Log2 - Completed Order for CustID: CUST5
Log1 - Billing Info for CustID: CUST4
Log2 - Completed Order for CustID: CUST4
Log1 - Billing Info for CustID: CUST3
Log2 - Completed Order for CustID: CUST3
Log1 - Billing Info for CustID: CUST2
Log2 - Completed Order for CustID: CUST2
Log1 - Billing Info for CustID: CUST1
Log2 - Completed Order for CustID: CUST1
```

Analysis: While waiting for Billing to complete, Order3 is cancelled.
However, Billing info is still received for this Order!
This is because the process instance is not terminated.

- **Stop** the Tester

16. Solve the issue with unterminated order.

- Open *ACMEWebOrder* > *Processes* > *weborder* > ***MainProcess.bwp***
- In cancelOrder event handler, add an **Exit** activity after the *Reply* activity



- **Save** changes

17. Restart Tester and observe behavior.

- Restart the Tester with the *Conversation* configuration
- **Observe** the messages in the console output

```
Log -  *** Submit Order for CustID: CUST1***
Log -  *** Submit Order for CustID: CUST2***
Log -  *** Submit Order for CustID: CUST3***
Log -  *** Submit Order for CustID: CUST4***
Log -  *** Submit Order for CustID: CUST5***
Log3 - Order CANCELLED for CustID: CUST3
Log1 - Billing Info for CustID: CUST5
Log2 - Completed Order for CustID: CUST5
Log1 - Billing Info for CustID: CUST4
Log2 - Completed Order for CustID: CUST4




W-CORE-500050: The BW process [testers.WebTester] instance faulted,

NG-SOAP-550080: Unable receive reply from the service provider for
```

Analysis: The exit activity causes the process instances to terminate Cust3 is not billed. However, Cust1 and Cust 2 are also not billed. So, Exit is not the right way to solve this problem.

- **Stop** the tester when done
- Shut down Business Studio

18. Export your solution as a ZIP file and shut down Business Studio.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File: `/home/tibco/bw6edu/dev/exports/`**`EventHandlers.zip`**
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created
- **Exit** Business Studio

Notes

Notes

Unit 15 Using Compensation

This unit describes compensation and ways to trigger compensations.

Topics

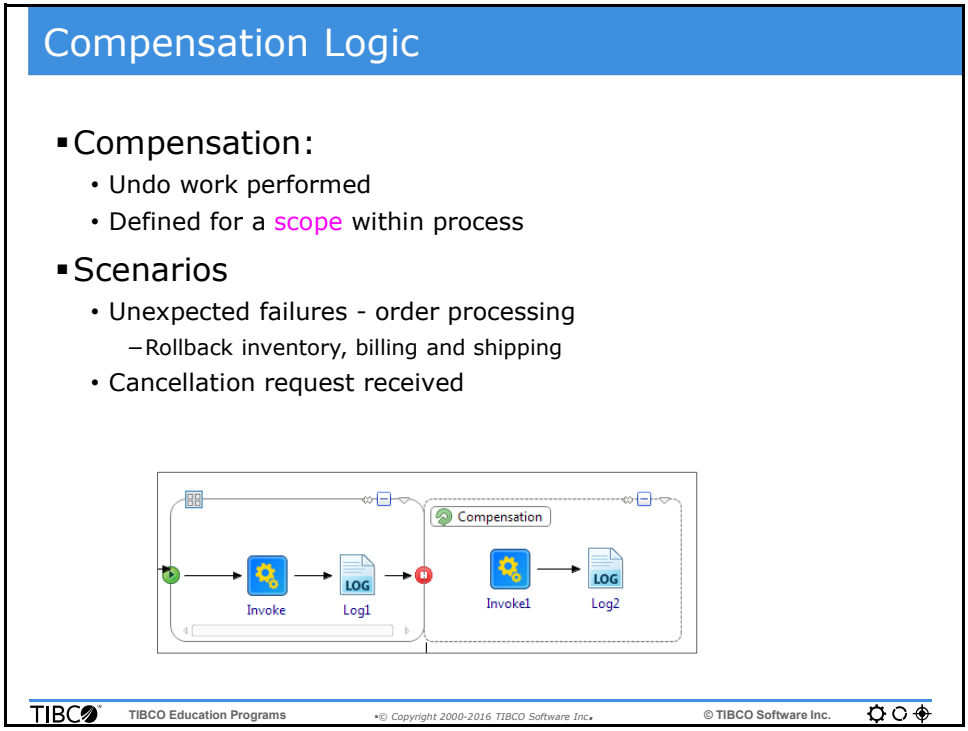
- *Objectives,* page 460
- *Compensation Logic,* page 461
- *Compensation Handler,* page 462
- *Explicitly Invoking Compensation Handler,* page 463
- *Compensation Handler: Implicit Invocation,* page 464
- *Rules for Compensation,* page 465
- *Compensation Logic Details,* page 466
- *Interface Specification for Lab O,* page 467
- *Implicit Compensation - Process-Pattern Mapping,* page 468
- *Explicit Compensation - Process-Pattern Mapping,* page 469
- *Review,* page 470
- *Lab Overview,* page 471

Objectives

- Knowledge
  - Explain purpose of compensations
  - Describe configuration of a compensation handler
  - Identify ways to trigger compensations
- Skills
  - Add compensating logic to process

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Objectives



Compensation Logic

A compensation logic maybe added to a process to undo the work done by previous activities in the process.

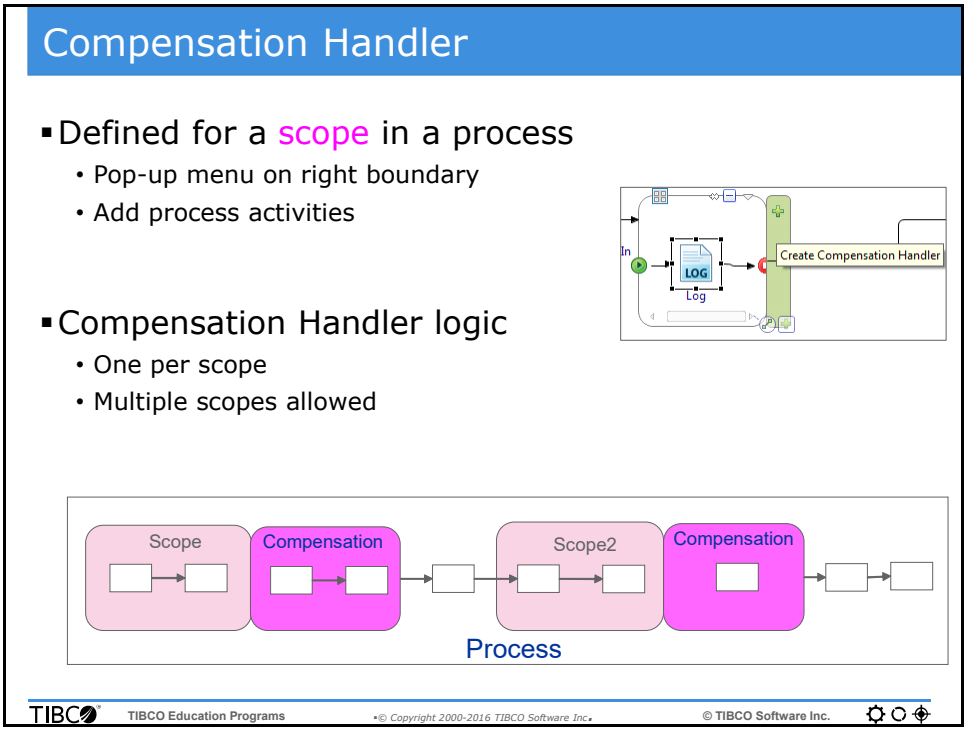
It does not perform a rollback of the previous activities, rather it reverse the actions performed by these activities.

A set of activities whose actions need to be reversed is enclosed within a scope. Next, compensation logic is defined for that scope.

Some scenarios where compensation logic may be needed are mentioned here:

Process error: When a business process fails due to unexpected errors, it may be important to undo the work done so that the process is terminated cleanly without any extra data. Thus, if there an error during order processing, it may be necessary to undo the actions for updating inventory, shipping and billing.

Business changes: If you receive a cancellation request during order processing, a similar kind of processing may be needed.

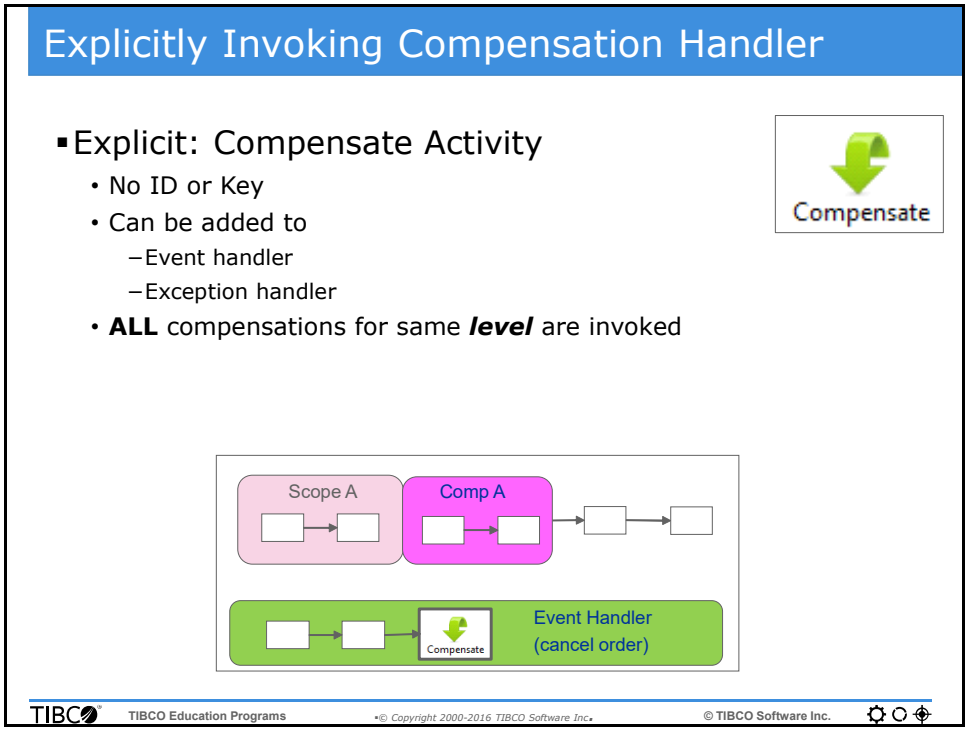


Compensation Handler

Scopes are used to define logical units of work. The definition of a scope is decided by the developer. The compensation logic for that scope is written in the compensation handler for the scope.

Compensation handlers are defined per scope. The compensation handlers of scopes that are directly enclosed by the scope and contain Compensate activity, can be executed. The handlers execute in reverse order, so the handler of the last completed scope is executed first.

A compensation handler is defined at a scope level and it is not available for the process itself.



Explicitly Invoking Compensation Handler

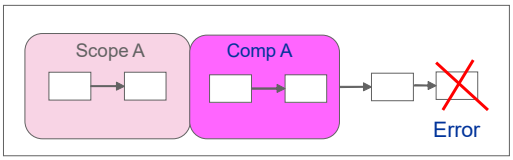
Compensation handlers can be explicitly invoked by using the compensation activity.

The compensation activity can be placed in an event handler or an exception handler. The activity itself has not ID or key settings.

This activity invokes all compensations for that level in the process. Scopes in the same process level are known as peer-level scopes.

Compensation Handler: Implicit Invocation

- Implicit invocation
  - Invoked on exception in business process
    - Exception must occur *outside* scope
  - Compensation(s) auto invoked
    - All compensations before **error**
    - Multiple compensations may be invoked
- Designing compensations
  - Database rollback, send cancellation messages



TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Compensation Handler: Implicit Invocation

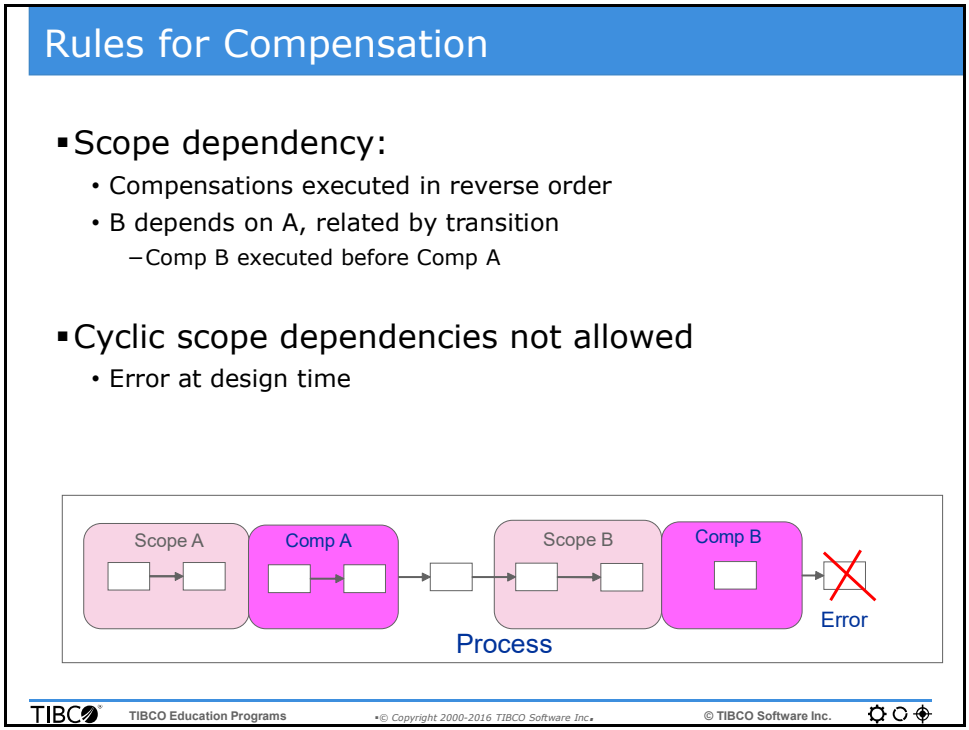
Compensations can also be implicitly invoked, upon an exception during process execution.

If an exception occurs outside a scope, the compensation handler for that scope is invoked automatically. No special configuration is required.

If there are multiple scopes with multiple compensations before the exception, all the compensations are invoked. Compensation handlers after the error cannot be invoked, since the process execution changes at that point.

When designing a compensation handler, it is therefore important that you decide the proper scope and also the appropriate compensation logic for that scope.

Defining undo logic for logical units of work requires some planning.

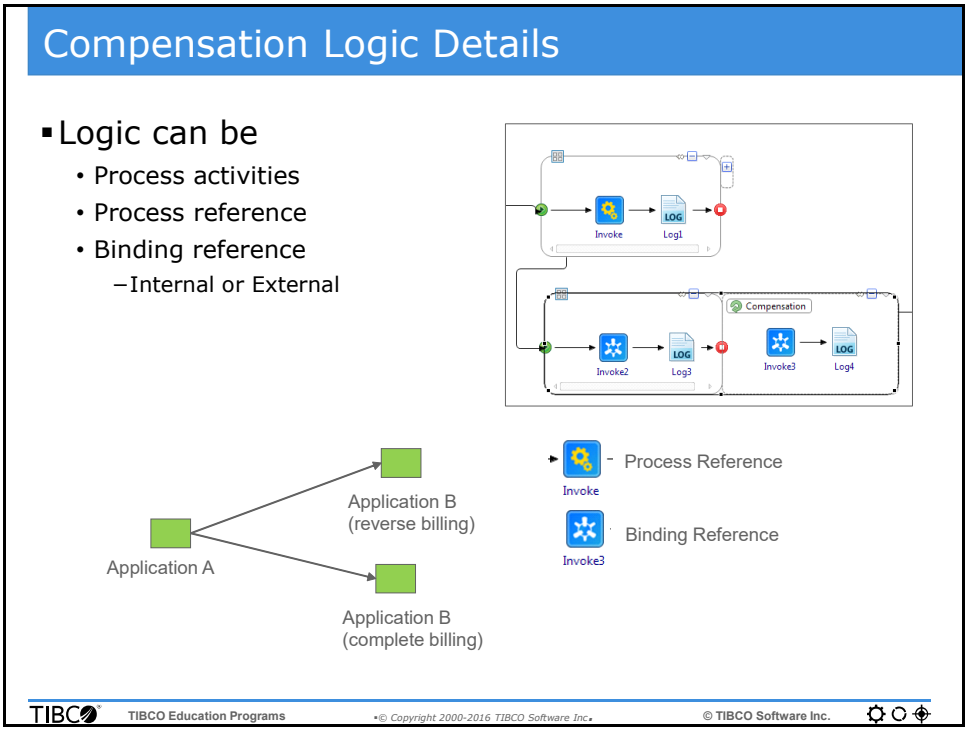


Rules for Compensation

When compensation handlers are executed, certain rules are enforced.

Scope dependency: When two scopes are related by transitions, scope dependency is established. When compensation logic is executed, it is done in reverse order of the execution of scopes. In the example shown above, the compensation logic for scope B is executed first, and the compensation for scope A is executed next.

Scope B executes after scope A. Scope A cannot be called after scope B, since this would establish a cyclic dependency. This is not allowed, and a design time error is thrown.



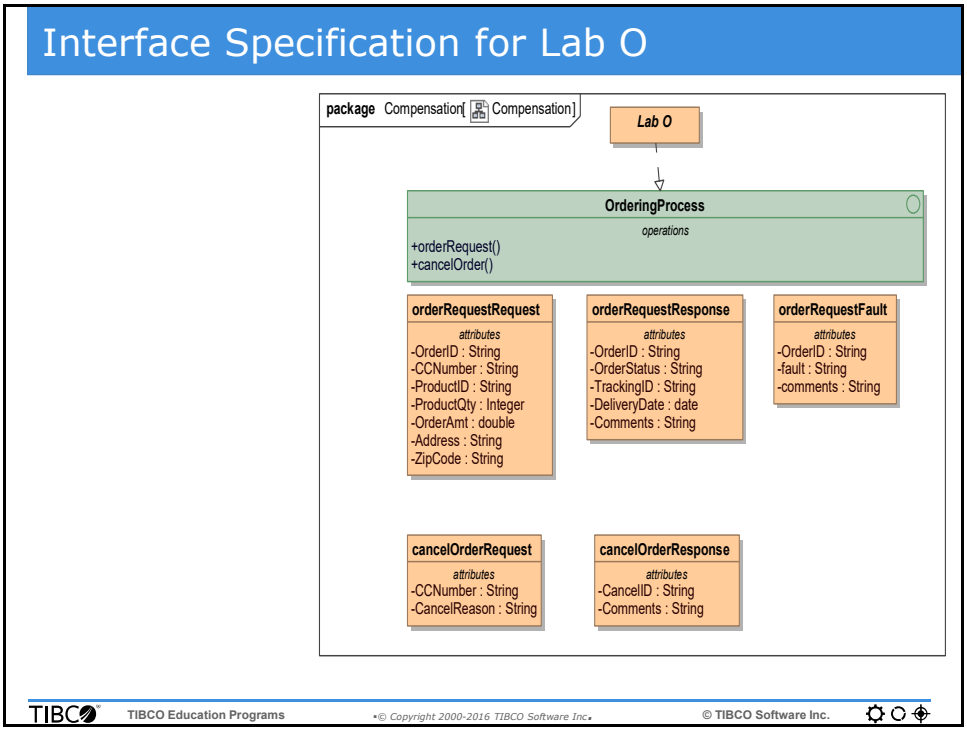
Compensation Logic Details

The compensation handler can be a simple set of process activities, within the handler itself.

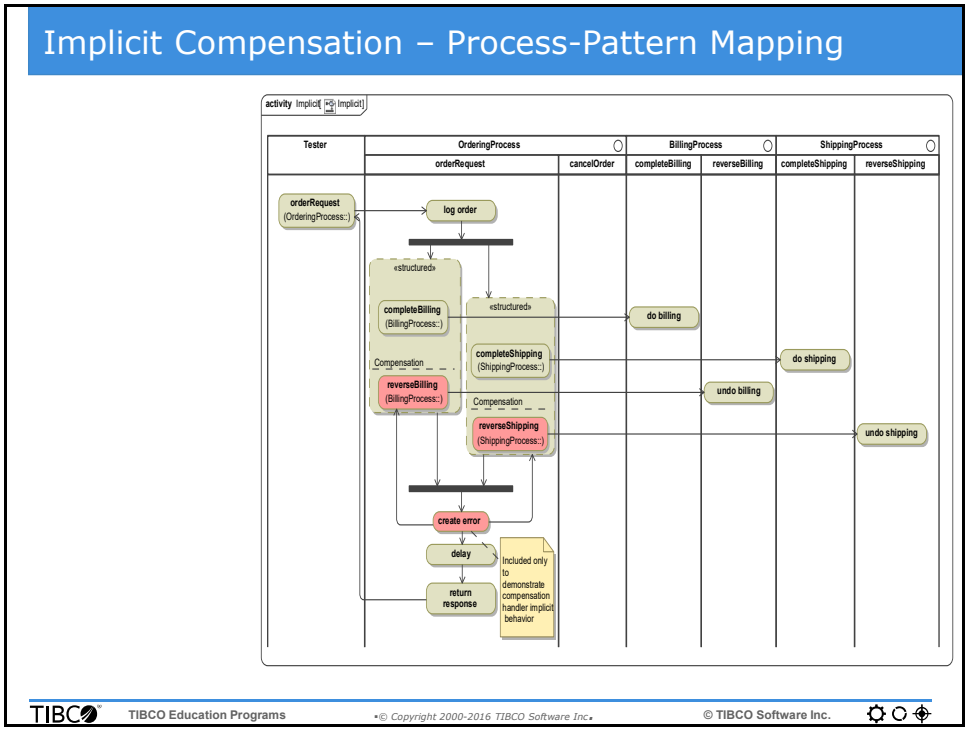
It can also invoke a sub-process within the application module (or one of the referenced shared modules).

It can also invoke a service operation from another module or en external service operation altogether, as shown.

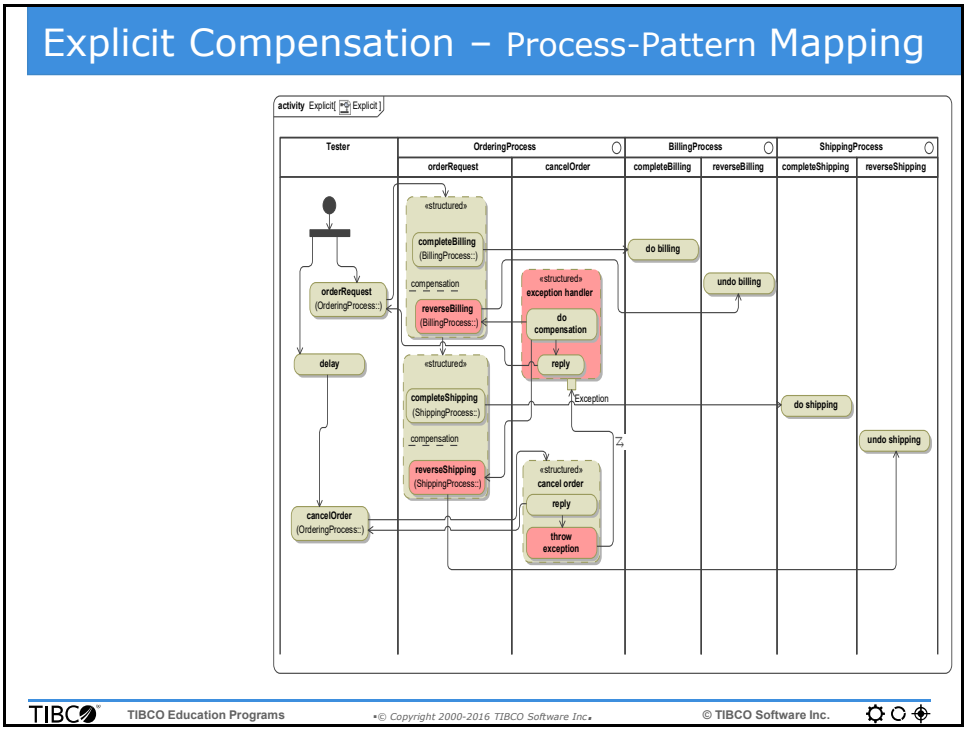
In short, the undo logic in a compensation handler can be as complex as you want.



Interface Specification for Lab O



Implicit Compensation - Process-Pattern Mapping



Explicit Compensation - Process-Pattern Mapping

Review

- Compensation: Undo logic
- Compensation handler: Associated with scope
- Invoking handlers: Implicit and Explicit
- Handlers executed in reverse order of scope

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Review

Lab Overview

- Implement compensation handler
  - Implicitly invoke by adding exception
  - Test
- Explicitly invoking compensation
  - Add compensation activity
  - Test

TIBCO TIBCO Education Programs •© Copyright 2000-2016 TIBCO Software Inc.• © TIBCO Software Inc.

Lab Overview

Notes

Lab O Compensating Transactions

Overview

In this lab you will import a different version of order processing service and add compensation handlers to it.

Objectives

- *Implicit Compensations,* page 474
- *Explicit Compensations,* page 481

Prerequisites

- The following TIBCO Software is installed and configured on your student machine:
  - BusinessWorks 6.3.1
- Student lab files extracted
- Completion of previous labs

Directions

Complete the exercises that follow.

Lab O - Exercise 1: Implicit Compensations

Overview

In this exercise add compensation handlers to the order processing service.

Steps

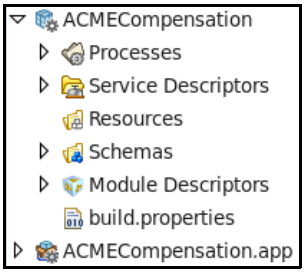
1. Launch TIBCO Business Studio and open your workspace.

- Launch TIBCO Business Studio and open your workspace at **`/home/tibco/bw6edu/workspace`**
- **Close** all open projects by right-clicking each module and application and selecting **Close Project**

Tip: Some files and schemas may be reused, so this will cause duplicate errors in the workspace. To avoid this, close all open projects.

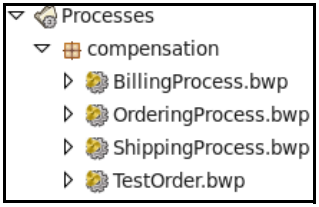
2. Import ConversationStarter application from ZIP file.

- Choose *File* > ***Import***
  - Select *General* > ***Existing Studio Projects into Workspace***, then click **Next**
  - Choose **Select Archive File**
- Browse and select `/home/tibco/bw6edu/dev/Others/ACMECompensationV2.zip`
- Confirm that both **ACMECompensation module** and **ACMECompensation application** are imported

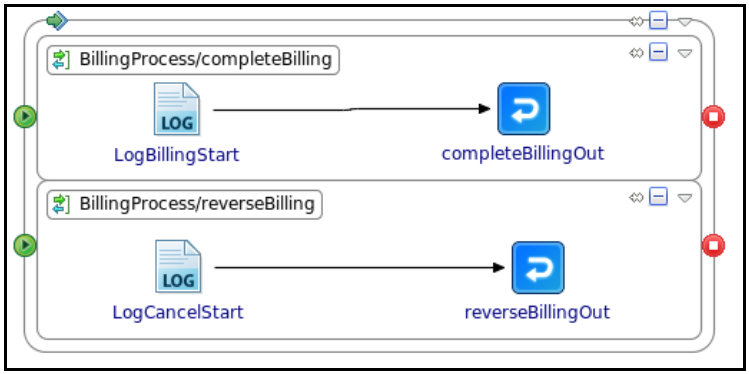


3. Examine imported application details and implemented services and processes.

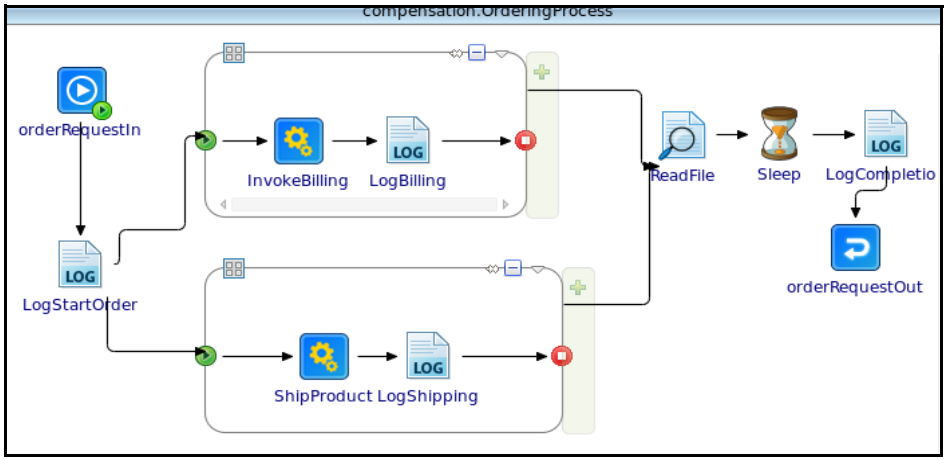
- **Verify** the following Processes exist in the *compensation package* as shown:



- Open **BillingProcess**
  - **Confirm** that both *completeBilling* and *reverseBilling* are implemented



- Similarly, open and examine **Shipping Process** and its two operations
- Open **OrderingProcess** and confirm that it calls the *Billing* and *Shipping* processes



4. Run the application and observe initial runtime behavior.

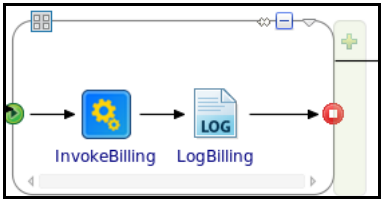
- In File Browser, navigate to */tmp* folder and create a file named **test**
- Define a new *Run Configuration* to named **`Compensation`** which will run the Compensation app only
- Click **Debug**
- Verify the application returns the following:

```
-300006: Started BW Application [ACMECompensation.app:1.0]
rtOrder - *** Submitted Order for CCNumberA1234
rt -  Shipping OrderID:AAA
lingStart - Billing for CC: A1234 Amount : 1212
ling - Billing Complete
pping - Shipping Complete
mpensation.Log - *** Completed Order for CCNumber: A1234
```

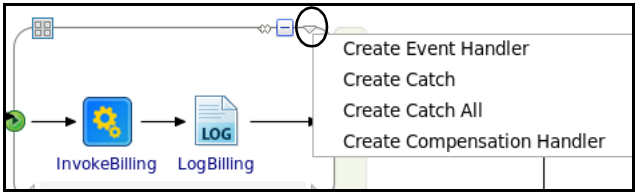
- **Stop** the tester, and switch back to Design perspective

5. Add a compensation handler for the call to Billing service IN OrderingProcess.

- Add a compensation handler using one of the following two options
  - Option 1: **Hover** on right side boundary of the group containing *InvokeBilling*
    - Click on '+' sign to add a compensation handler

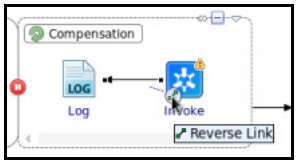


  - Option 2: Click on the *down arrow* on the group window and see a menu and select **Create Compensation Handler**



6. Add compensation logic to reverse the billing entries.

- In the Compensation Handler area, add a **Log activity** and an **Invoke activity**
- Create **transition** from *Log* to *Invoke* activity
- Select to highlight the transition, then click the **reverse link icon**
- **Verify** the transition now goes from *Invoke* to Log *activity*



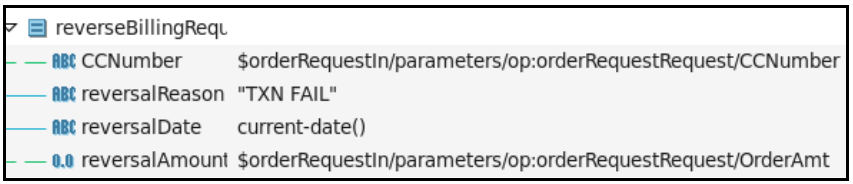
Tip: We are defining reverse flow to examine runtime behavior of such transitions .

- Configure log activity,
  - Name: **`LogCancelBilling`**
  - Input tab > Message: **`"***** Billing REVERSED *****"`**

7. Update the invoke activity to include CC Number.

- *General Tab* > *Name*: **`reverseBilling`**

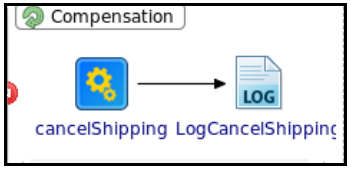
- Input tab:



- **Save** changes

8. Define compensation handler for call to Shipping service.

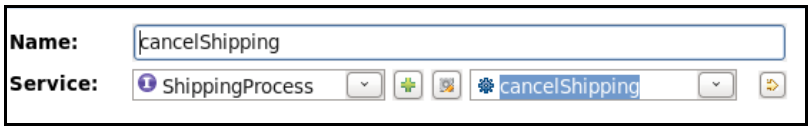
- Add a Compensation Handler and Log and Invoke activities to the group containing ShipProduct activity as shown



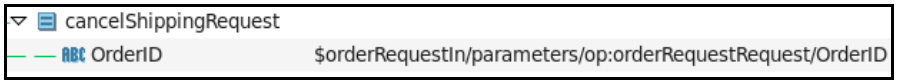
- Create **transition** from Invoke to Log activity
- Configure Log activity
  - Name: **`LogCancelShipping`**
  - Input > Message: **`"***** Shipping CANCELLED ***** "`**

9. Configure Invoke activity.

- *General* Tab: Name: **`CancelShipping`**

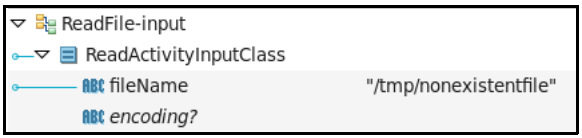


- Input:



10. Update read file to read a nonexistent file, to cause a runtime error .

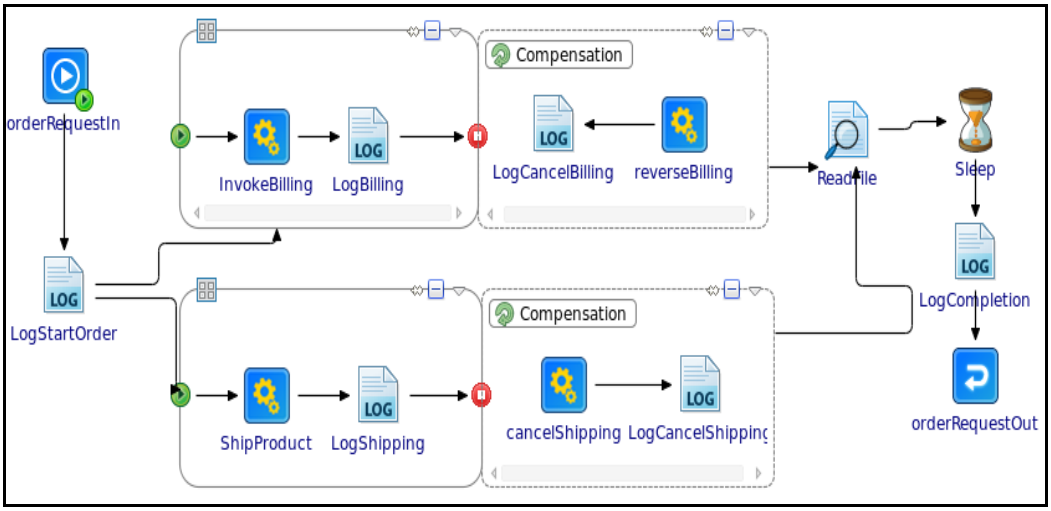
- Select **Read File** activity
- `Properties > Input > fileName -->"/tmp/nonexistentfile"`



- **Save** your changes and **verify** there are no errors in your solution

11. Verify the final process definition.

- Rearrange the activities to see the process diagram clearly as shown



Testing

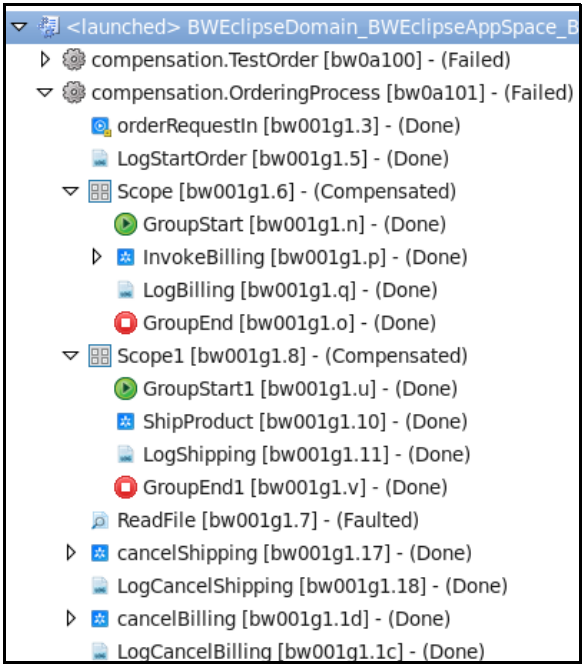
12. Launch the tester in debug mode.

- Right-click in the OrderingProcess and select **Debug as BusinessWorks application**
- **Observe** the console output, and scroll up the log to see the messages
  - **Verify** the process fails due to a non-existent file in the Read File activity
  - **Verify** compensation logic was executed successfully for both shipping and billing

```
Started BW Application [Compensation.application:1.0]
- **** Submitted Order for CCNumber :  A1234
ogStart -  Shipping OrderID:AAA
t - Billing for CC: A1234 Amount : 1212
Billing - Billing Complete
Shipping - Shipping Complete
ogCancel -  Cancel Shipping - OrderIDAAA
ing -  **** Shipping CANCELLED *****
 - Cancel Billing for A1234 Amount: 1212
ng - **** Billing REVERSED ****
```

13. Examine the process execution logic.

- **Examine** the process execution of BusinessWorks Jobs and **observe** order of execution



Analysis: These are the significant points. Billing and Shipping are executed sequentially, despite being parallel transitions. Read file fails and the compensation logic is executed after it.

Reversed Transition - *reverseBilling* is executed before *logCancelBilling*

- **Stop** tester when done

Lab O - Exercise 2: Explicit Compensations

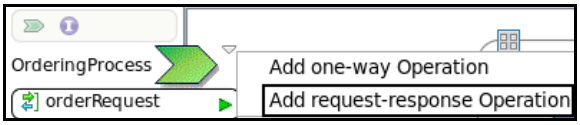
Overview

In this exercise use the exception handler to trigger the compensation for Billing and Shipping.

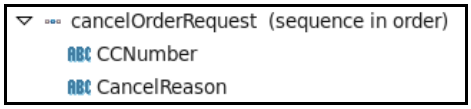
Steps

1. Add a new operation, cancelOrder in MainProcess, and set its input and output.

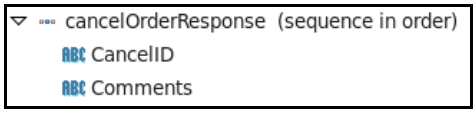
- In OrderingProcess, locate the *Service* chevron and click the **down arrow** (top right corner)
  - Choose **Add Request-Response Operation**



- **Verify** new operation named *operation* is added
- Select **operation**, then *Properties* >
  - *General* tab
    - Click the **Light Bulb,** rename to **`cancelOrder`**
  - *Input* Tab
    - Change the name (in) to **`CCNumber`**
    - Add new element named **`CancelReason`** of type string



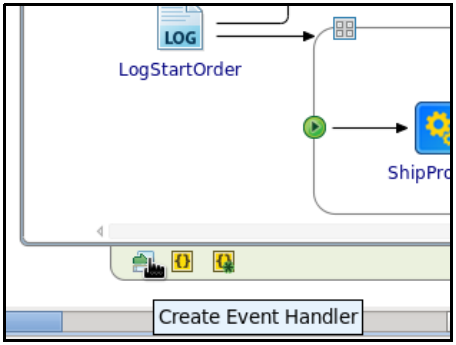
  - *Output* tab:
    - Add two new elements named **`CancelID`** and **`Comments`** as type string



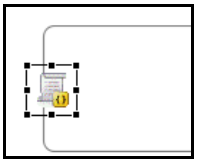
- **Save** changes

2. Implement the cancelOrder operation by adding an event handler.

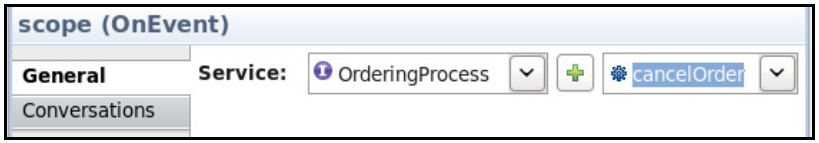
- **Hover** on *lower boundary of OrderingPocess* to make a floating menu appear



  - Click **Create Event Handler** icon
- Click on **Event Handler** icon

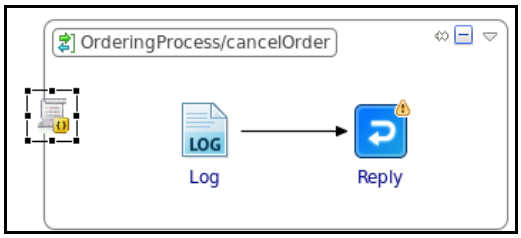


- Configure the scope as shown.



3. Add activities in the event handler, and set the log message.

- Add **Log Activity** and a **Reply activity**
- Create a **transition** from Log to Reply

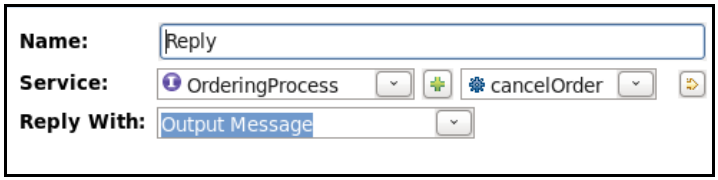


- Log activity
  - Name: **`LogCancel`**
  - Input shown below:

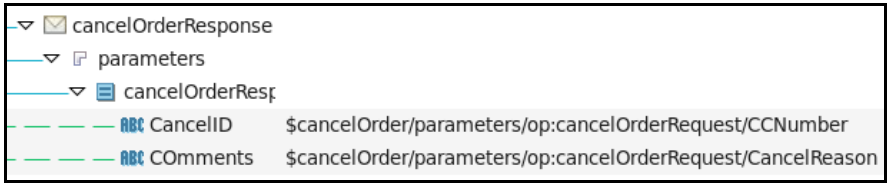
ABC *logLevel?*	
ABC message	concat(" Cancellation Request for CCNumber :", $cancelOrder/parameters/op:cancelOrderRequest/CCNumber)

4. Configure the Reply activity.

- Configure the **Reply** activity
  - *General* Tab:

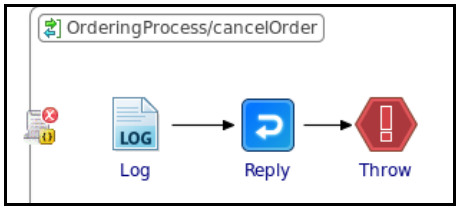


  - Input tab:

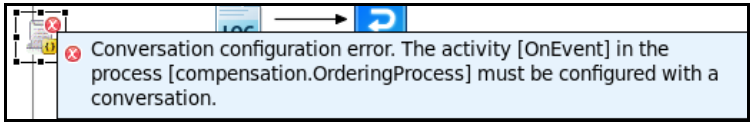


5. Add a **throw** activity after reply connect it.

- **Verify** the completed event handler is as shown

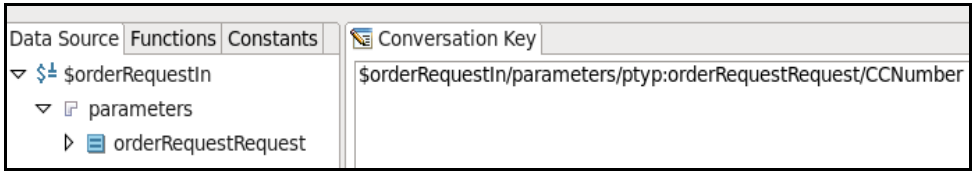


- **Save** changes
- **Verify** there is one problem with the event handler
- **Hover** over the error icon to see details - there needs to be a conversation



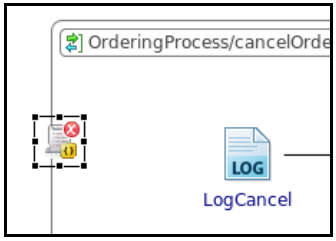
6. Update the Order receive activity to start a conversation.

- Select the **orderRequestIn** activity (Process starter activity)
  - In *Properties* > ***Conversations***, add a new conversation
  - Configure Conversation key to be **CCNumber** as shown:

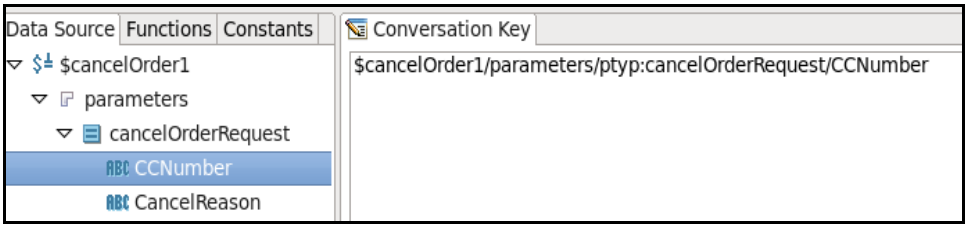


7. Update the event handler to join the conversation.

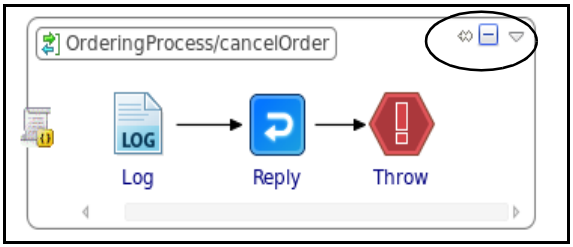
- Select the **event handler icon** for cancelOrder (on the left side of the event handler area)



- In *Properties* > ***Conversations***, click **Join**
- Map *CCNumber* to the *Conversation Key area* as shown

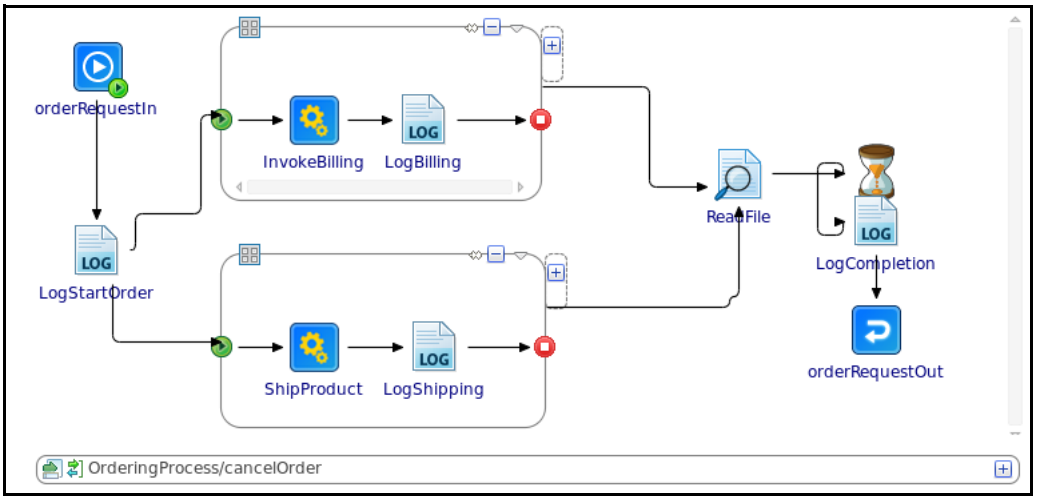


- **Save** changes and verify that there are no errors
- Collapse the event handler area by clicking the **dash symbol icon** in the upper right



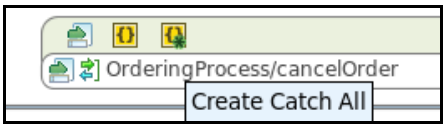
8. Collapse the compensation handlers to make the process readable.

- **Collapse** the compensation handlers
- **Rearrange** activities to make the process readable



9. Add an exception handler to the process.

- **Hover** over the cancelOrder event handler and choose to **Create Catch All** exception handler



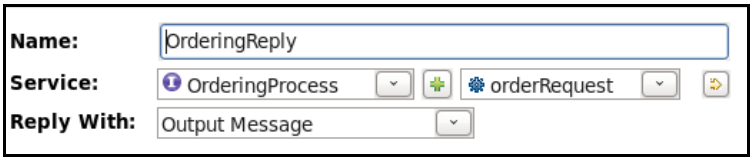
- Add a Log activity name it **`LogException`**
  - Input: Message: **`"In Exception Handler"`**

10. Add a compensate activity to the exception handler.

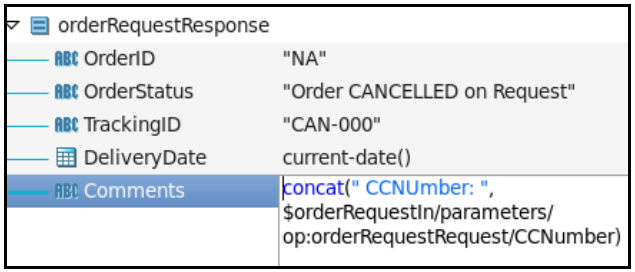
- Add a **Compensate** activity, and connect it to the **log**
- There is no configuration for compensate activity

11. Add a reply activity to respond to the OrderProcessing request and configure it .

- *General* tab > Name: **`OrderingReply`**



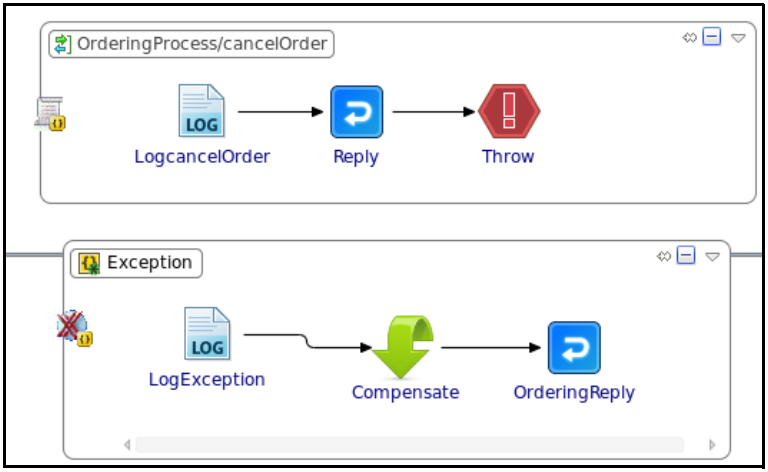
- Input**:**



- **Save** changes

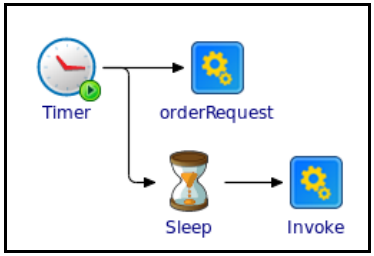
12. Verify completed exception handler and event handlers.

- Verify exception process appears as shown below:



13. Update the tester process to invoke cancel order operation.

- Open **TestOrder** process
- Add an **Invoke** activity and a **Sleep** activity as shown:



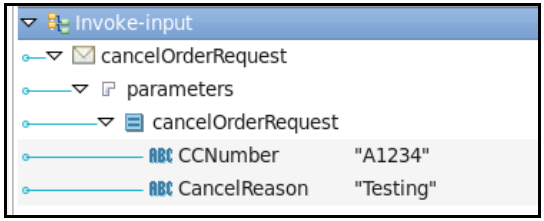
- Configure Sleep activity for `5000` milliseconds

Tip: We need a sleep activity, so that the OrderProcessing can be started before the cancel event is sent. If not, the cancel event may trigger early, and it will be lost.

14. Configure the Invoke Activity.

- General > Name: **`CancelOrder`**

- Input:



Testing

15. Fix the read file error, so that the process will not cause an exception.

- In OrderingProcess, open the **Read file** activity
  - *Properties* > *Input* > ***fileName***: **`"/tmp/test"`**

Note: Recall from an earlier step, that this file exists .

- **Save** changes

16. Launch the tester in debug mode and confirm that explicit compensation works as designed.

- Right-click the process and select **Debug BusinessWorks application**
- **Observe** the detailed messages in console log

```
HOR-FRWK-300006: Started BW Application [ACMECompensation.app:1.0]
A.LogStartOrder - *** Submitted Order for CCNumberA1234
A.LogStart -  Shipping OrderID:AAA
A.LogBillingStart - Billing for CC: A1234 Amount : 1212
A.LogBilling - Billing Complete
A.LogShipping - Shipping Complete
A.LogCancel -  Cancel Request for CCNumber:A1234
A.LogException - In Exception Handler
A.LogCancel -  Cancel Shipping - OrderIDAAA
A.LogCancelShipping - ***** Shipping CANCELLED ****
A.LogCancelStart - Cancel Billing for A1234 Amount: 1212
A.LogCancelBilling - ***** Billing REVERSED ****
```

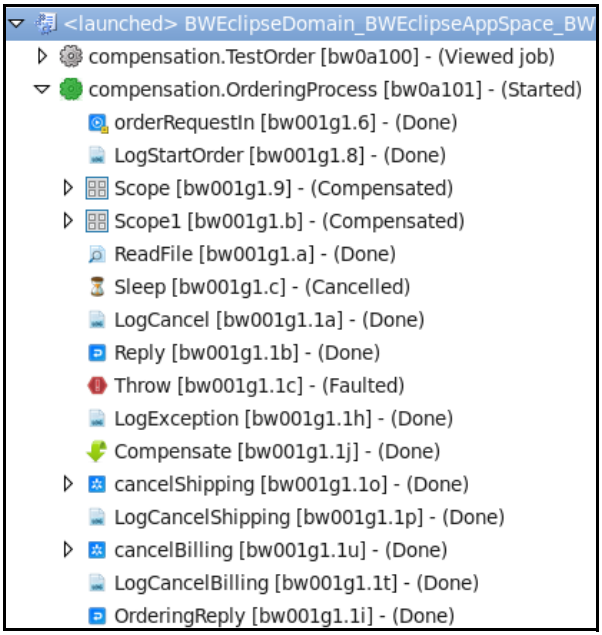
Analysis: The execution flow is as follows:

- Shipping and Billing are completed.
- While the process is waiting , a cancelOrder requests is received.
- The event handler throws an exception.
- The exception handler explicitly invokes a compensation task.
- Both Billing and Shipping compensation logic is executed.

This demonstrates explicit compensation logic.

17. Examine detailed process execution logic.

- Double-click the **Debug tab** to maximize the window
- **Expand** the Application details to view process flow:



- When you are done, **stop** the tester
- **Close** Business Studio

18. Export your solution as a ZIP file and shut down Business Studio.

- Choose *File* > *Export* then *General* > ***Studio Projects to Archive***
  - In the next window, choose **Select All Projects**
    - To Archive File:
      `/home/tibco/bw6edu/dev/exports/`**`Compensations.zip`**
    - Save in ZIP format
- Click **Finish**
- Verify ZIP file has been created
- **Exit** Business Studio

Notes

Unit 16 Enterprise Deployment

This unit discusses enterprise mode deployment details. It also examines the usage of TIBCO Enterprise Administrator.

Topics

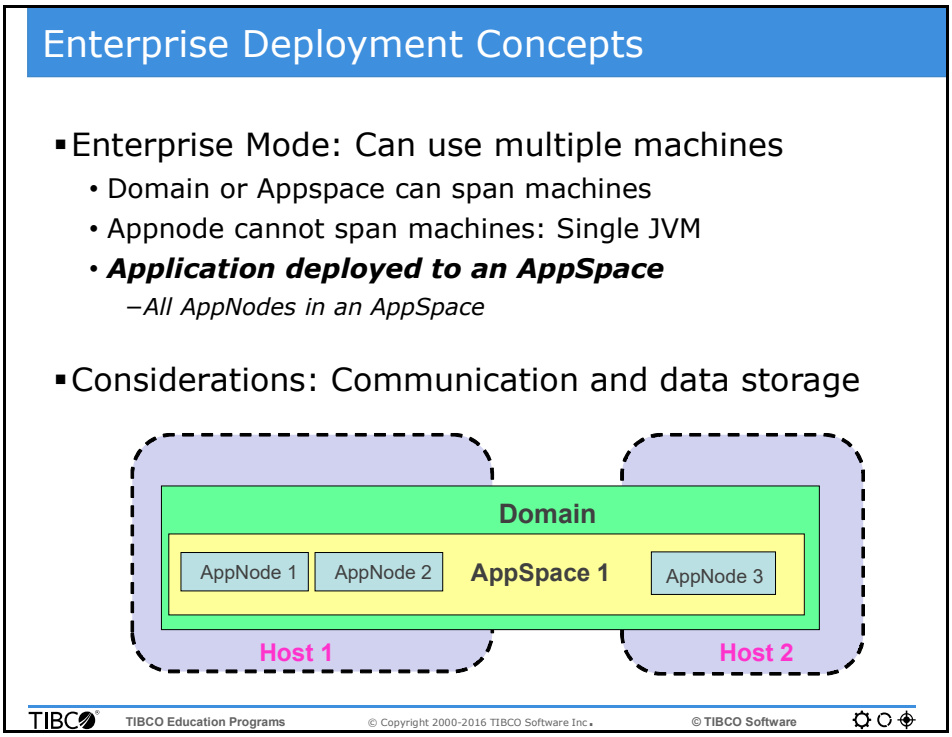
- *Objectives,* page 494
- *Enterprise Deployment Concepts,* page 495
- *Enterprise Mode: Communication and Data Storage,* page 496
- *TIBCO Enterprise Administrator,* page 497
- *Comparing Administrator and TEA,* page 498
- *Domain Communication and Runtime Components,* page 499
- *Setting up TEA for BusinessWorks,* page 500
- *TIBCO Enterprise Administrator Features,* page 501
- *BusinessWorks Administration Views,* page 502
- *Deployment Options,* page 503
- *Deployment Choice 1: bwadmin,* page 504
- *Deployment Choice 2: TIBCO Business Studio,* page 505
- *Deployment Choice 3: TIBCO Enterprise Administrator,* page 506
- *Review,* page 507
- *Lab Overview,* page 508

Objectives

- Differentiate between runtime environments
- Discuss enterprise mode deployment details
- Explain the role of bwagent
- Examine usage of TIBCO Enterprise Administrator
- Identify deployment options for archives

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Objectives



Enterprise Deployment Concepts

Enterprise Mode: This mode of application deployment is used when you want application fault-tolerance and high availability.

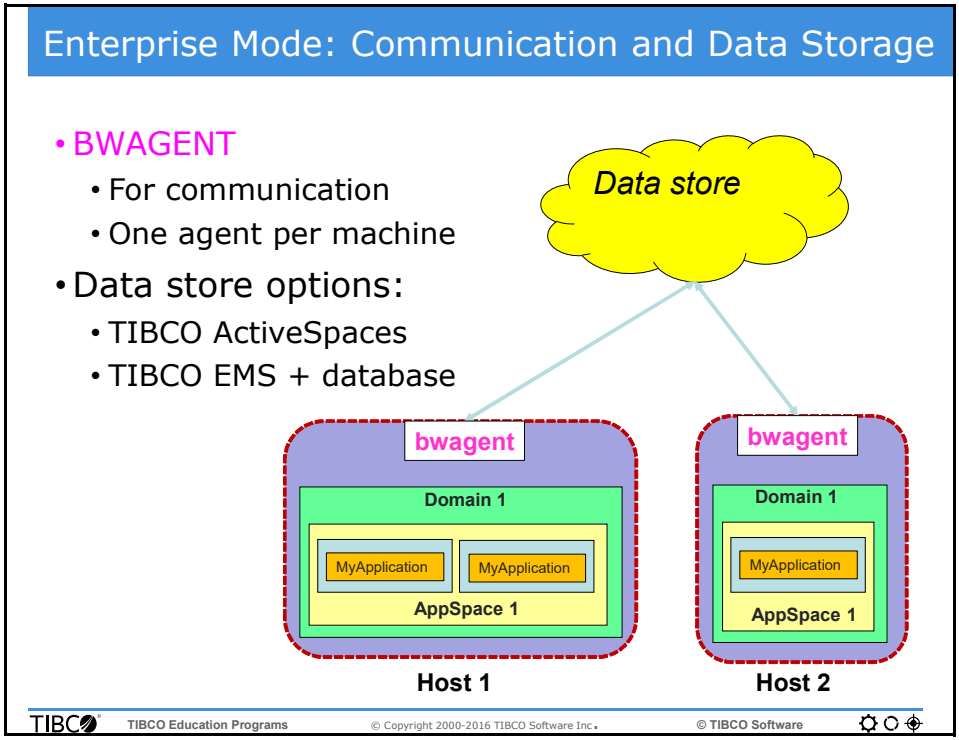
Setting up and using this mode requires additional runtime components, which we will examine now.

In Enterprise mode of deployment, an AppSpace can span machines. Each machine in turn can have one or more AppNodes that belong to the AppSpace. An AppNode is a JVM and a JVM cannot span machines.

Regardless of Mode: An Application is always deployed to an AppSpace.

The AppSpace is responsible for deploying the application to all AppNodes in the AppSpace.

When deploying in Enterprise mode, particular attention needs to be paid to the communication between machines and the storage of metadata (data about the Environment). These details are examined next.



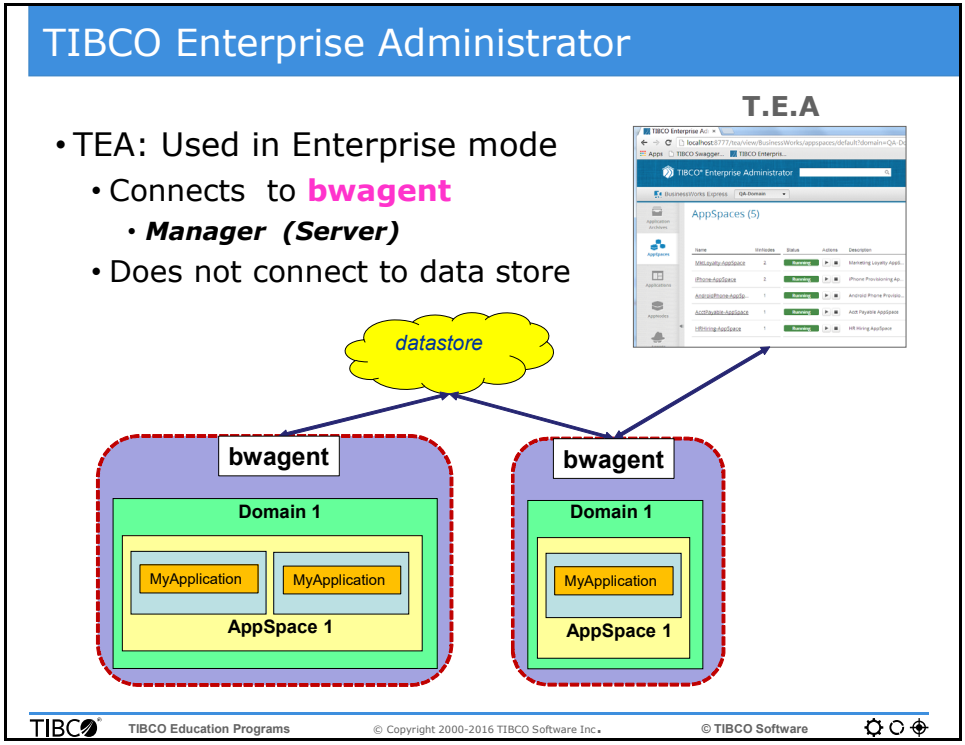
Enterprise Mode: Communication and Data Storage

bwagent: Daemon process responsible for provisioning AppNodes and applications, performing administration commands, and synchronizing data from the datastore with the local file system. There is one bwagent for each installation. Agents located on different machines communicate with each other. When multiple bwagents are configured to communicate with each other using a common datastore, they form a bwagent network.

Data Persistence options:

TIBCO ActiveSpaces for data persistence and communication.

External database for persistence and TIBCO Enterprise Messaging Service for communication.

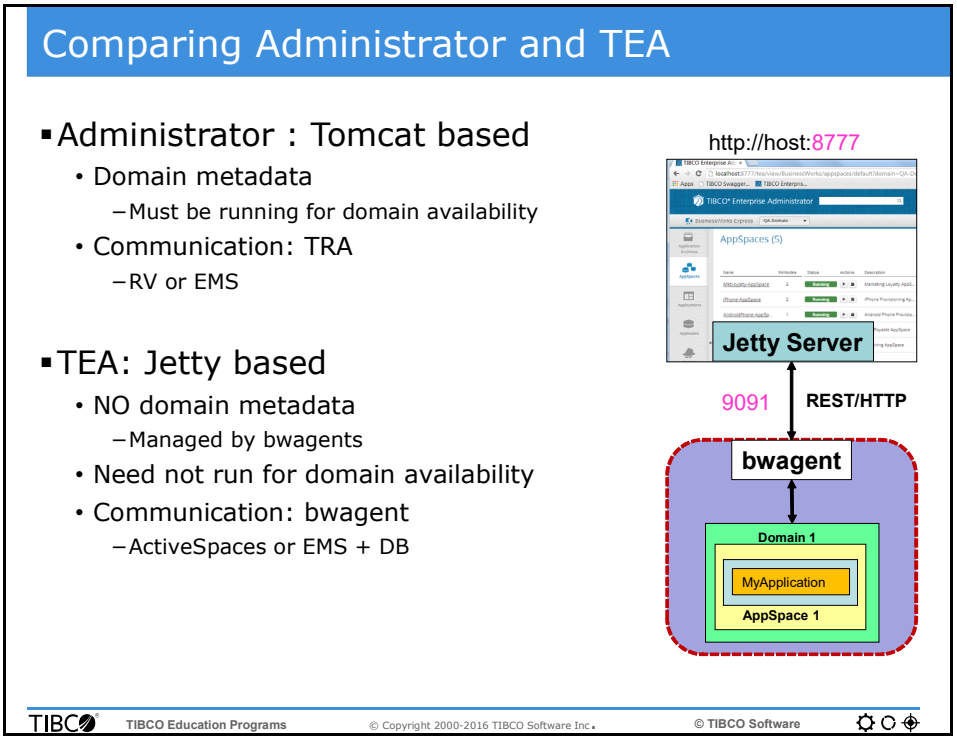


TIBCO Enterprise Administrator

TEA server is hosted within a web server provides a browser based UI. The server interacts with the agent to get data about the products registered on the TIBCO Enterprise Administrator. The server is responsible for:

- Collecting data on all the products registered with it
- Caching data for faster searches
- Hosting all the TIBCO Enterprise Administrator server views Responding to auto-registration requests from agents
- Providing details about the machines on which the products are running
- Providing user management features such as granting and revoking a user's permissions

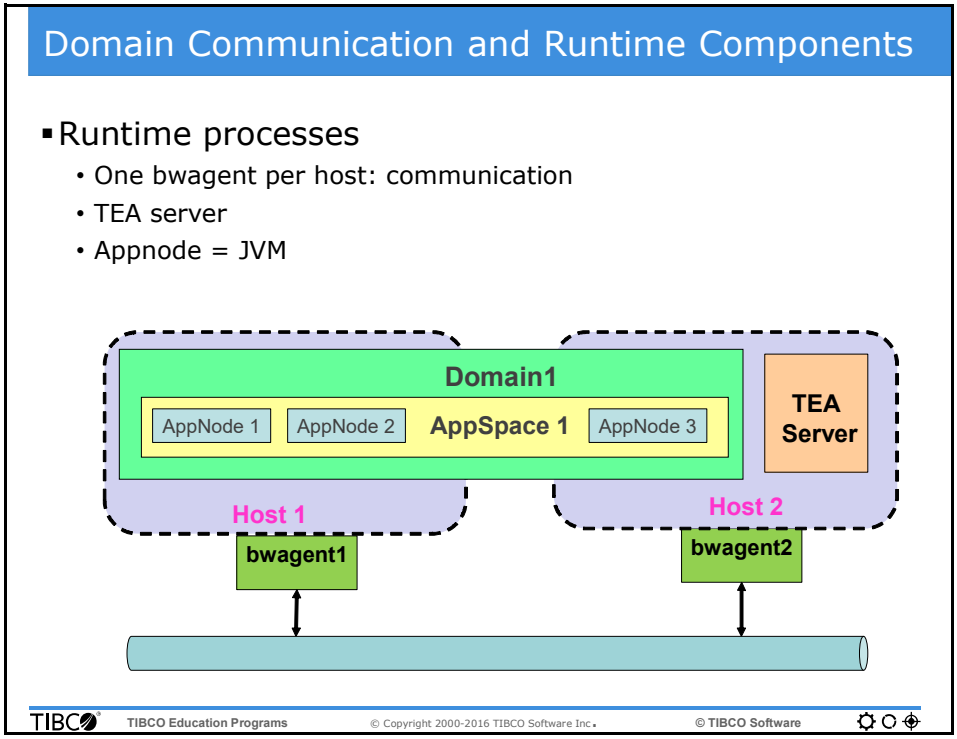
TEA does not directly connect to the datastore, and it relies on the bwagent to provide the information.



Comparing Administrator and TEA

TIBCO Administrator and TIBCO Enterprise Administrator vary greatly in capabilities and functionality. The following list summarizes the key differences from the domain perspective:

- TEA does NOT store domain metadata, therefore, its availability is not relevant to the availability of the domain.
- Domain and application metadata is manages by the network of bwagents.
- Unlike Administrator, there is no bwagent per domain, rather, there is a bwagent per machine, which is responsible for all enterprise domains on that machine.
- BWAgents use ActiveSpaces as the default communication mechanism, EMS + Database is also provided as an option when the number of bwagents is large.
- TEA communicates with bwagent using REST (HTTP/JSON). These requests are serviced by the bwagent and responses provided.



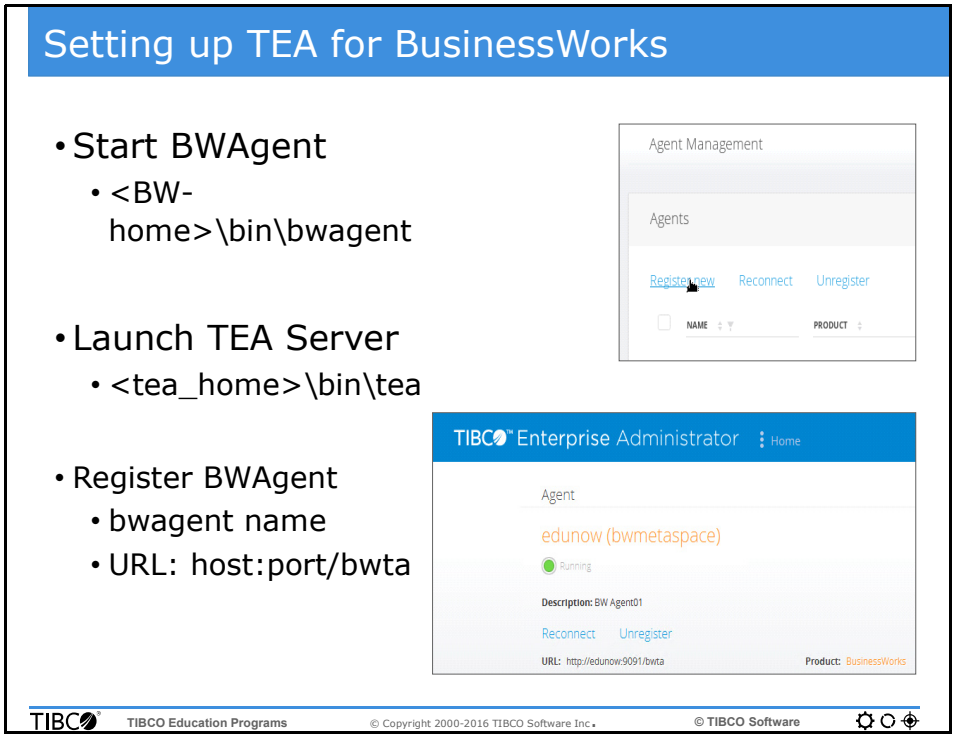
Domain Communication and Runtime Components

In version 6, the number of runtime components is reduced.

1. There is one bwagent per installation (there is typically one installation per host), regardless of number of domains.

2. There is one JVM per running AppNode. Domain and AppSpaces are logical constructs. All BusinessWorks application run in this one AppNode.

3. There is one TEA server which may be running on any one of the machines.



Setting up TEA for BusinessWorks

An agent is a bridge between the TIBCO Enterprise Administrator server and a product. When registered with TEA, it discovers the product that must be exposed to the administrator. The agent creates a graph of objects specific to the product that needs to be rendered on the TIBCO Enterprise Administrator server UI. The agent interacts with the server using the REST API.

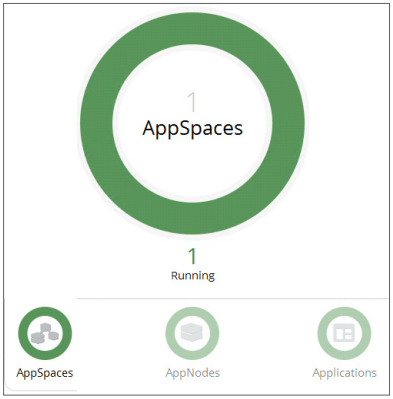
To enable the Admin UI, the bwagent must be registered with a running TEA server. If the registered bwagent becomes unavailable, the connection between the TEA server and the agent network is automatically recovered.

- Start TEA by running the TEA executable in <tea_home>\bin.
- Register a bwagent using the bwadmin command line tool, with the registerteaagent command.

You can also register the bwagent from the TEA interface. Provide the url http://<host>:9091/bwta for registering a bwagent.

TIBCO Enterprise Administrator Features

- Capabilities:
  - Centralized Administration
  - Shared Services model
    - Agent based
  - Web & command line interfaces
  - Management
    - Domains/ AppSpaces/AppNodes
  - Register agents
  - Machine monitoring



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

TIBCO Enterprise Administrator Features

TEA provides centralized administration for one or mode domains and all the objects within the domain such as AppSpaces, Appnodes and applications.

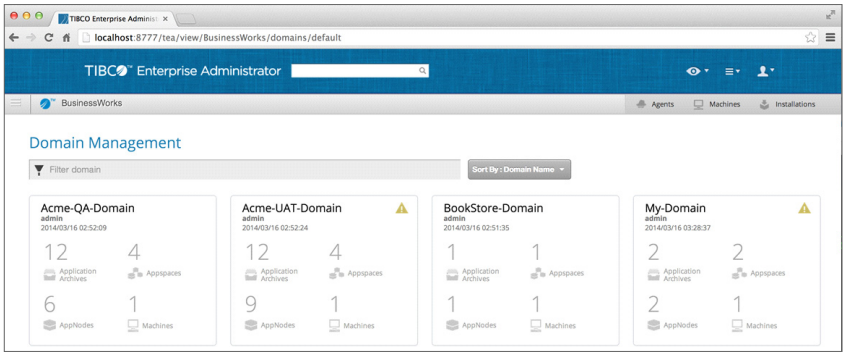
Register Agents: Products have a TEA agent which is registered with the TEA server, and administration is enabled. It thus provides single-point access to multiple products deployed across an enterprise. You can easily manage and monitor runtime artifacts.

Products with TEA Agents: EMS, Hawk, MDM and more Shared Services Model: Shares common administrative concepts across all products, promoting a consistent and reusable shared services model.

TEA Shell: A command line utility provided with TEA. It is a remote shell based on the SSH protocol that provides the command line interface for the full range of TEA operations. The scripting language is similar to that of bash from Unix.

BusinessWorks Administration Views

- Domain summaries
- Domain-independent views of
  - Agents, Machines and Installations



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

BusinessWorks Administration Views

The screen shot above shows an typical enterprise environment which has a few domains. Each domain has one or more AppSpaces and each AppSpace has AppNodes and deployed applications. The TEA interface shows a dashboard view of all running components. You can now drill down into an individual domain, all the way to application level, and examine details.

You can also start and stop AppSpaces, AppNodes and application. All these view are unique to a BusinessWorks administration environment. The bwagent instructs TEA on the proper display of the data, and also provides all the data.

Deployment Options

- Key Options
  - Command line (scripting with bwadmin)
  - TIBCO Business Studio
  - TIBCO Enterprise Administrator
- Option availability depends on mode
  - Local
    - Command line only
  - Enterprise
    - All options available

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Deployment Options

There are three deployment options for BusinessWorks applications.

Command Line: This is a combination of two utilities for building a EAR file and deploying it to a runtime environment.

TIBCO Business Studio: Provides a deployment view, where Applications can be deployed directly to an Enterprise environment.

TIBCO Enterprise Administrator (TEA): Provides the most features for application deployment and lifecycle management.

The Local mode supports command line only. Enterprise mode supports all three options.

Deployment Choice 1: bwadmin

- Use **bwadmin** utility script file
  - Commands:
    - Define Domain, AppSpace, AppNode
    - Upload and Deploy EAR
    - Start AppSpace
  - Execution: bwadmin **config -cf** c:/temp/MyScript.txt

```
echo "Create test-Domain"
create -descr "Test Applications Domain" domain test-Domain

echo "Create test-AppSpace and AppNode"
create -d test-Domain -minNodes 1 -descr "Test AppSpace" app
create -d test-Domain -a test-AppSpace -descr "test AppNode"

echo "Upload and Deploy Application MyApp.ear"
upload -d test-Domain C://temp/MyApp.ear
deploy -d test-Domain -a test-AppSpace -descr " Application"

echo "Start test-AppSpace"
start -d test-Domain appspace test-AppSpace
```

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Deployment Choice 1: bwadmin

The command line option involves the use of bwadmin utility to define the runtime environment and deploy EAR files to it.

In addition, the bwdesign utility is a command line tool that provides the ability to load BusinessWorks modules and generate EAR files.

To deploy a EAR file, use the bwadmin utility.

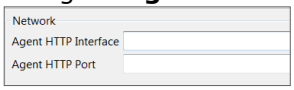
This utility provides the same command options whether using Local mode or Enterprise mode.

The entire runtime environment can be created using a series of commands. You can also package these commands into a script file and execute them at once.

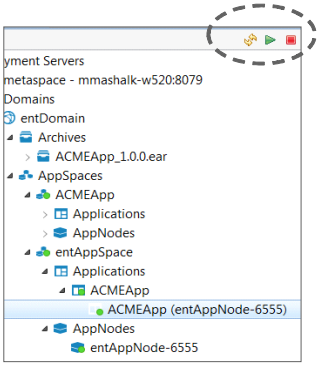
The example above shows how this can be done. The echo commands are used as a means to track progress in execution.

Deployment Choice 2: TIBCO Business Studio

- Deployment Servers View
  - Connect using **bwagent**



  - Drag application to AppSpace
  - Build and drop EAR
    - On ***Archives*** == upload only
    - On ***AppSpace*** == Upload and deploy



- Start / stop options:Application, AppSpace, AppNode

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Deployment Choice 2: TIBCO Business Studio

The second choice in Application Deployment is provided through TIBCO Business Studio.

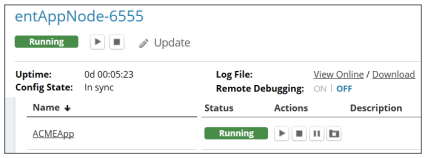
In this Deployment Server view, you can define a new runtime environment to connect. Connection to a runtime environment (also called network in this view), is accomplished through the BusinessWorks agent. If the connection to bwagent is established successfully, you will be able to browse all runtime objects in the environment. This includes Domains, Applications, AppSpaces and AppNodes. To deploy an application, simply drag the application from the project explorer to the deployment server window. The EAR file is automatically generated.

If you drop the application to the Archives object, it simply uploads the EAR file as an application archive. If the application is dragged and dropped onto an AppSpace, the EAR file is uploaded, and the application is deployed to the AppSpace.

The Deployment Server view provides options to start and stop runtime objects. Starting an AppSpace starts all AppNodes in the space. The Application in an AppSpace can be started and stopped, and an AppNode in an AppSpace can be similarly managed.

Deployment Choice 3: TIBCO Enterprise Administrator

- Complete Application Lifecycle Management
  - Create all runtime objects
  - Upload & Deploy Application
  - Start/ stop: Application, AppSpace, AppNode
- Additional options
  - View Logs
  - Update properties
  - Manage process starters



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Deployment Choice 3: TIBCO Enterprise Administrator

The final deployment choice is provided by TIBCO Enterprise Administrator (TEA)

All actions available in bwadmin command line are available in TEA. Applications, AppSpaces and AppNodes can be started and stopped.

In addition, TEA provides more options that are not available in the other tools. AppSpace, AppNode and application log files can be viewed or downloaded. You can also update properties for these components.

Finally, you can enable or disable process starters for an application as needed.

Review

- Enterprise mode uses bwagent and TEA
- Application always deployed to an AppSpace
- Deployment: All AppNodes in an AppSpace
- Bwagent: communication and metadata
- TEA: Complete application lifecycle management
- Multiple deployment choices for Enterprise mode

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Review

Lab Overview

- Setup Enterprise
- Deploy Using BusinessStudio

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software

Lab Overview

Notes

Notes

Lab P Application Deployment

Overview

In this lab you build an EAR file and then deploy it using the BusinessWorks administrator command line interface. You will define a domain, an appspace and an appnode in the process.

Objectives

- *Deploying an Application, page 512*

Prerequisites

- The following software installed and configured on your **EDUCLT** machine:
  - TIBCO ActiveMatrix BusinessWorks 6.3

Directions

- Complete the exercises that follow

Lab P - Exercise 1: Deploying an Application

Overview

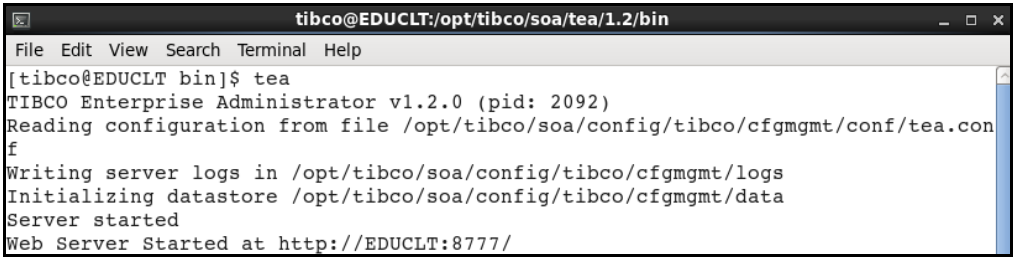
In this exercise you will deploy your application to TEA and test the solution.

Steps

1. Start TEA Server.

- Open a Terminal and navigate to `/opt/tibco/soa/bw6/tea/2.2/bin`
  - Start the TEA server using this command:

    `tea`
- Verify the Server has started:



- *Terminal* > *Set title* to **"TEA"**
- Minimize this window

2. Start bwagent.

- Open a File Browser and navigate to `/opt/tibco/soa/bw6/bw/6.3/bin`
  - Right-click **bin** folder and choose *Open in Terminal*
    - In the Terminal window, execute the following command to set the execution mode to enterprise:

      `bwadmin mode enterprise`
  - Start the bwagent (in the terminal window):

    `bwagent startagent`

```
[tibco@EDUCLT bin]$ bwagent startagent
TIBCO ActiveMatrix BusinessWorks version 6.3.2, hotfix 1, build V36, 2016-05-27
BW Agent name EDUCLT
BW Agent stores data locally
Initializing datastore using
 TIBCO ActiveSpaces (Java). Enterprise Edition. Version 2.1.6.011
...................
Initialization of datastore complete
BW Agents in the group that store data:
Name EDUCLT, Host 10.108.0.39
13:32:18.753 WARN  [main] o.e.j.s.handler.RequestLogHandler - !RequestLog
TIBCO-BW-AGENT-300002: BusinessWorks Agent started successfully.
```

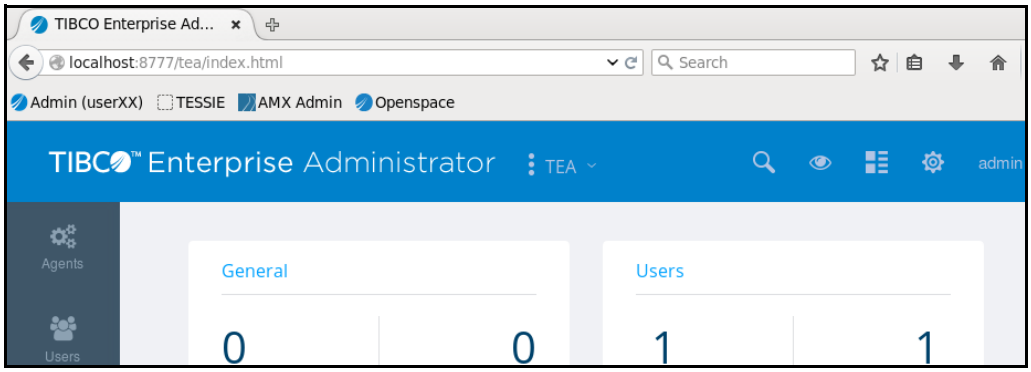
  - Note down the name of the BW Agent _____
- Terminal > Set title to **`"BWAgent"`**

3. Examine the bwagent.ini file.

- Navigate to `/opt/tibco/soa/bw6/bw/6.3/config`
  - Open **bwagent.ini** using gedit
  - Notice the port for tea.agent.port is set to 9091
  - Note down this port

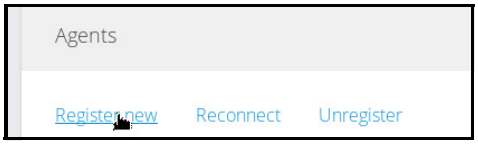
4. Login to TEA server.

- Use this URL: **`http://localhost:8777/tea`**
  - Username/password: **`admin`**

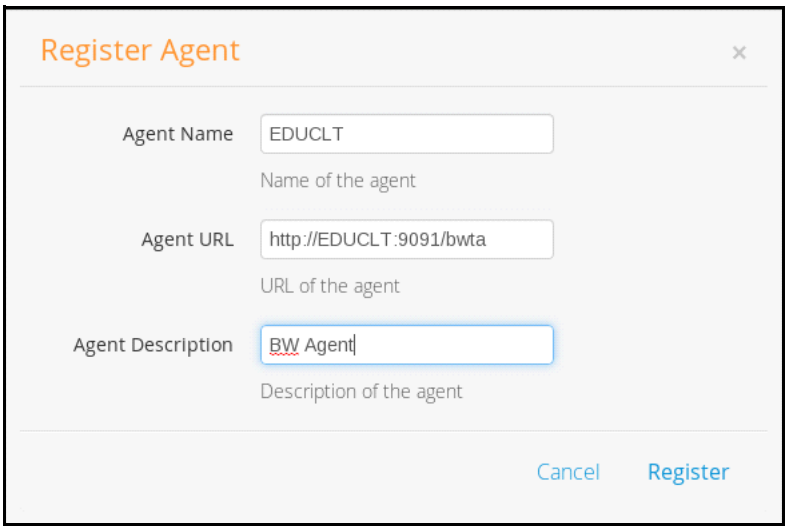


5. Register the agent.

- Click on the **Agents** icon on the left side
  - Click **Register new** link



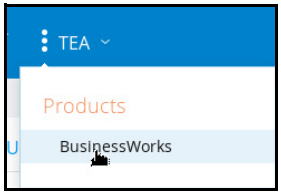
  - Agent Name: **`EDUCLT`**
  - Agent URL: **`http://EDUCLT:9091/bwta`**
  - Agent Description: **`BW Agent`**



  - Click **Register**

6. Create a new domain.

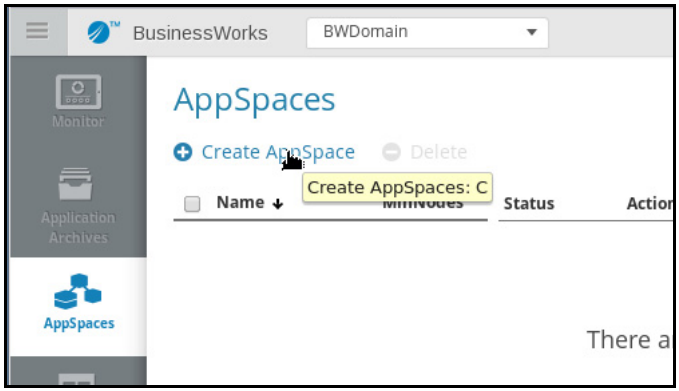
- From the top menu, select the drop-down and choose **BusinessWorks**



  - Click **Create Domain** button
  - Domain Name: **`BWDomain`**
  - Agent: **EDUCLT** (select from the drop-down list)
- Click **Create**

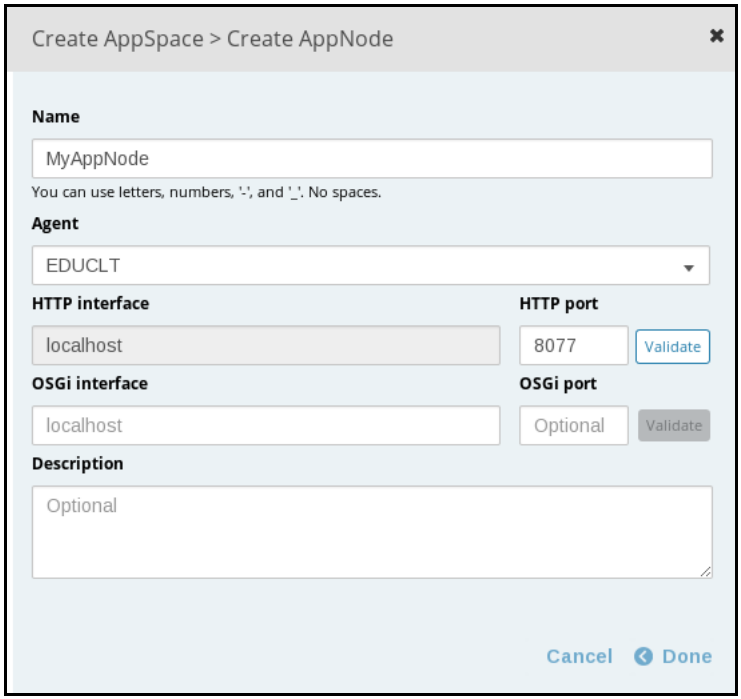
7. Create a new AppSpace.

- Click **AppSpaces** icon
  - Click the line Create AppSpace



  - Name: **`MyAppSpace`**
  - Agent: **EDUCLT**
  - Click the Create AppNode link
  - In the new window, enter the Node details:
  - Name: **`MyAppNode`**

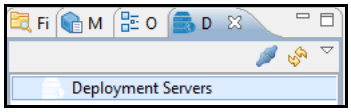
- Agent: **EDUCLT**
- http Port: **`8077`** (Click Validate and verify the port is available)



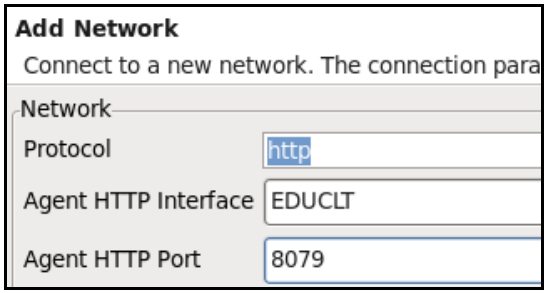
- Click **Done** and **Create**
- In the AppSpaces list, click on **Start** to start the appspace MyAppSpace

8. Create a Deployment server.

- Launch Business Studio for workspace: **`/home/tibco/bw6edu/workspace`**
- Select the *Deployment Servers* tab (below Project Explorer)



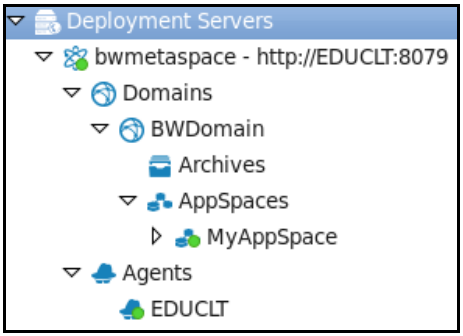
- Right-click **Deployment Servers** and select *Add Network*
- Network Parameters:
  - Agent HTTP Interface: **`EDUCLT`**
  - Agent HTTP Port: **`8079`**



  - Click **Finish**

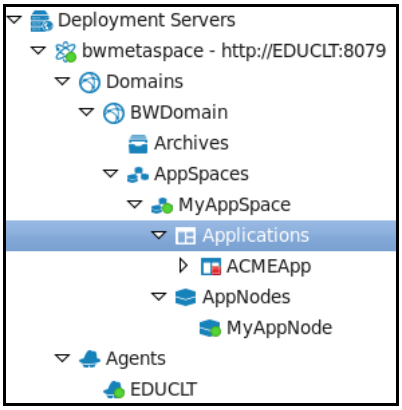
9. Examine the domain and the AppSpace.

- Under *Deployment Servers*, expand *bwmetaspace* > ***Domains***
  - Explore the available Domain



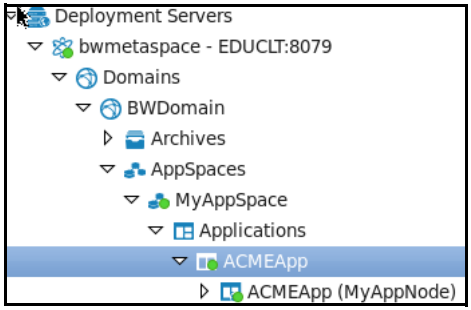
10. Deploy the application.

- Open the following projects (Open > Project)
  - ACMEApp
  - ACMEAppModule (Also opens ACMEShared)
- From *Project Explorer* view, drag **ACMEApp**
  - Drop it on *Deployment Servers* >*bwmetaspace* > *Domains* > *BWDomain* > *AppSpaces* > ***MyAppSpace***
- Verify the application is deployed by expanding Applications under *MyAppSpace* .



11. Start the application.

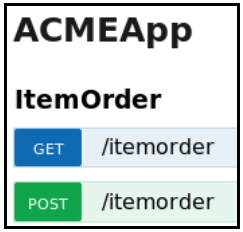
- Choose *Deployment Servers* >*bwmetaspace* > *Domains* > *BWDomain* > *AppSpaces* > *MyAppSpace* > *Applications* > ***ACMEApp***
- Right-click **ACMEApp** and choose *Start* from the menu



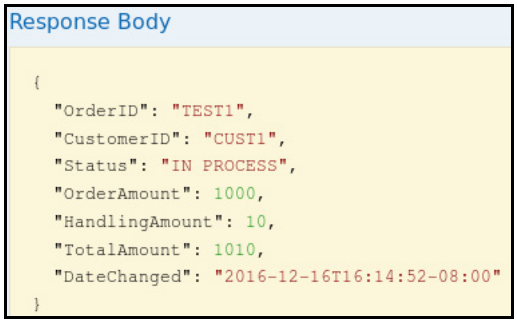
  - The color of the application icon changes from red to green
    - Click Refresh if needed to see the changes

12. Test your application using Swagger UI.

- In a browser window, open this URL:
  `http://localhost:7777/ACMEApp`
  - Click on **ItemOrder**, then on **GET**



  - **Enter OrderID: `TEST1`**
  - **Try it Out!** and verify that the results are returned, since the application is running



13. Shutdown the run time and design time environment.

- Exit Business Studio
- Back in the TEA UI, select the AppSpace MyAppSpace
  - Click the Stop icon
  - Log Out
- Back in the BW Agent Terminal window, terminate the agent using <Ctrl+c>
- Similarly, stop the TEA server
  Note: This shuts down the run time environment

Notes

Unit 17 Using TIBCO ActiveSpaces

This unit will provide an introduction to the TIBCO ActiveSpaces architecture and terminology. Installation steps and prerequisites are also discussed.

Topics

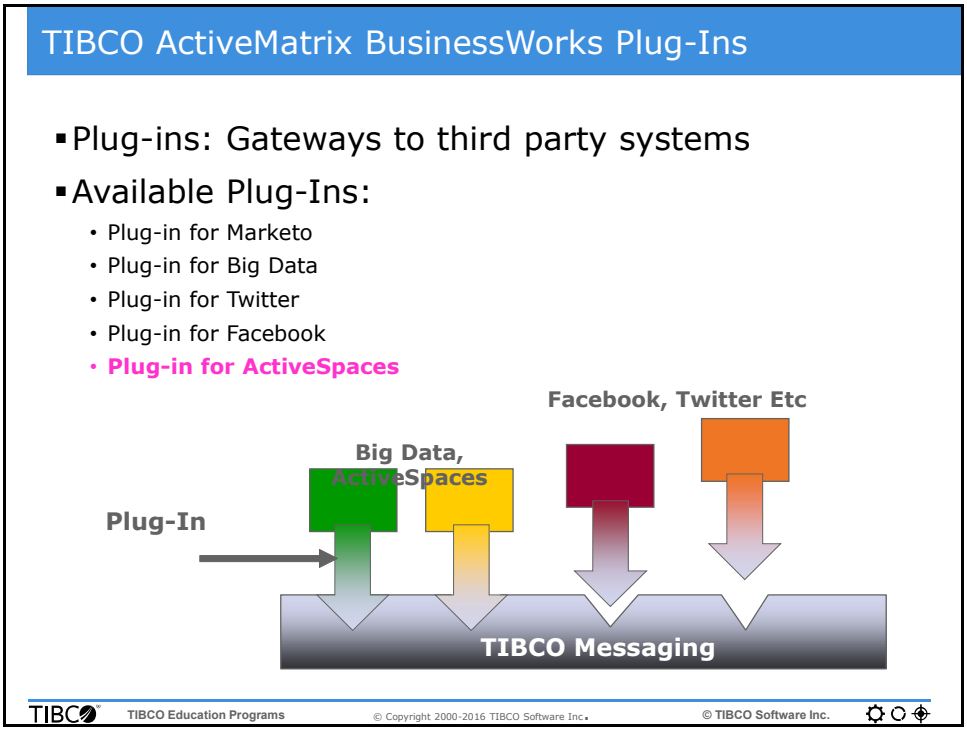
- *Objectives,* page 524
- *TIBCO ActiveMatrix BusinessWorks Plug-Ins,* page 525
- *Fast Data and TIBCO ActiveSpaces,* page 526
- *Data Grid: An Introduction,* page 527
- *ActiveSpaces Enables TIBCO Products,* page 528
- *ActiveSpaces and BusinessWorks,* page 529
- *Installing TIBCO ActiveSpaces,* page 530
- *ActiveSpaces Components,* page 531
- *Plug-in Capabilities,* page 532
- *Installing BusinessWorks Plug-In for ActiveSpaces,* page 533
- *Plug-in Installation Pre-Requisites,* page 534
- *ActiveSpaces Palette Details,* page 535
- *Review,* page 536

Objectives

- Introduce TIBCO ActiveSpaces features
- Discuss use of ActiveSpaces in TIBCO products
- Identify key ActiveSpaces components/utilities
- Install and verify ActiveSpaces plug-in

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

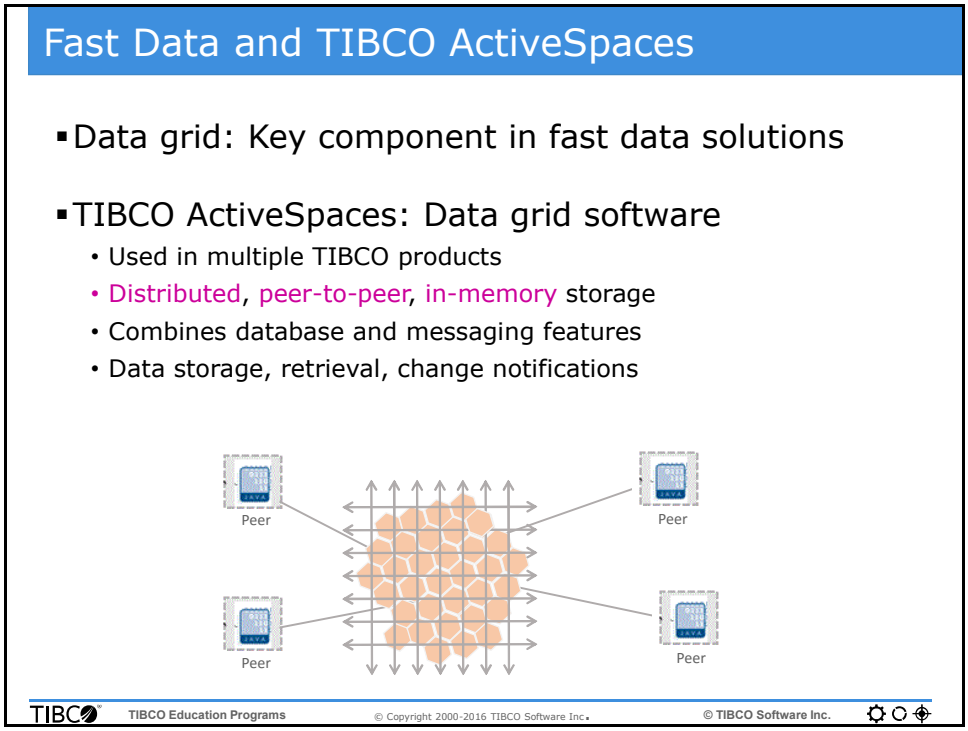
Objectives



TIBCO ActiveMatrix BusinessWorks Plug-Ins

An adapter or plug-in allows for bi-directional communication between the application and the messaging layer.

Plug-ins can be developed using the Plug-in Development Kit (PDK) for BusinessWorks 6.x.



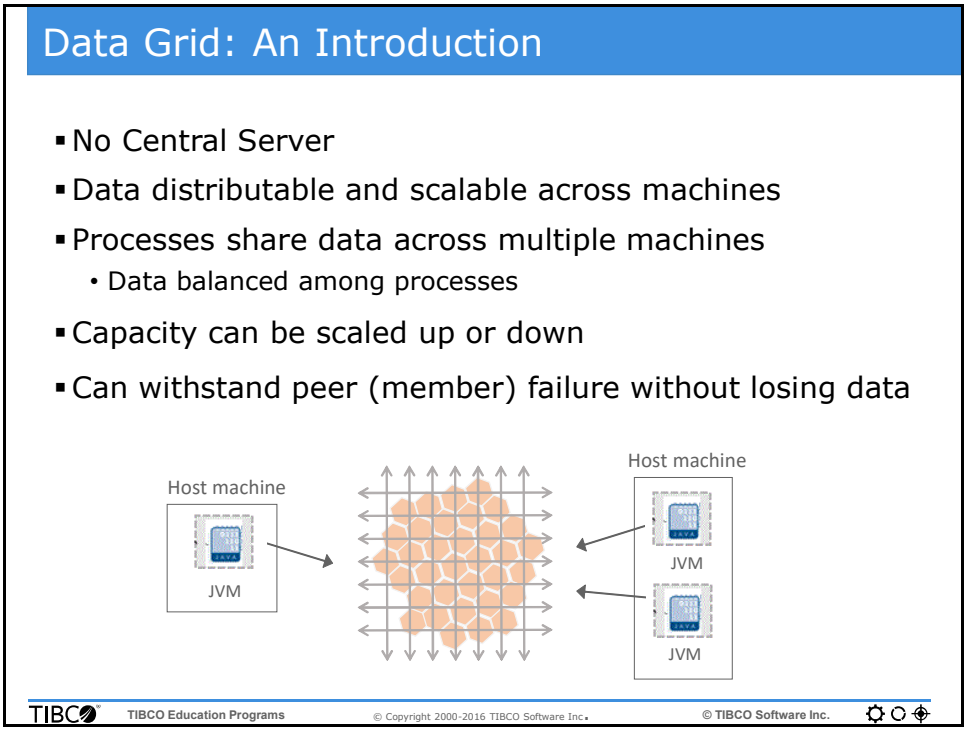
Fast Data and TIBCO ActiveSpaces

ActiveSpaces is data-grid software, which is software that combines features of a database with features of a middleware messaging system.

ActiveSpaces manages dynamic data with the use of in-memory data structures called Space. Each space is like a table in a database.

With ActiveSpaces, it is easy for you to create distributed applications that exchange and modify data shared between processes and across a network, using a much simpler programming model than alternative distributed object programming approaches or database "sharding".

You can use ActiveSpaces to implement Space-based architectures, which use a data grid approach for data storage and processing. Data storage and access is virtualized, by storing data in a shared Data communication and process coordination are both handled by the space, and data access is transactional.

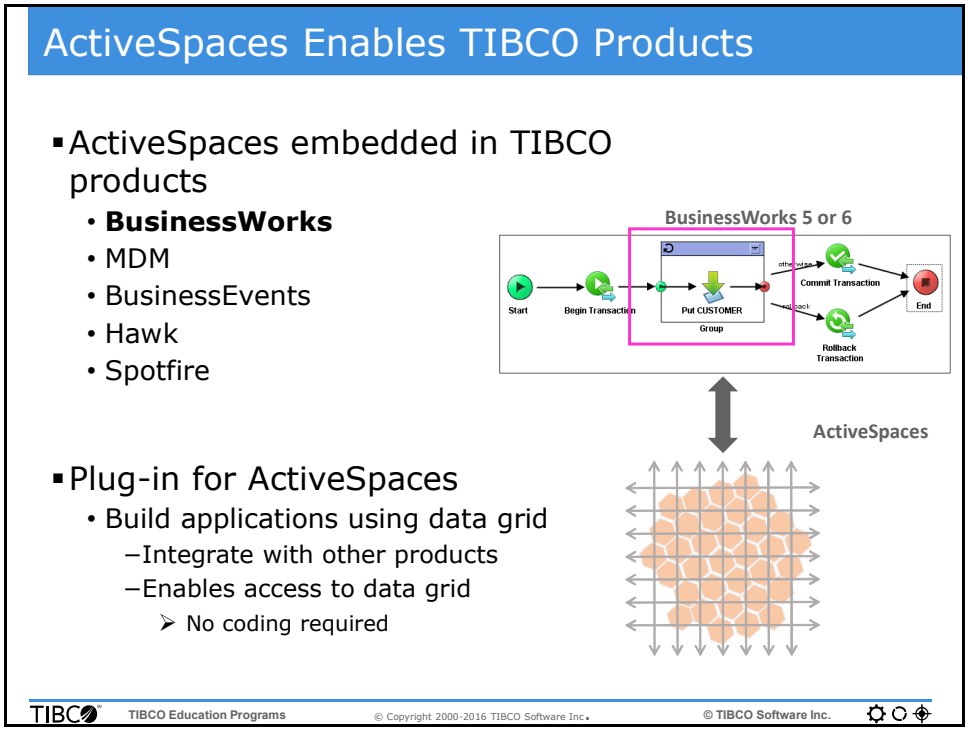


Data Grid: An Introduction

Data grid: An architecture that enables you to access, modify and transfer extremely large amounts of distributed data. Data may be geographically distributable and is generally spread out across multiple machines. These machines are sometimes referred to as the grid.

One of the major features of a data grid is that it always has current data, so it is dynamic. Since the data grid has extremely large amounts of data, the volume and velocity of these changes is large. Traditional tools such as databases are too slow to maintain this data. In-memory solutions are typically recommended.

In-memory data grids must also support the ability to replicate data to tolerate process and machine failures.

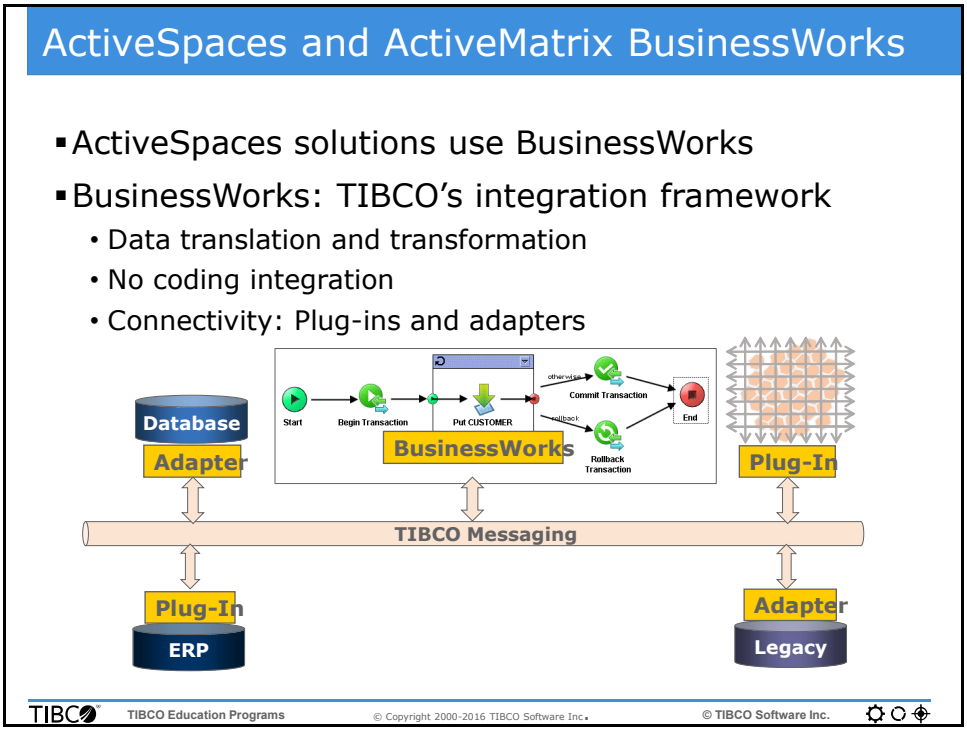


ActiveSpaces Enables TIBCO Products

ActiveSpaces is also being embedded in other TIBCO products, such as BusinessEvents and Master Data Management (MDM). ActiveSpaces is compatible with other TIBCO products.

ActiveSpaces has a BusinessWorks plug-in that allows you to build ActiveSpaces applications directly inside BusinessWorks processes. This allows for integration with other TIBCO products using ActiveSpaces.

BusinessEvents, BusinessEvents Extreme, and Spotfire all have ActiveSpaces communication channels. ActiveSpaces is also used for internal product communication and data storage in TIBCO products such as BusinessEvents, MDM, and Hawk.



ActiveSpaces and BusinessWorks

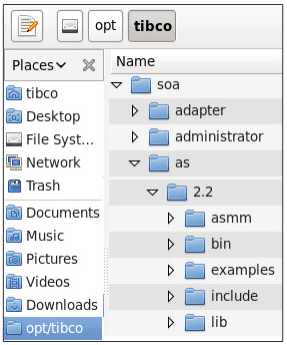
TIBCO ActiveMatrix BusinessWorks forms the integration framework for your Enterprise Service Bus implementation. Within this framework, you can develop automated processes for example, that take data sent by one adapter, and transform the data to be received by another adapter. The integration framework has monitoring, management and security built-in, so that you can quickly "lock down" your integration. Messaging is also included as part of this framework.

Using adapters and messaging, BusinessWorks connects to applications of different types and databases to integrate all aspects of your enterprise. Similarly, BusinessWorks plug-ins provide connectivity to newer applications such as Facebook and Twitter.

Business Works plugs into the bus infrastructure and supports TIBCO messaging options such as Enterprise Message Service (EMS) and Rendezvous. It helps in integrating applications through its data transformation and translation capabilities.

Installing ActiveSpaces

- Installation (32 or 64-bit)
  - **Ex: TIB_activespaces_2.2.0_linux_x86_64.zip**
- Post-Installation
  - Verify installation folders
  - Configure environment variables
    - Shut down all TIBCO applications
      - **AS_HOME**, **JAVA_HOME**
      - **PATH** and **CLASSPATH**
  - Run ActiveSpaces components
    - Verify data grid can be created



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Installing TIBCO ActiveSpaces

To install the Enterprise Edition of ActiveSpaces 2.x, run the **TIBCO Universal Installer** included with the product package.

Once installed, add environment variables used by ActiveSpaces:

- AS_HOME
- JAVA_HOME

Configure the PATH and CLASSPATH variables to point to the ActiveSpaces /bin and /lib directories.

In FileSystem, verify the new ActiveSpaces parent folder and sub-folders were created as shown here.

ActiveSpaces Components

- ActiveSpaces Agent: as-agent
  - *Easiest way to test installation*
- TIBCO API: Java, C and .Net
  - Run sample API programs: ASPaint, ASChat, others
- Administration utilities
  - ASMM (web-based, read only)
  - as-admin (command line)
- *BusinessWorks plug-in for ActiveSpaces*
  - Discussed next

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

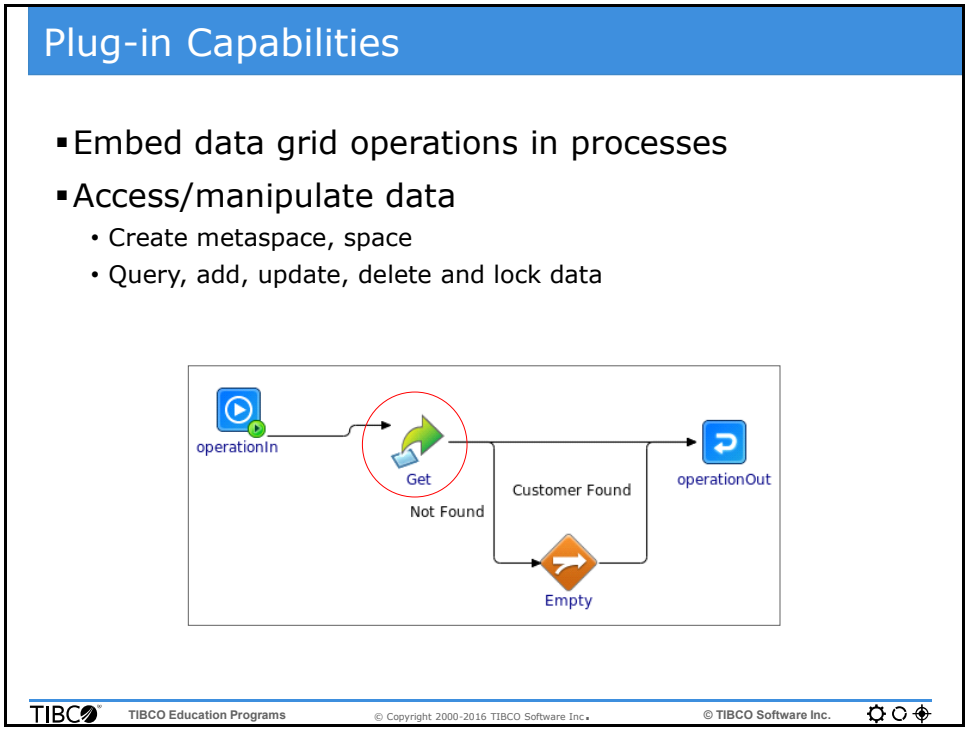
ActiveSpaces Components

ActiveSpaces provides a pre-built Agent process called **as-agent** that joins all distributed spaces in a metaspace and keeps the metaspace “alive”. This is the fastest way to test a successful installation.

An **API** is provided that enables coherent data storage and retrieval, network data transport, and network data representation. The APIs include Java, C, and .NET libraries.

ActiveSpaces provides two administration tools:

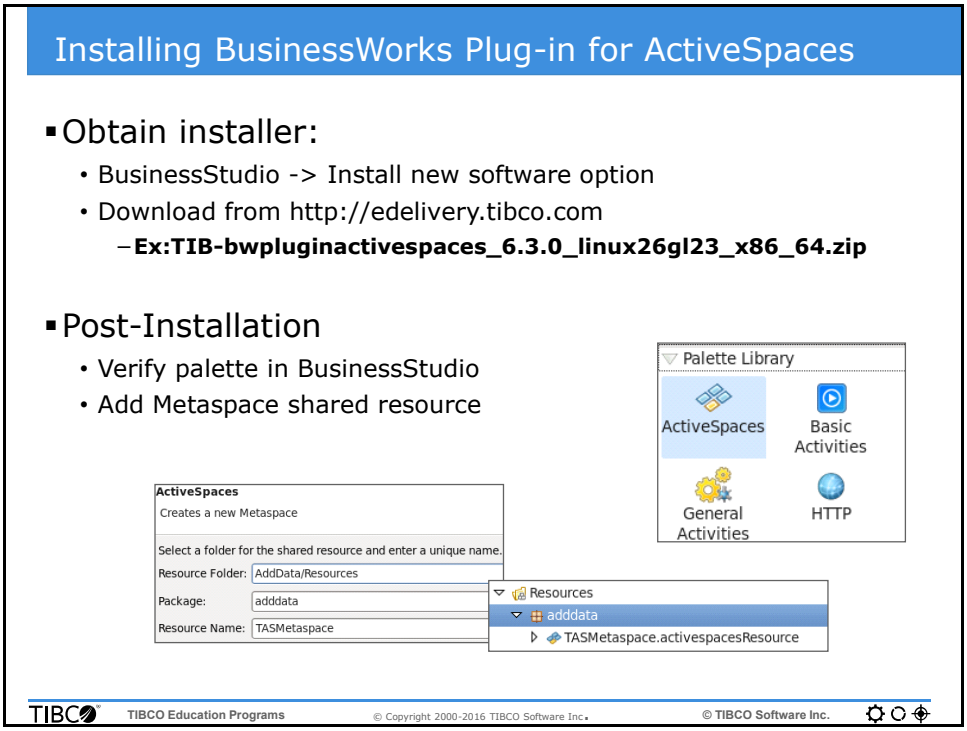
- **as-admin** - command line utility used to connect to a metaspace, create, modify or remove a space, and display information about existing spaces and members.
- **ASMM** - administrative GUI that lets you connect to a metaspace and display information about spaces and other system resources.



Plug-in Capabilities

When you install the BusinessWorks Plug-in for ActiveSpaces, the ActiveSpaces palette is automatically added to TIBCO BusinessStudio.

Using the plug-in, you can build ActiveSpaces applications directly inside BusinessWorks processes. Placing the BusinessWorks plug-in in the middle of your BusinessWorks solution provides access to data maintained in the data grid.



Installing BusinessWorks Plug-In for ActiveSpaces

You can directly download and install the ActiveSpaces using TIBCO Business Studio.

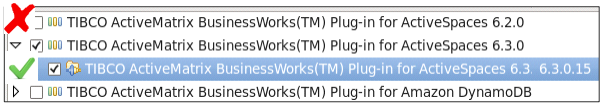
In the Install Software section of Business Studio, simple enter this url: **http://update.tibco.com/eclipse/bw/6.3**

This locates all available plug-ins for the current version of BusinessWorks.

You can also download the plug-ins from TIBCO's software delivery site http://edelivery.tibco.com for offline installation.

Plug-in Installation Pre-Requisites

- Pre-requisite software
  - BusinessWorks 6.3.x
  - ActiveSpaces 2.2.x+ Enterprise Edition
  - Java: 32 or 64-bit, must match ActiveSpaces version
- Install Plug-in last, use latest version
  - Do not install multiple versions
  - BusinessWorks Plug-in for ActiveSpaces 6.3.0



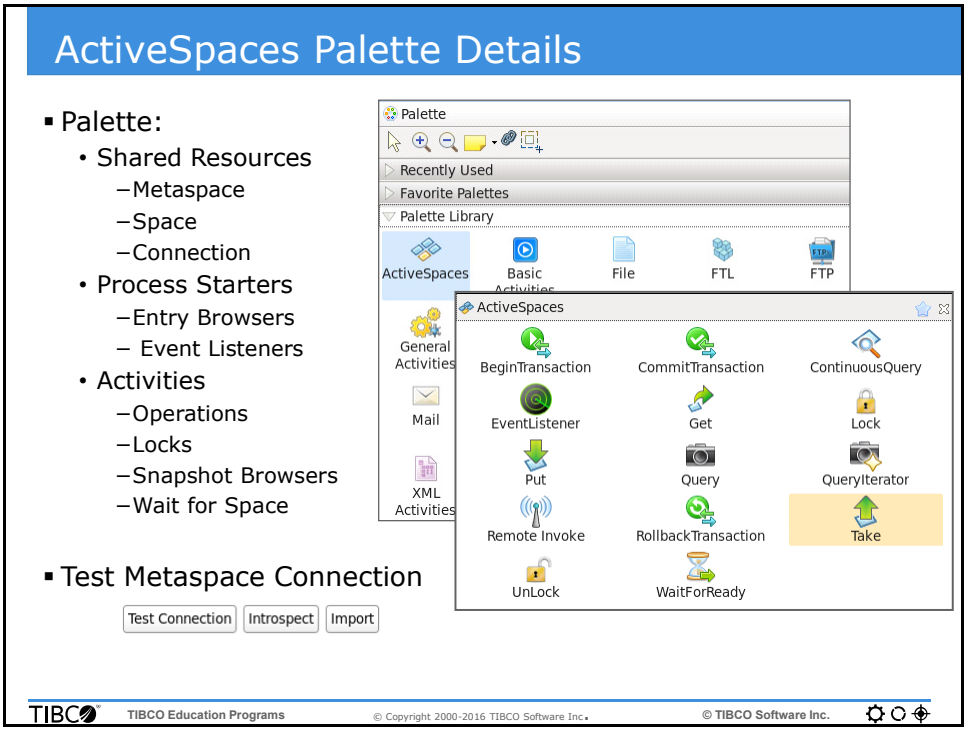
TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Plug-in Installation Pre-Requisites

The pre-requisite software installation requirements for the ActiveSpaces Plug-in are identified above.

When you install the plug-in , remember to install only one version of the plug-in.

Also, remember to install the latest version as shown above.



ActiveSpaces Palette Details

TIBCO ActiveMatrix BusinessWorks Plug-in for ActiveSpaces provides the following features:

- Add, retrieve, and delete tuples in a space.
- Invoke, commit, and revert space transactions.
- Iterate through the content in a space and listen for the incoming events in a specified space.
- Retrieve entries from a space according to specified filter conditions.
- Lock and unlock tuples in a space.
- Remotely invoke members over a space.

Review

- Data grid: key component of fast data solutions
- ActiveSpaces: In-memory, distributed data grid
  - Java & C API, Administration utilities, agent, plug-ins
- BusinessWorks Plug-in for ActiveSpaces:
  - Embeds ActiveSpaces operations in processes

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Review

Lab Overview

- Install TIBCO ActiveSpaces software
- Configure linux environment for ActiveSpaces
- Create a BusinessWorks process and create a metaspace resource

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Lab Overview

Notes

Lab Q Enabling ActiveSpaces

Overview

In this lab you configure your environment to support ActiveSpaces 2.x. You will then install the TIBCO ActiveMatrix BusinessWorks Plug-in for ActiveSpaces 6.1.0 and verify that it works.

Objectives

- *Setup Environment,* page 540
- *Install and Verify Plug-in,* page 542

Prerequisites

- The following TIBCO Software is installed and configured on your student machine:
  - BusinessWorks 6.3.x
  - ActiveSpaces 2.2.x
  - BusinessWorks Plug-in for ActiveSpaces 6.3.0
- Student lab files extracted

Directions

Complete the exercises that follow.

Lab Q - Exercise 1: Setup Environment

Overview

In this exercise you will verify that your can run the ActiveSpaces agent and connect to the default metaspace.

Steps

1. Define a script file to set environment variables and PATH.

- Using *File Browser*, go to `/home/tibco` and create a new file **`BW6Env`**
  - Open the file using **gedit**
  - Add the following entries to the file:

```
export AS_HOME=/opt/tibco/soa/as/2.2
export PATH=$AS_HOME:$AS_HOME/bin:$AS_HOME/lib:$PATH:
export CLASSPATH=$AS_HOME/lib:$CLASSPATH:
export LD_LIBRARY_PATH=$AS_HOME/lib:$LD_LIBRARY_PATH
```

- Save and close the file

2. Run the as-agent.

- Open a new Terminal window and set the environment variables

  `source BW6Env`

- Run the TIBCO Activespaces Agent

  `as-agent`

- Verify the as-agent process runs without any network errors:

```
TIBCO ActiveSpaces Agent
Enterprise Edition. Version: 2.2.0.017 20-May-2016 SVN Rev.29736
Copyright(c) 2009-2016 TIBCO Software Inc. All rights reserved
*************************************************************
[2017-01-13T15:23:27.990][26072][139725304112928][INFO][transport] ip_address=10
.108.0.39 port=50000
[2017-01-13T15:23:33.012][26072][139725081925376][INFO][ms.$gmp] ---------------
----------------
[2017-01-13T15:23:33.012][26072][139725081925376][INFO][ms.$gmp] configuring new
 manager
[2017-01-13T15:23:33.012][26072][139725081925376][INFO][ms.$gmp] manager = as-ag
ent-a6c0027-c350 (a6c0027-c350-58796170-b, 10.108.0.39:50000)
[2017-01-13T15:23:33.012][26072][139725081925376][INFO][ms.$gmp] self node = as-
agent-a6c0027-c350 (a6c0027-c350-58796170-b, 10.108.0.39:50000)
[2017-01-13T15:23:33.012][26072][139725081925376][INFO][ms.$gmp] local node is M
ANAGER now
[2017-01-13T15:23:33.012][26072][139725081925376][INFO][ms.$gmp] ---------------
----------------
[2017-01-13T15:23:33.102][26072][139725078767360][INFO][ms.metaspace] using site
 definition: type=[local] site=[default] routed=[false]
[2017-01-13T15:23:33.115][26072][139725304112928][INFO][ms.metaspace] Connected
metaspace : name=[ms] listen=[tcp://10.108.0.39:50000] discovery=[tibpgm] autoJo
inRole=[seeder] member_name=[as-agent-a6c0027-c350] version=[2.2.0.017]
```

- If you see any errors, ask your instructor for assistance
- Type <**`Ctrl+C`**> to terminate this as-agent process

Importante dar de baja el proceso.
El as-agent usa el puerto 50000 por default y en la pp 570 Lab R Ex 1. Se pide configurar una conexión al MetaSpace. Si este proceso está activo, el Lab R no funcionará.

Tip: This step verifies that the environment variables have been setup correctly and that the default ports are available. *as-agent* looks for an existing metaspace. If none is available, it sets up a default metaspace named MS and connects to it.

You do not need to run as-agent after this step.

Lab Q - Exercise 2: Install and Verify Plug-in

Overview

In this exercise you will install the ActiveSpaces plug-in and test success by creating a metaspace connection.

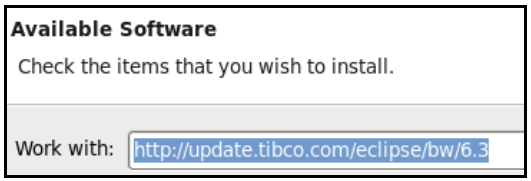
Steps

1. Launch TIBCO Business Studio and define a new workspace.

- Open a Terminal window and enter the following command
  `source BW6Env`
- Change directory to **`/opt/tibco/soa/bw6/studio/4.0/eclipse`**
- Enter the following command to launch TIBCO Business Studio
  `./TIBCOBusinessStudio`
  - Workspace: **`/home/tibco/bw6edu/workspace`**

2. Add a URL to locate BusinessWorks plug-ins.

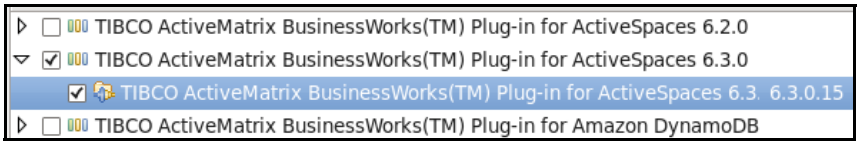
- Select *Help* > ***Install New Software...*** > ***Add*** and Enter the Location as shown below



- Press **<Enter>**
- Observe the *pending* message followed by a list of plugins

3. Select appropriate ActiveSpaces plug-in and install it.

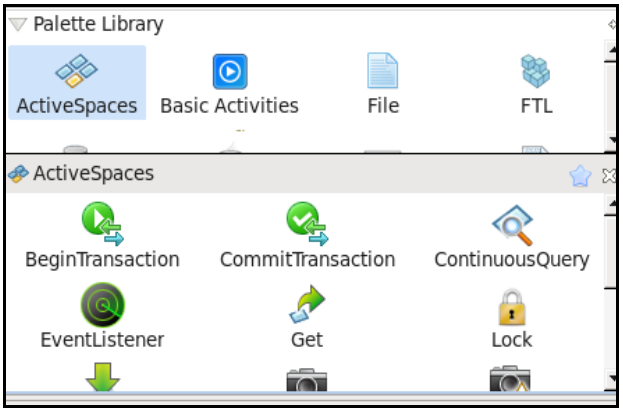
- Select the **ActiveMatrix BusinessWorks Plug-in for ActiveSpaces** as shown below
  - Verify that you are installing version **6.3.0.15** only as shown below



- Select **Next** at the **Availabe Software** window
- Select **Next** at the **Install Details** window
- Select **I accept the terms of the license agreements**
- Select **Finish,** observe that the software begins installation
- Select **OK** at the Security Warning screens
- Select **Restart Now**

4. Verify that the ActiveSpaces palette is available in the Palette Library.

- Browse to and open *StartEnd*> *Processes* > ***StartEndProcess***
- Expand the **Palette Library** and select the **ActiveSpaces** palette
- Observe the ActiveSpaces palette activities that are now available



5. Connect to the default metaspace to test ActiveSpaces configuration.

- In the Project Explorer, right-click on select *New* > *BusinessWorks Application Module*
  - Project name: **`ActiveSpaces`**
  - Accept remaining defaults and click Finish
- In Project Explorer, expand *ActiveSpaces* > ***Resources***
- Right-click and select *New*> ***Metaspace*** and click **Finish**
- In the Metaspace Editor click **Test Connection**
- Observe that the **Connect to Metaspace Successfully** message appears

- **Save** your workspace
- Close Business Studio when done

Tip: If you get a "connecting to metaspace..." message, and the message persists after 30 seconds, it indicates that Business Studio cannot successfully set up and connect to metaspace. This happens when the environment is not set correctly, and you have not started Business Studio from the command line.

To fix: Terminate Business Studio and check your environment settings in BW6Env. Then launch Studio from the command line, as mentioned earlier in this exercise.

Notes

Notes

Unit 18 Working with Data

In this unit we discuss defining spaces, and data operations that can be performed by members on spaces.

Topics

- *Objectives,* page 548
- *How a Data Grid Works,* page 549
- *Data Grid Terminology - Spaces and Metaspaces,* page 550
- *ActiveSpaces Components in Pictures,* page 551
- *Tuple Definition,* page 552
- *Designing Applications with ActiveSpaces Plug-in,* page 553
- *Adding a Metaspace,* page 554
- *Space and Space Fields,* page 555
- *Space Properties,* page 556
- *Defining a Space Connection,* page 557
- *Creating a Space Tuple,* page 558
- *Retrieving Space Tuple with Get Operation,* page 559
- *Retrieving Space Tuple with Take Operation,* page 560
- *Command-Line Administration,* page 561
- *GUI Based Administration,* page 562
- *Viewing Metaspace and Spaces,* page 563
- *Viewing Space and Member Data,* page 564
- *Review,* page 565
- *Lab Overview,* page 566

Objectives

- Explain how BusinessWorks uses ActiveSpaces
- Explore the ActiveSpaces Designer palette
- Examine operations for working with space data
- Differentiate between Get and Take operations
- Build a process to retrieve and insert data

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Objectives

How a Data Grid Works

- Processes (Members) join grid, can contribute resources
  - Memory and CPU
  - More processes = larger data grid
- Data Replication
  - Multiple copies stored across data grid
  - Provides fault tolerance
- Data Distribution
  - Different data records stored in different parts of the grid
  - Re-balanced when processes join/leave
  - Distribution can be tuned (e.g. data centers)

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

How a Data Grid Works

Applications (called Member Processes) running on machines on the network form a cluster. Each process contributes storage capacity (RAM and/or disk) and CPU (to service requests to read/write data). An increase in the number of processes produces a corresponding increase in the memory and processing power available to the data grid.

The data grid automatically evenly distributes data between the contributing processes: the resources of all the contributing machines are pooled together.

Data is automatically redistributed when processes are added or removed from the data grid.

Optional features also allow you to control how data is distributed across data centers.

Data Grid Terminology – Spaces and Metaspaces

- **Space:** a structured container for data records
  - Like a database table, accessed by one or more processes
- **Metaspace**: manages spaces and members
  - Like a database, data stored by processes
  - Processes connect to a metaspace, access data in spaces
- **Member** (Peer)
  - Application/process running on a host
  - Optional: Elects to contribute resources to a space
  - Can join several spaces

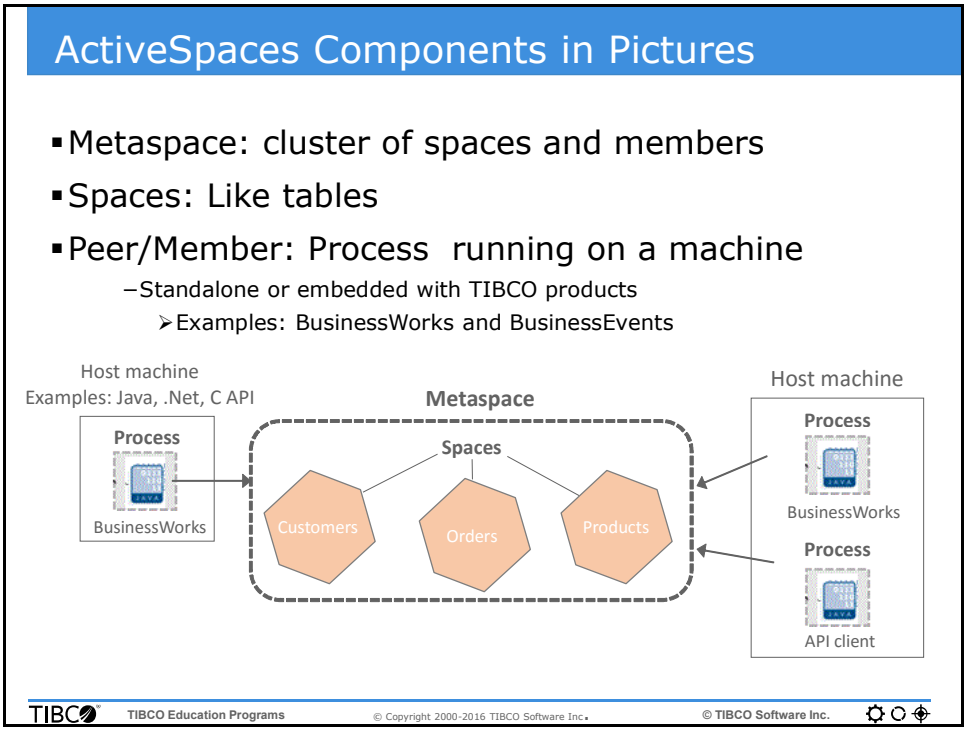
TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Data Grid Terminology - Spaces and Metaspaces

Peer processes share resources across the network to store data in distributed data structures called a **Space**. A space, like a database table, stores a particular type of structured data.

A **metaspace** is a set of network communication parameters that allows member processes to share data records in Spaces. Each Space (similar to a database table) is a container with a specific Schema that stores data used by applications.

Each **Peer** process (also called a member) that joins a space can elect to contribute resources to keep a space alive, such as CPU and Memory. It is fine for a process to join several spaces and elect to contribute resources to each of those spaces, or some or none of those spaces.

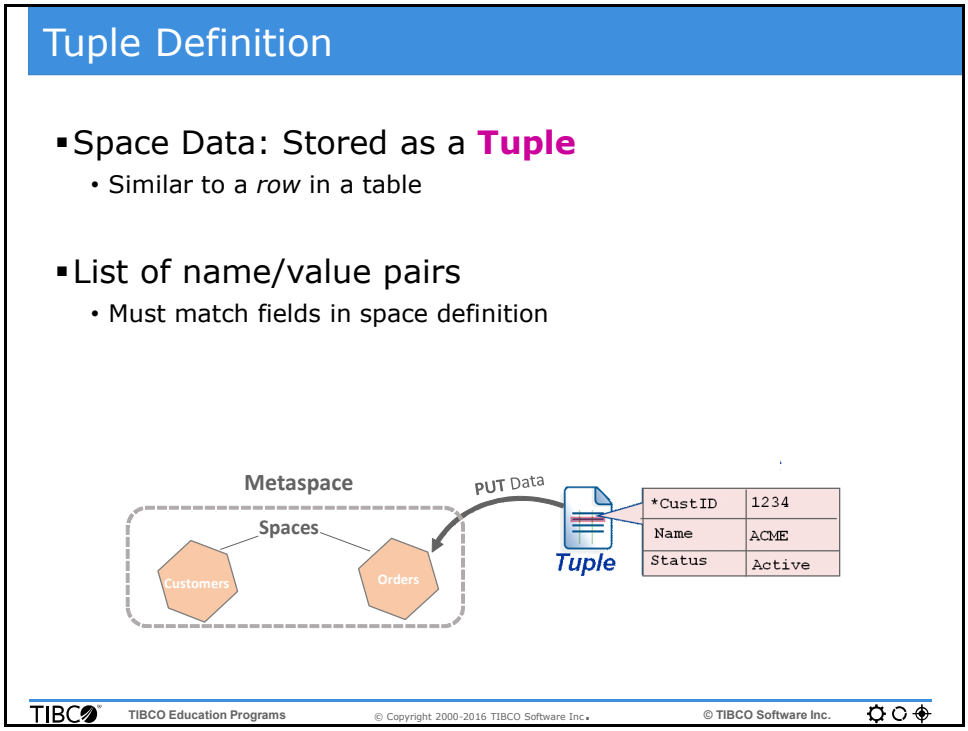


ActiveSpaces Components in Pictures

ActiveSpaces is TIBCO's data grid and uses the same familiar terminology used in most other data grids: metaspace (cluster), space and peers. It provides an infrastructure for building highly scalable, fault-tolerant distributed applications. It combines the features and performance of databases, caching systems, and messaging software to support very large, highly volatile data sets and event-driven applications.

Unlike many other database cache technologies, ActiveSpaces is also optimized for many write operations, not just reads.

ActiveSpaces can be installed as a standalone application (Enterprise Edition) where it can be accessed by client applications such as BusinessWorks (separate ActiveSpaces license is required). It can also be accessed using a Java, C or .Net API.



Tuple Definition

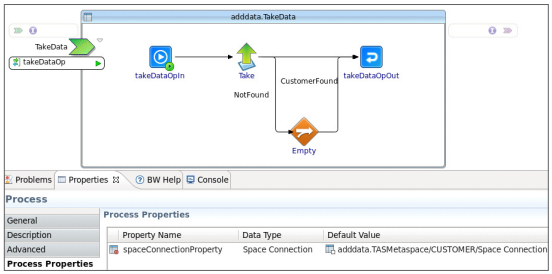
You put data into a space by creating a data structure called a **Tuple** and then putting the tuple into the space. A tuple is a unit of data in a space defined by a set of typed fields (e.g. the Space schema), similar to a row of data in a database table.

When a tuple is stored into a space, the fields that it contains must match the names and types of the fields described in the space definition. If there is a type mismatch between a field contained in the tuple and the type of the field defined in the space field definition, ActiveSpaces may perform an automated field conversion if possible, or the operation will fail.

Fields flagged as "Nullable" are optional, so you do not have to fill in these fields for the PUT or UPDATE operation to succeed.

Designing Applications with ActiveSpaces Plug-In

- In Application Module, add Metaspace Shared Resource
  - Contains: Metaspace, Space, Space Connection
- Add ActiveSpaces activities to process
  - Reference shared resource
- Run BusinessWorks application, test and verify



TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Designing Applications with ActiveSpaces Plug-in

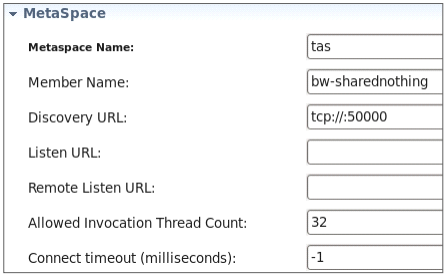
Processes first connect to a metaspace, then access one or more Spaces to access data. This relationship is defined as Shared Resources in a BusinessWorks project when using the BW Plug-in.

You must add a Metaspace shared resource to your BW application to access data in ActiveSpaces.

Drag one or more activities from the ActiveSpaces palette into the design panel, and then create transitions between each activity on the design panel. You also can select activities to add from other palettes in the Palettes panel, for example, JDBC activities.

Adding a Metaspace

- Metaspace Shared Resource details
  - Name
  - Discovery URL
    - How to connect to the metaspace
  - Listen URL
    - How members communicate with each other
  - Introspect
  - Import
- Test Connection
  - Must do before proceeding



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

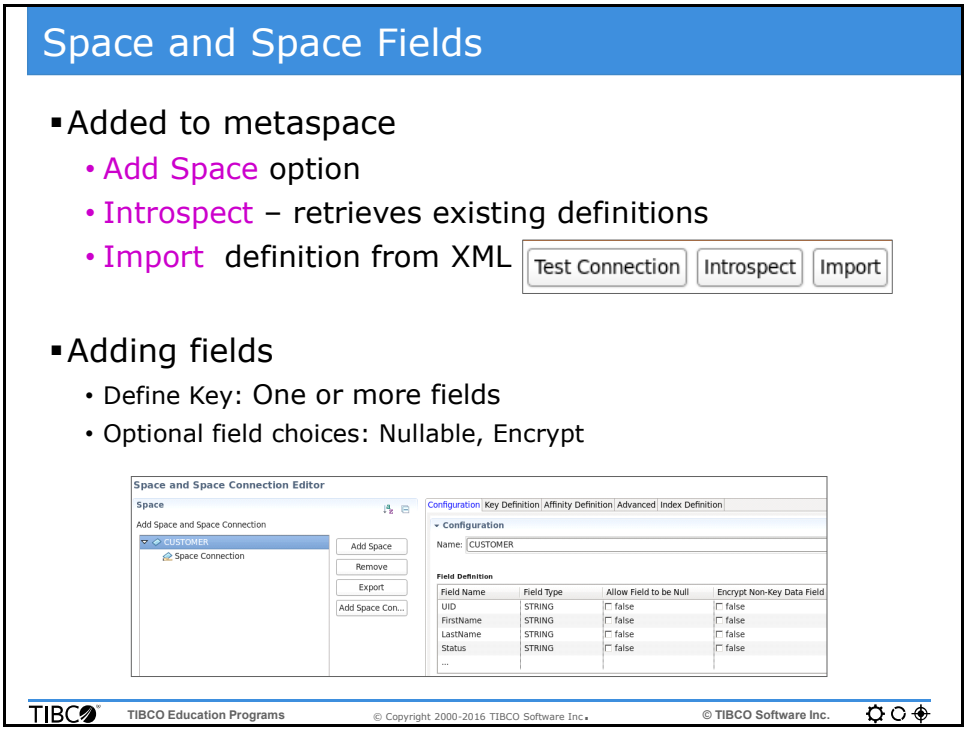
Adding a Metaspace

In BusinessWorks Plug-in for ActiveSpaces, Metaspace is a BusinessWorks shared resource representing a metaspace.

You can drag a Metaspace shared resource, then configure its properties.

The **Test Connection** button allows you to test if your Discovery URL and Listen URL are allowed on the local system and the current network.

Multiple metaspaces are supported for a single project. However, each metaspace must be uniquely named.



Space and Space Fields

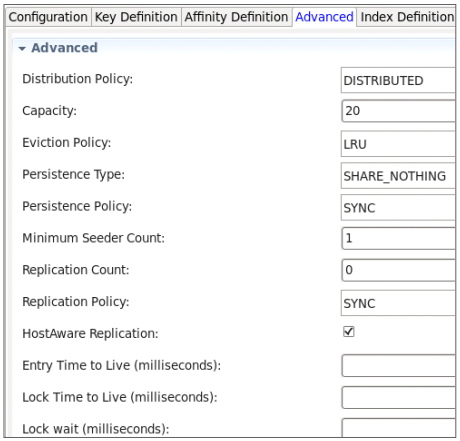
You must set one or more fields as the key or keys when manually adding fields in a space. If a space is imported or introspected, it is not necessary to add field definitions. A field is associated with a name, a type, whether or not the field is the key, and whether or not the field is nullable. You decide which field or fields should be used as the Key for the Space in the Key tab. In this tab you browse for all currently defined fields, then check the check box for each field you want to be included in the Space key.

Note: The Space key values are used in the internal distribution algorithms used by every Space operation.

You can also define additional Indices to speed up queries. A runtime administrator can also add or drop a Space Index using the runtime as-admin tool. Non-key fields can also be allowed to be encrypted if you check the "Encrypt Non-Key Data Field", but the data will only actually be encrypted if the Metaspace containing the Space implements a secure Metaspace that supports and allows data encryption.

Space Properties

- Configured on the Advanced tab
  - Distribution
  - Replication
  - Locking
  - Data Eviction
  - Persistence
  - Operation Timeouts



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Space Properties

The Advanced tab contains additional space properties that can be optionally configured.

A space is Distributed by default, but you can choose to make it Non-Distributed, which means every entry will be controlled and stored on a single seeder member.

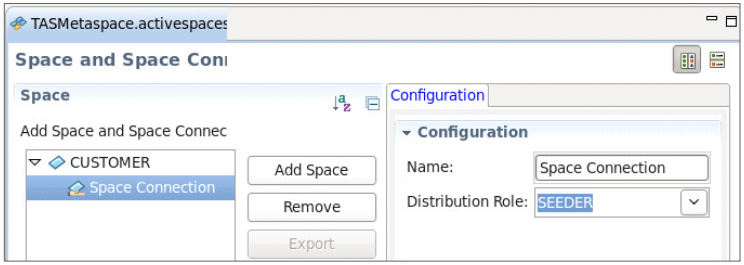
You can also configure the amount of Replication, to provide high availability of space data. The default Update Transport is Unicast, so there is no requirement to use Multicast with ActiveSpaces.

The Minimum Seeder Count will block a space from being used until enough seeders are connected to provide this level of replication.

Data locking, persistence, and eviction policy can also be configured here.

Defining a Space Connection

- Space Connection – used by activities
  - Configuration in Metaspace Resource
- Connection always linked with a Space
  - Can add multiple connections
  - Distribution Role: SEEDER (default) or LEECH

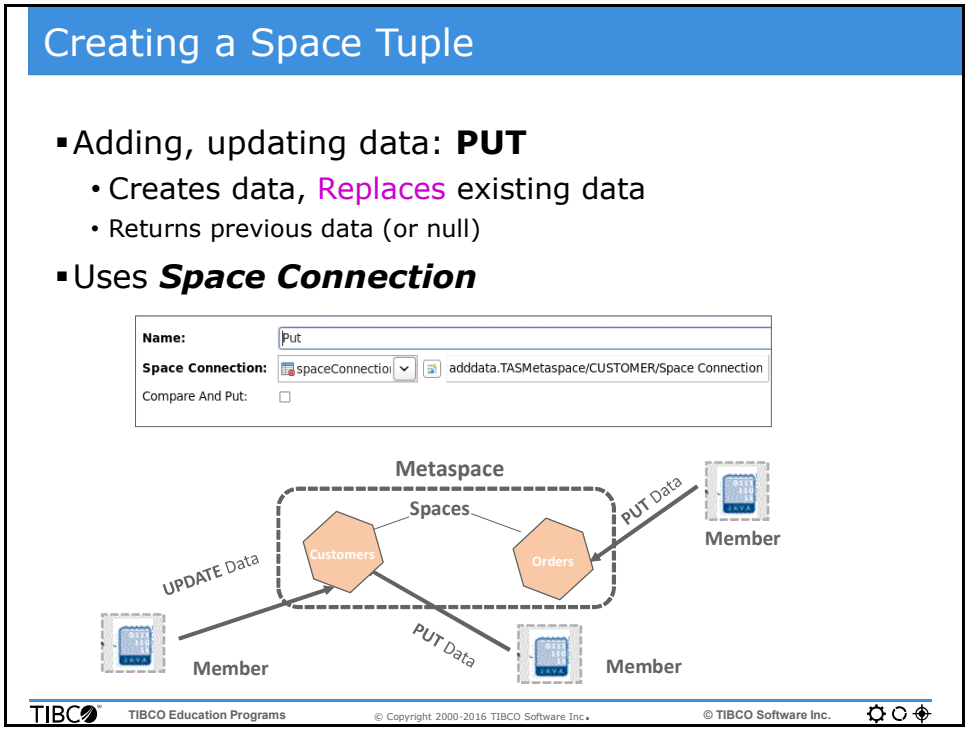


TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Defining a Space Connection

Once a space has been defined in a metaspace, it can then be joined by applications and will be automatically joined by agents if the space is distributed. Processes join a space either as a seeder or a leech; this is determined by the Distribution Role assigned to the Space Connection.

A single space can have multiple space connections, each with a different distribution role.

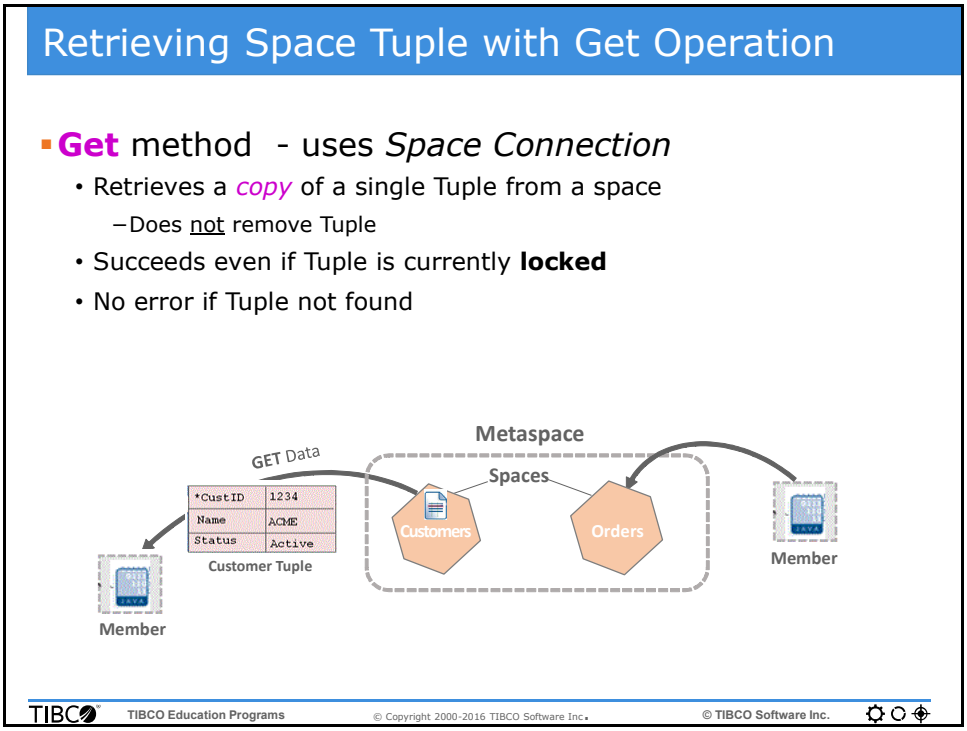


Creating a Space Tuple

To communicate with an ActiveSpaces metaspace, you must first define a space connection, which is part of a metaspace definition.

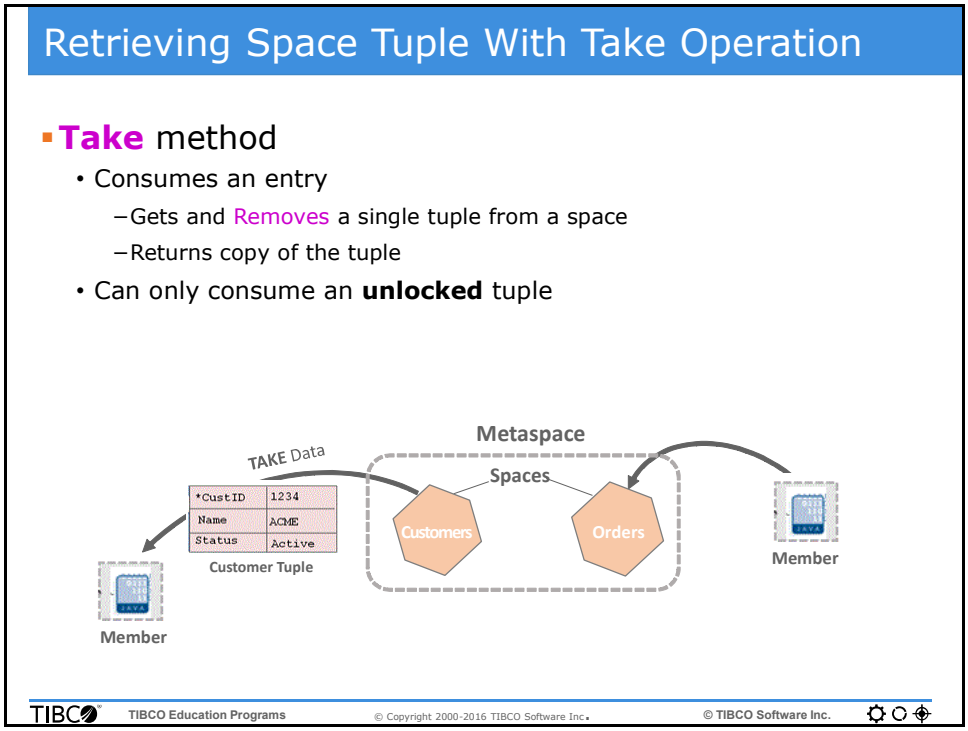
Metaspace is a BusinessWorks shared resource representing a metaspace. Add a Metaspace shared resource, then configure its properties.

You put data into a space by creating a data structure called a Tuple and then putting the tuple into the space. A tuple is a unit of data in a space defined by a set of typed fields (e.g. the Space schema), similar to a row of data in a database table. When a tuple is stored into a space, the fields that it contains must match the names and types of the fields described in the space definition. If there is a type mismatch between a field contained in the tuple and the type of the field defined in the space field definition, ActiveSpaces may perform an automated field conversion if possible, or the operation will fail.



Retrieving Space Tuple with Get Operation

To retrieve a single tuple from a space, a Get operation is performed. A Get makes a *copy* of a single tuple and retrieves the copy. It leaves the tuple in the space intact. Get operations can be performed on tuples even if they are locked by another operation.



Retrieving Space Tuple with Take Operation

To remove a single tuple from a space, use the Take operation. A Take returns the complete tuple to the user and at the same time *removes* the tuple from the space. The Take operation can only consume (remove) a tuple that is unlocked; a tuple can only be taken by one member, no matter how many try at the same time.

Get and Take operations are used when retrieving a *single* tuple using an exact key match. Later in the course, we discuss how to use an Entry Browser to retrieve multiple tuples matching a certain filter pattern.

Command Line Administration

- Administration tool
  - Uses java installation
  - Invoke from a command line:

    $AS_HOME/lib/**java -jar as-admin.jar**

  - Functions
    - Connect to metaspace
    - Create new metaspace
    - Add/remove/update spaces
    - View spaces

```
as-admin> connect
Connected to metaspace ms
as-admin> show spaces
Space names:
        CUSTOMER
        ORDERS
```

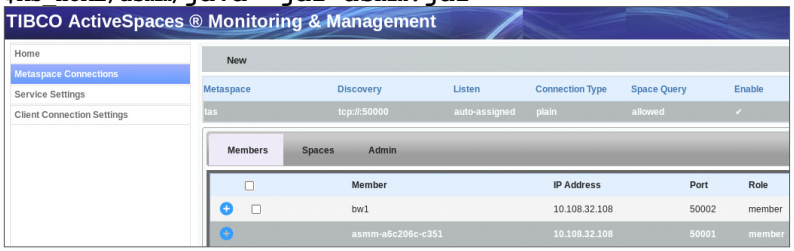
TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Command-Line Administration

as-admin - a command line utility used to connect to a metaspace, create or remove a space, and display information about existing spaces and members. The as-admin utility is invoked from a command line.

GUI Based Administration

- ActiveSpaces Monitoring and Management (**ASMM**)
  - Web based GUI, connects to *multiple* metaspaces
    - View members, space definitions,
    - View Schemas, entry statistics
    - Browse space entries
- From a terminal window:
  - `$AS_HOME/asmm/java -jar asmm.jar`



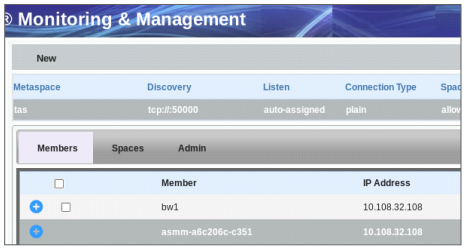
© Copyright 2000-2016 TIBCO Software Inc.
© TIBCO Software Inc.

GUI Based Administration

ActiveSpaces Monitoring and Management (**ASMM**) - an administrative GUI that lets you connect to a metaspace and display read-only information about spaces and other system resources. The ASMM and as-admin utilities do not join any spaces; they only store space definitions.

Viewing Metaspace and Spaces

- Metaspace information
  - Connected members
  - Spaces created
- Space information
  - Joined members, roles, space operations
  - Space definition and schemas



Members > Space Seeder Operation Statistics

Seeder	Puts	Gets	Takes	Queries	Expires	Invokes
agent1	0	0	0	0	0	0
bw-sharednothing	30	0	0	0	0	0

TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Viewing Metaspace and Spaces

The Metaspace Navigator lists all spaces that have been created for a metaspace. Selecting the metaspace name in the panel displays members (processes) that are currently connected to the metaspace, and any data they have put (added) into spaces.

Selecting a space in the Metaspace Navigator displays information about that space, including members that have joined the space and a summary of operations they have performed on the space, and read-only views of the space definition, schema (field definitions) and indices (if created).

Graphs provide information about how data is distributed between members in a space, and activity (puts, gets, takes, etc.) over the life of the space.

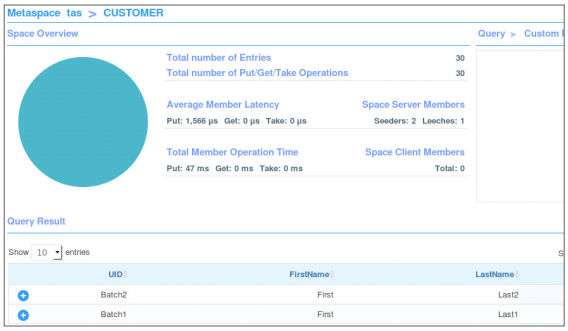
Viewing Space and Member Data

- Space > Members tab
  - View space **statistics**
    - Entries, Puts, Takes, Gets

Members > Details Info

Member	Host	Port	Process
agent1	10.108.32.108	50002	as-agent
bw-sharednothing	10.108.32.108	50000	as-java

- Space Browser
  - View individual data entries and **values**



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Viewing Space and Member Data

The Members tab displays real-time statistics of operations (puts, gets, takes, etc.) performed on the space by each member process.

Use the Space Browser to query a space and view the resulting data entries and their values. The Space Browser is accessed from the Utilities menu. You can apply filters to effectively “select” a subset of entries in a space (for example, Status LIKE "New"). One or more fields defined in the space schema can be used in the filter. If no filter is defined, all entries in the space are returned.

Review

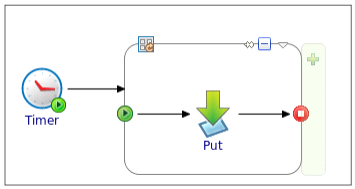
- Metaspace resource defines connection
- Spaces contain Space Connections
  - Type: Seeder or Leech
  - Used by activities
- Put Operation: Storing data as tuple in a spaces
- Get takes a copy of data; Take removes data

TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Review

Lab Overview

- Build processes to add & retrieve data
  - Add metaspace, space and space connection
  - Add activities that use space connection
- Use Tester to run processes
- Verify data operations in ASMM tool



TIBCO TIBCO Education Programs © Copyright 2000-2016 TIBCO Software Inc. © TIBCO Software Inc.

Lab Overview

Notes

Notes

Lab R Perform Data Operations

Overview

In this lab you will design simple BusinessWorks processes to add and retrieve data from a space.

Objectives

- *Add Data,* page 570
- *Retrieve Data,* page 581

Prerequisites

- The following TIBCO Software is installed and configured on your student machine:
  - BusinessWorks 6.3.x
  - ActiveSpaces 2.2.x
  - BusinessWorks Plug-in for ActiveSpaces 6.3.0
- Student lab files extracted
- Completion of previous labs

Directions

Complete the exercises that follow.

Lab R - Exercise 1: Add Data

Overview

In this exercise you create metaspace and space definitions directly in the TIBCO ActiveSpaces BW plug-in. Next, build a simple BW process to load data into a space. This allows you to test your space definition, and also reviews the steps to create a basic BusinessWorks process.

Steps

1. Launch TIBCO Business Studio and open your workspace.

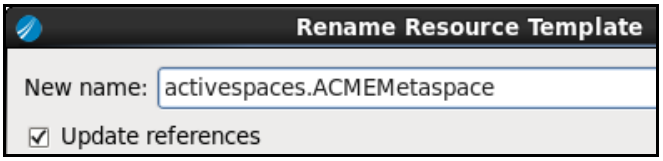
- Open a Terminal window, set the Environment and launch TIBCO BusinesStudio

```
source BW6Env
cd /opt/tibco/soa/bw6/studio/4.0/eclipse
./TIBCOBusinessStudio
```

- Open Workspace: **`/home/tibco/bw6edu/workspace`**

2. Update the Metaspace shared resource to your project.

- Update the **Metaspace** resource
  - Name: **`ACMEMetaspace`** (click on light bulb to rename)



  - Metaspace Name: **`ACME`**
  - Member Name: **`ACME`**
  - Discovery URL: **`tcp://:50000`**
  - **Save** and **Test Connection**

Tip: You changed the discovery protocol to TCP (from the default tibpgm).

- Click **Introspect** button, notice that there are **no** spaces defined for this metaspace

3. Define a Space shared resource.

- In *ACMEMetaspace* window, click **Space and Space Connection** tab
- Click **Add Space**, change the name to **`CUSTOMER`**
  - Click the **+** button to add the following fields to the CUSTOMER space definition:

Configuration | Key Definition | Affinity Definition | Advanced | Index Definition

Configuration

Name: CUSTOMER

Field Definition

Field Name	Field Type	Allow Field to be Null	Encrypt Non-Key Data Field
OrderID	STRING	false	false
CustomerID	STRING	false	false
Status	STRING	false	false
OrderAmount	DOUBLE	false	false
HandlingAmount	DOUBLE	false	false
OrderDate	DATETIME	false	false

- In *Key Definition* tab> **Key Field Names**, **Browse** and select **(checkmark)** : `OrderID`

Configuration | Key Definition | Affinity Definition | Advanced | Index Definition

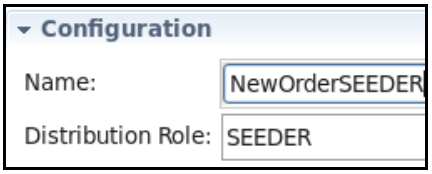
Key Definition

Key Index Type	Key Field Names
HASH	OrderID

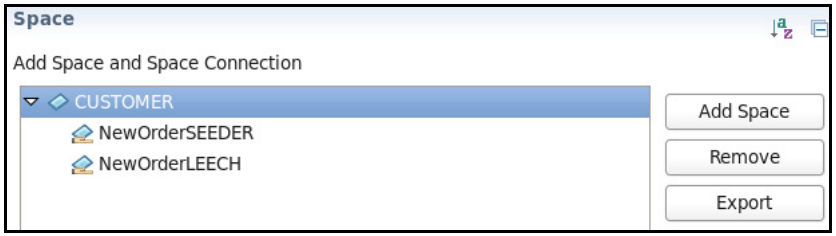
- **Save** your workspace

4. Add Seeder and Leech Space Connections for the NewOrder space.

- Click **Add Space Con...** and a space connection is added to the NewOrder space
  - Name: **`NewOrderSEEDER`**
  - Keep the default **SEEDER** Distribution Role



- Similarly, add another connection
- Select CUSTOMER and Click **Add Space Con...**
  - Name: **`NewOrderLEECH`**
  - Distribution Role: **LEECH**



- **Save** your workspace

Tip: Seeder: Used to populate a space in a metaspace. Leech : Consumes data from a metaspace. We will use the Leech in the next lab

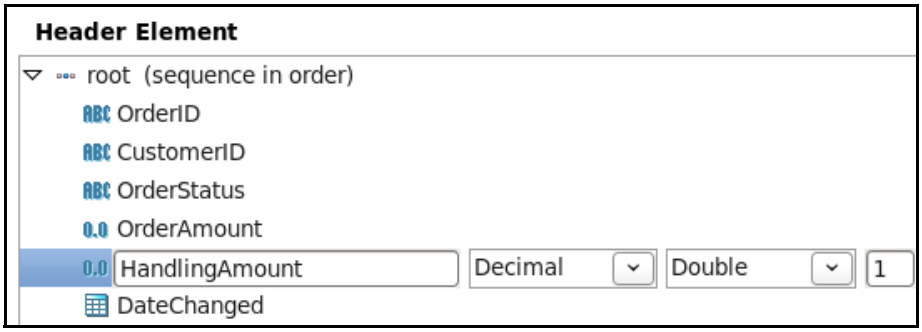
5. Connect to the tas metaspace in ASMM.

- Start ASMM UI
  - Open a Terminal window and run **`source BW6Env`**
  - Navigate to `$AS_HOME/asmm`
  - Run the command: **`java -jar asmm.jar`**
    - Wait for the message "*ASMM Admin Server started: http://localhost:8680/*
- Open Firefox and go to **`http://localhost:8680`**
  - Login with *admin/admin*
- Select **Metaspace Connections**
- Click on **New**
  - Metaspace Name: **`ACME`**
  - Discovery: **`tcp://:50000`**
  - Click **Save**
- Notice that there are no spaces defined yet for this metaspace. **Why?**

Analysis: Although the resource is connected to the metaspace, it does not join any spaces until you run a process that performs a space operation.

6. Define a new resource of type data format to parse incoming file records.

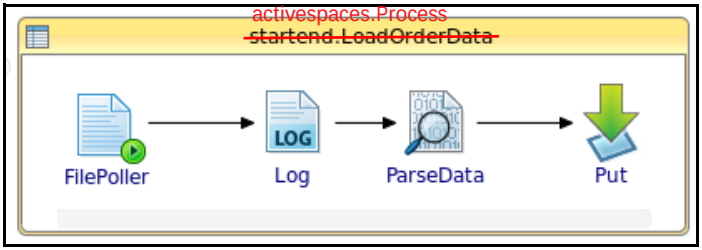
- Back in Business Studio, in *ActiveSpaces* > *Resources*
  - Right click on **ACMEMetaspace** and choose *New* > ***Data Format***
  - Name: **`ACMEOrderDataFormat`**
  - Click **Finish**
- Data Format Configuration: (keep defaults)
  - Format Type: Delimiter Separated
  - Col Separator: , (comma)
  - Col Separator Parse Rule: **Treat all characters as entered as a single column separator string**
  - Line Separator: **New Line**
  - Fill Character: **Space**
- Data Format Editor: Click "Select Input Header Type Definition icon" and click "Create New Schema"
  - Add these fields under root (Using Add Primitive Element button)
    - **`OrderID`**: String
    - **`CustomerID`** : String
    - **OrderStatus**: String
    - **`OrderAmount`**: Decimal/Double
    - **`HandlingAmount`**: Decimal/Double
    - **`DateChanged`**: Date & Time/ Date



  - Verify fields appear as shown above
  - Select the **root** element and click **OK**
  - **Save** changes

7. Open the empty process defined for the Application Module.

- In Project Explorer, browse to *activespaces* > ***Process***
- Add process activities as shown below:



8. Configure the file poller activity to read NewACMEOrders.txt.

- For **File Poller** activity*: Properties* > ***General***
  - File Name: **`/tmp/NewACMEOrders.txt`**
  - Polling Interval: 5 seconds
  - Include Existing Files: **checked**
  - Content as: **`Text`**

9. Log file name in the log activity.

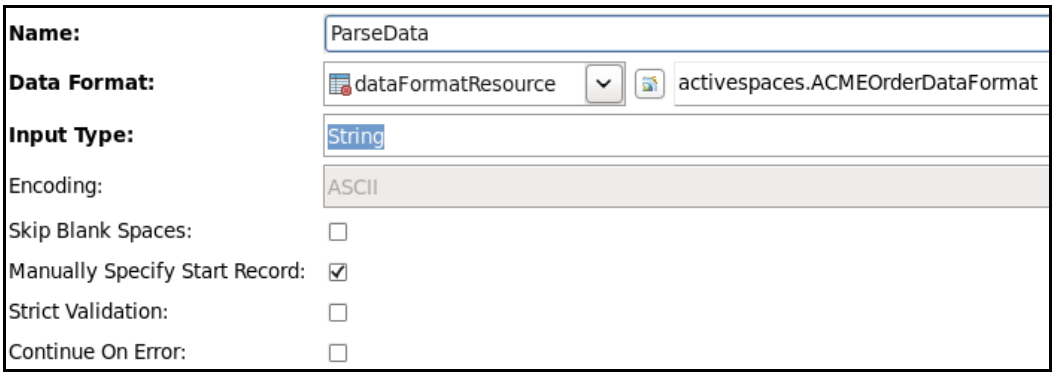
- For **Log** activity*: Properties* > ***Input***
  - Message: **`concat ("Reading File: ", $FilePoller/fileInfo/fileName)`**
- **Save** your workspace

10. Examine the contents of NewACMEOrders.txt file.

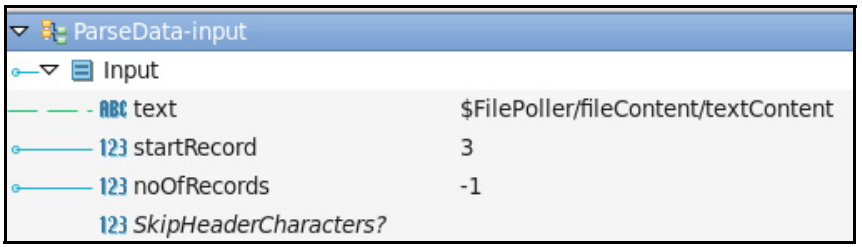
- Open the `/home/tibco/bw6edu/dev/Files/`**`NewACMEOrders.txt`** file using an editor like gedit
- Confirm that the first two lines in the files are used for the header
- Close the file

11. Update parse data activity to parse all records at once, after skipping file headers.

- For **ParseData** activity : *Properties* > ***General***
  - Data Format: **`activespaces.ACMEOrderDataFormat`** (use magnifying glass)
  - Input Type: **String**
  - Manually Specify Start Record: **Checked**



- Input Tab, complete as shown:



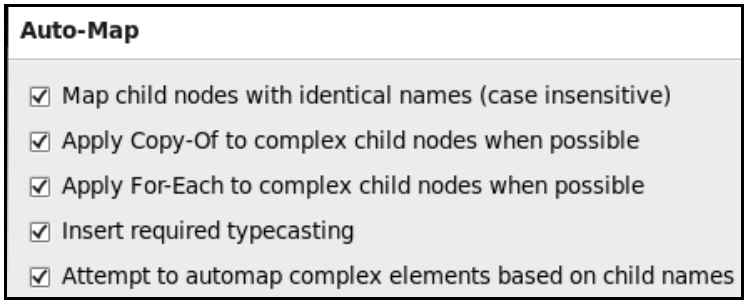
- **Save** all changes

12. Update Put activity to use the space connection define earlier.

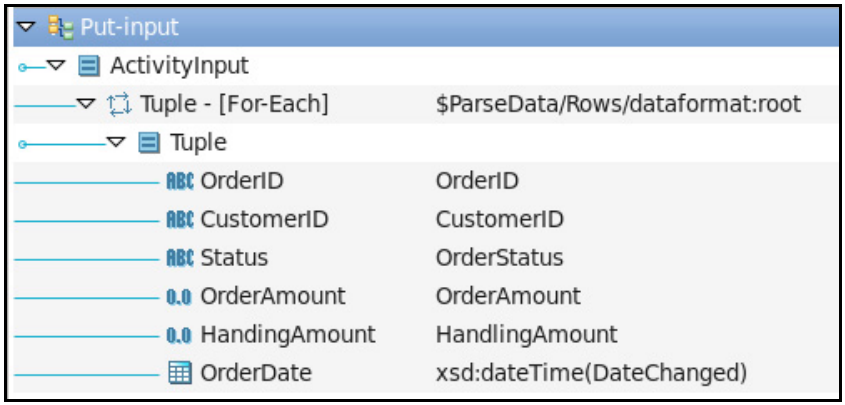
- In *Put* > *Properties* > ***General***, choose the SEEDER connection as shown

13. Map the output of parse data to input of the put activity.

- In Put activity > Properties > Input
  - Drag $ParseData > Rows > root* to *Put-Input* > *ActivityInput* > ***Tuple+***
  - A wizard pops up
  - Choose For each... and click **Next**
  - In Auto-Map screen, select all options



  - Click **Finish**
    - Map `OrderStatus --> Status`
    - Map `DateChanged --> OrderDate`, click *Mapper check and repair*
  - End result is as shown

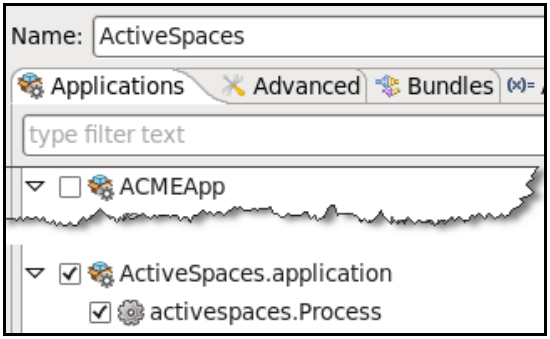


- **Save** all changes and confirm there are no errors (problems tab)

Testing

14. Run the PutData process using the BW Tester.

- In *Run > Debug Configurations,* define a new configuration to run ***ActiveSpaces.application***



  - Click **Debug**
- Copy `/home/tibco/bw6edu/dev/Files/`**`NewACMEOrders.txt`** to /tmp
- Observe console for processing details:

```
c.t.b.p.g.Log.ActiveSpaces.Log - Reading File: NewACMEOrders.txt
[2017-01-16T10:31:58.287][30145][140311702128384][INFO][transport]
ip_address=10.108.0.39 port=50000
[2017-01-16T10:31:58.288][30145][140311651899136][INFO][] discovery :50000
resolved to 10.108.0.39:50000
[2017-01-16T10:32:03.288][30145][140311651899136][INFO][ACME.$gmp]
-------------------------------
[2017-01-16T10:32:03.288][30145][140311651899136][INFO][ACME.$gmp] configuring new
manager
[2017-01-16T10:32:03.288][30145][140311651899136][INFO][ACME.$gmp] manager = ACME
(a6c0027-c350-587d119e-120, 10.108.0.39:50000)
[2017-01-16T10:32:03.288][30145][140311651899136][INFO][ACME.$gmp] self node =
ACME (a6c0027-c350-587d119e-120, 10.108.0.39:50000)
[2017-01-16T10:32:03.288][30145][140311651899136][INFO][ACME.$gmp] local node is
MANAGER now
[2017-01-16T10:32:03.288][30145][140311651899136][INFO][ACME.$gmp]
-------------------------------
[2017-01-16T10:32:03.302][30145][140311648741120][INFO][ACME.metaspace] using site
definition: type=[local] site=[default] routed=[false]
[2017-01-16T10:32:03.306][30145][140311605581568][INFO][ACME.$members] member
joined: ACME (a6c0027-c350-587d119e-120, 10.108.0.39:50000)
[2017-01-16T10:32:03.306][30145][140311702128384][INFO][ACME.metaspace] Connected
metaspace : name=[ACME] listen=[tcp://10.108.0.39:50000] discovery=[tcp://:50000]
member_name=[ACME] version=[2.2.0.017]
```

- **Note:** asmm is already connected as a member

15. View tuples in the NewOrderSpace with ASMM.

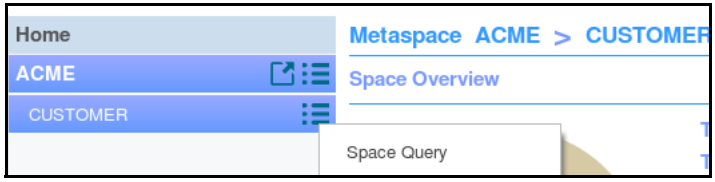
- Open a browser and go to `http://localhost:8680` **(admin/admin)**
  - Click **Start** for ASMM Monitoring Service
  - Click on the Monitoring URL link **`http://localhost:8686`**
- Verify that the NewOrderSpace is listed (created by the PutData process) with **25** tuples

16. Investigate the NewOrderSpace.

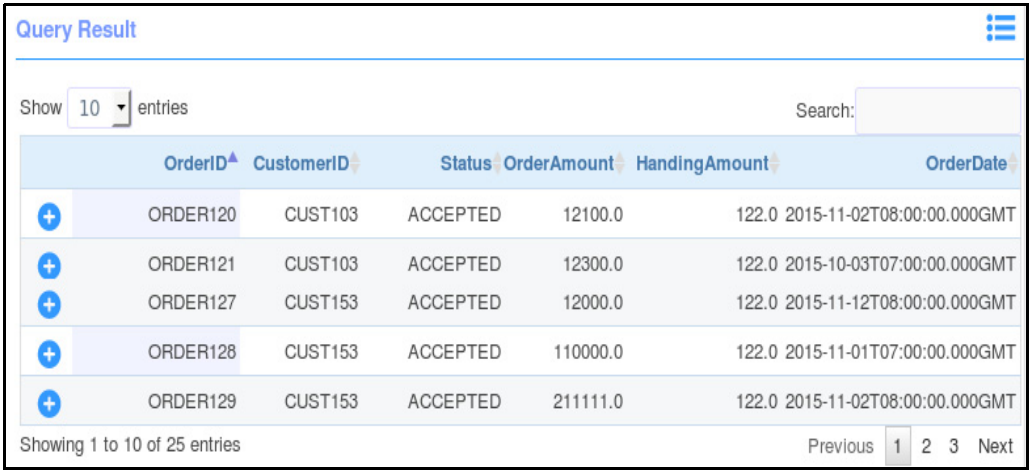
- Mouse over the tas metaspace CUSTOMER details and click on MONITOR
- The Metaspace tas Overview appears
- Observe that the top region of the screen cycles through the seven different detailed charts
- Observe the Advisories Activity Monitor chart

17. View the NewOrderSpace details.

- Select *CUSTOMER* > ***Space Query***



- The Query window appears (right corner), click the magnifying glass



Query Result

Show 10 entries Search:

	OrderID	CustomerID	Status	OrderAmount	HandingAmount	OrderDate
+	ORDER120	CUST103	ACCEPTED	12100.0	122.0	2015-11-02T08:00:00.000GMT
+	ORDER121	CUST103	ACCEPTED	12300.0	122.0	2015-10-03T07:00:00.000GMT
+	ORDER127	CUST153	ACCEPTED	12000.0	122.0	2015-11-12T08:00:00.000GMT
+	ORDER128	CUST153	ACCEPTED	110000.0	122.0	2015-11-01T07:00:00.000GMT
+	ORDER129	CUST153	ACCEPTED	211111.0	122.0	2015-11-02T08:00:00.000GMT

Showing 1 to 10 of 25 entries Previous 1 2 3 Next

- When you are finished testing, stop the BWApplication running in TIBCO Business Studio
  - Close the browser windows
  - Enter <**`Ctrl+C`**> in the Terminal running ASMM

Lab R - Exercise 2: Retrieve Data

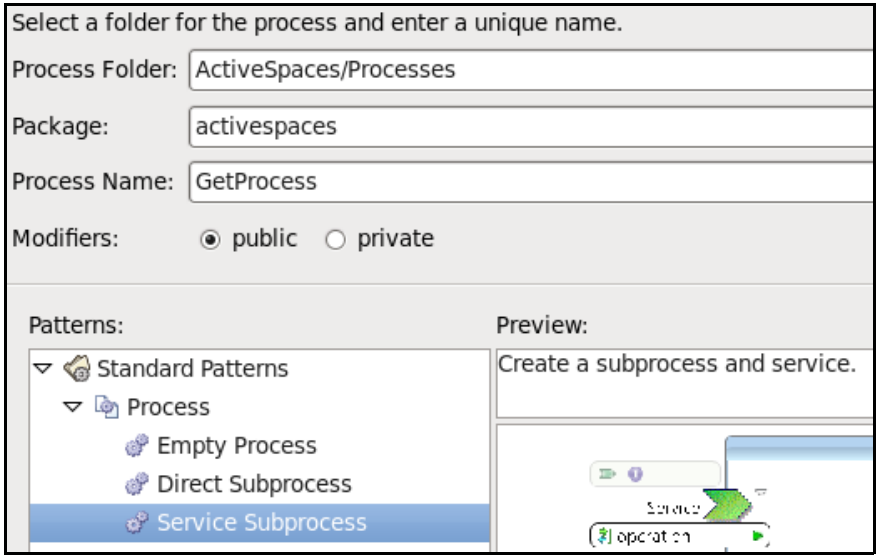
Overview

Configure a BusinessWorks process to perform single GET operations from the CUSTOMER space.

Steps

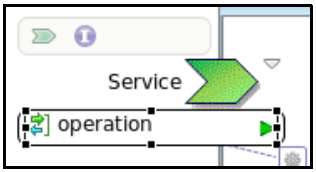
1. In TIBCO Business Studio create GetProcess to GET data from a space.

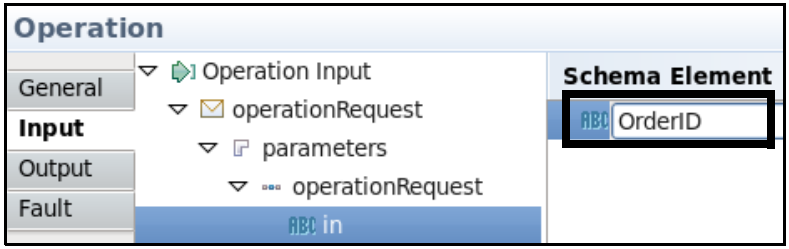
- In Project Explorer, right-click on *ActiveSpaces* > *Processes* > ***activespaces***
- Select New > BusinessWorks Process
  - Process Name: **GetProcess**
  - Patterns: **Service Subprocess**



- Click **Finish**
- Navigate to the *GetProcess-Operation* > *Properties* > ***Input***

- Double-click on **operationRequest > in** and rename it to **`OrderID`**

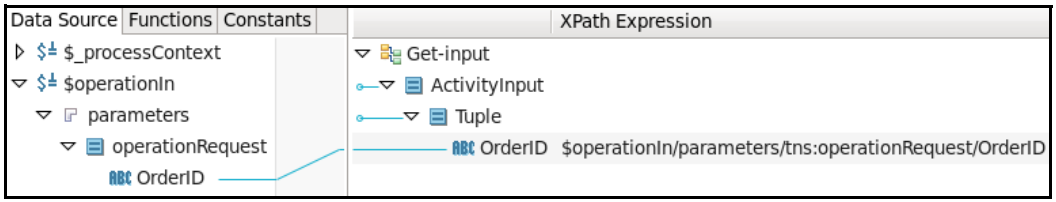




- Select the Operation *Output* tab
- Double-click on **operationResponse > out** and rename it to **`OutString`**
- **Save** changes

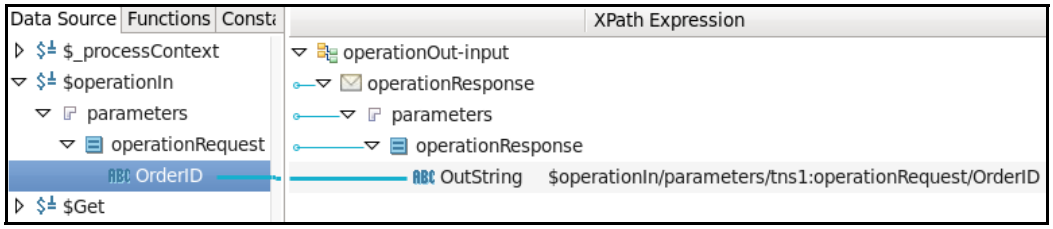
2. Add a GET activity to the process.

- From *ActiveSpaces* palette, select a **Get** activity
  - Drop it on the connection between **Start** and **End**
  - Connection should be created automatically
- Select the **Get** activity and configure
  - Name: **`Get`**
- In the *General* tab set the Space Connection to **ACMEMetaspace/CUSTOMER/NewOrderLEECH**
- Select the *Input* tab and map **operationRequest/OrderID**

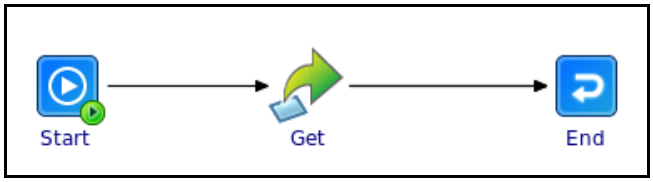


3. Configure the process End activity.

- Update Input so that it maps to the new value (OrderID)



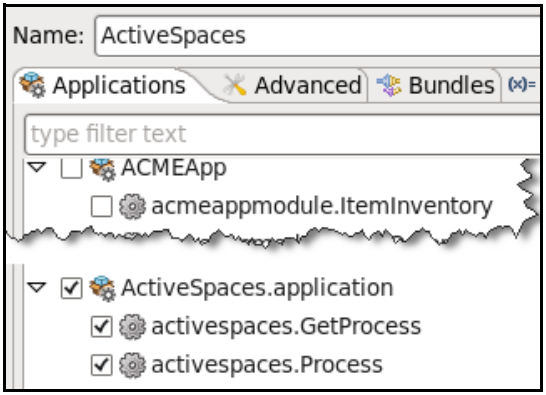
- **Save** your workspace



Testing

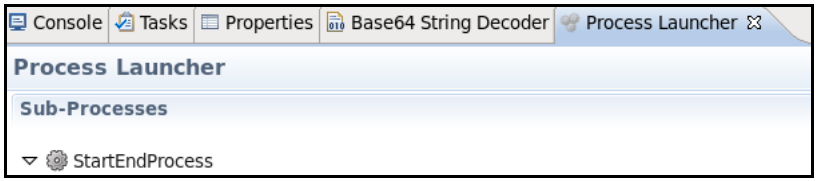
4. Run the GetData process in the Tester.

- Run the Process and GetProcess applications in ***Debug Mode***



- Select the *BusinessWorks Jobs* tab

- The applications startup
  - Observe the message "Reading File: NewACMEOrders.txt"
- Select *Process Launcher* tab and select the refresh icon refresh icon



- Expand and select *activespaces.GetProcess* > *operation* > ***operationInput***
- Enter a OrderID of ORDER120 and Invoke a job
  - Click on the green **Invoke** button
  - Observe the results returned by the GET activity (job data)

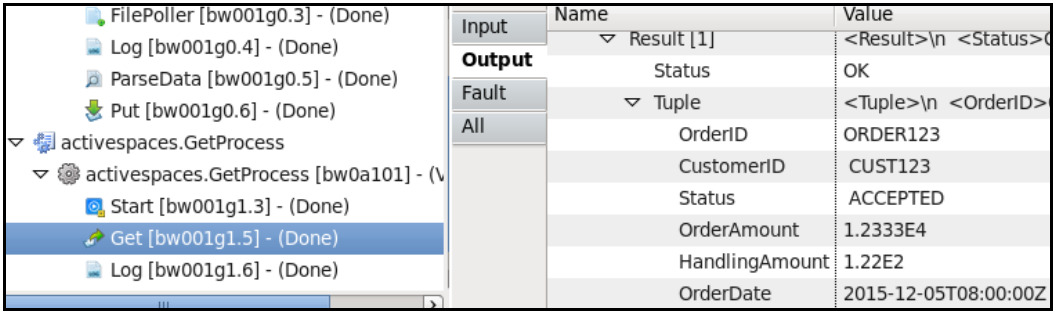
Tip: The debugger runs both the PUT and GET Process. The PUT process runs successfully, after reading the records from the file and doing a batch PUT operation.

Since the debugger continues running, the metaspace and space are also available and managed by the debugger.

The sub-process for GET is invoked when the debugger is running. It uses the same metaspace connection and fetches data from the space.

5. View the Get operation job data.

- In the Debug panel select the Get activity
- Select the Job Data pane on the right side of your screen
- Select Input and Tuple to view the incoming Job Data
- Select the Output tab and view the results



6. Submit another job and view results.

- In the Process Launcher tab, click Refresh
- Submit a different CustID
  - CustID: **`ORDER500`**
  - You should see no results
- When you are finished testing:
  - Stop the BWApplication running in TIBCO BusinessStudio

Notes

Appendix A Common Operating System Tasks

Overview

We recognize that while some who attend our training courses are quite familiar with the Linux operating system and how to accomplish various configurations and steps, others may be less familiar. Please use this appendix as a reference as needed.

Sections

- *Starting Applications,* page 588
- *Viewing Status of Processes,* page 589
- *Terminating a Process,* page 589
- *Obtain IP address of your machine,* page 590
- *Basic commands from the Terminal Window,* page 590
- *Using gedit to open/modify files,* page 591
- *Copy/paste/rename files in File Browser,* page 591
- *Create and use File Browser bookmarks,* page 592
- *Naming a Terminal Window,* page 592
- *Managing Environment Variables,* page 594
- *Working with Java Code,* page 596
- *Setting Up Email,* page 597
- *Screen Resolution,* page 598
- *Running Virtual Machines in VMware Player,* page 599

Starting Applications

Many TIBCO and other applications can be started from the Applications menu.

The *Applications* > ***TIBCO*** menu has many shortcuts to run the scripts that are in the scripts folder on the Desktop.

If there is no menu item, try the approach described below.

- **To start a script (*.sh) file**:
  - Open a Terminal and navigate to the directory that contains the .sh file
  - Type in **`./<program>.sh`** and click the <Enter> key
    - Example: to start TIBCO EMS server:
      - `cd /opt/tibco/soa/ems/8.0/bin`
      - `./tibemsd64.sh`
- **To start an executable**:
  - In File Browser navigate to the directory that contains the file
  - Double-click the executable
    - Example: to start TIBCO Designer:
      - In File Browser, go to /opt/tibco/soa/designer/5.8/bin
      - Double-click on designer
- **To start a program**:
  - Open a Terminal and navigate to the directory that contains the file
  - Type in the file name and click the <Enter> key
    - Example: to start Oracle SQL Plus:
      - `cd /opt/oracle/app/oracle/product/11.2.0/client_1/bin`
      - `sqlplus`

Viewing Status of Processes

Linux considers anything running on the computer to be a "process".

There is a script built into the image which will find *most* of the TIBCO processes you may be interested in. Use this menu path to launch it:

Applications > *TIBCO* > ***Show Running***

However, for some processes you may need to manual search.

- Example of how to view a list of EMS processes that are running.

  `pgrep -l ems`
- Example of how to view a **detailed** list of EMS processes that are running.

  `ps -ef | grep ems`

A useful utility is **top**. This utility (launched by typing **`top`** into a terminal) shows all running processes.

- Launching as **`top -p <PID>`** will show details about only the one process
- Type P to sort to CPU utilization.
- Type M to sort by RAM utilization
- Type **`q`** to exit top.

Terminating a Process

To terminate a process, you will want to know the process id (PID). The PID can be discovered using one of the approaches described in the *Viewing status of processes* section above.

The following command allows you to terminate a process.

`kill -9 <pid>`

If using the top utility:

- Type **`k`** and then enter the PID to terminate the process

Obtain IP address of your machine

- Launch a Terminal window and enter the following command:

  `/sbin/ifconfig`

  TIP: look for the eth0 inet addr details.

- *If your machine is running in the cloud (e.g. on Amazon EC2)*, then the `/sbin/ifconfig` command will return the "internal" ip address. To discover the "external" ip address, type in this line:

  `curl ifconfig.me`

Basic commands from the Terminal Window

`pwd`	Tells your current location (for example **`/home/tibco)`**
`cd`	Allows you to change directory (**`cd /home/tibco/BWEDU`**)
`cd ..`	Allows you to move back one directory
`clear`	Allows you to clear the terminal window
`ls -l`	Returns a detailed list of your current directory
`env`	Displays your working environment variables and their values
`echo $CLASSPATH`	Returns your CLASSPATH entries
`echo $LD_LIBRARY_PATH`	Returns LD_LIBRARY_PATH entries
<**Up**> and <**Down**> arrow keys	Allow you to access command history (commands you have already entered in a session)

Using gedit to open/modify files

Open file in gedit from the command line

- Launch a terminal window
- *Example*: enter the following command to edit the hosts file in the /etc directory

  `gedit /etc/hosts`

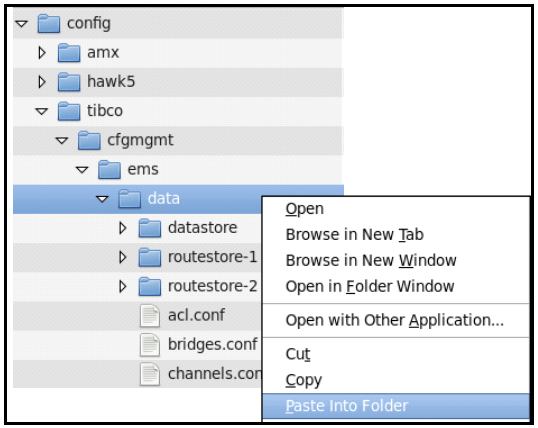
Open a file in gedit from the File Browser

- Launch File Browser on the top menu bar (file cabinet icon)
- Browse to *File System* > *etc*
- Right-click on hosts and select Open with gedit

  Note: If **Open with gedit** is not available from the drop-down menu, select **Open with Other Application** and select **gedit** from the list

Copy/paste/rename files in File Browser

- Copy/paste/rename example
  - Launch File Browser on the top menu bar (file cabinet icon)
  - In File Browser navigate to **`/opt/tibco/config/tibco/cfgmgmt/ems/data`**
  - Right-click on the **tibemsd.conf** and select **Copy**
  - Right-click on the `/opt/tibco/config/tibco/cfgmgmt/ems/`**`data`** directory and select **Paste Into Folder**



  - Right-click on **tibemsd (copy).conf** and **Rename** the file to **`tibemsd-dom.conf`**

Create and use File Browser bookmarks

- File Browser allows you to bookmark frequently visited locations
  - To create a bookmark, simply navigate to the folder you want to bookmark
  - Verify that the folder location you want to bookmark appears in the location bar
    Location: **/home/tibco/BWEDU**
  - Choose *Bookmarks > Add Bookmark*
  - The bookmarks are displayed in the **Places** view of the **Side Pane**.
  - To navigate to the bookmarked directory, double-click the bookmark
- There are also a number of TIBCO-specific bookmarks created for you:
  - `tibco` (user's directory where you can add your own files and folders)
  - `/opt/tibco` (where TIBCO software applications are installed)
  - `Oracle/client_1` (Oracle application directory)
  - `EDUSVR Share` (link to your instructor's Share folder)

Naming a Terminal Window

1. Launch a Terminal window
2. Select *Terminal > Set Title....*
3. Enter a Title (e.g. "**EMS Server**") and click **OK**

Downloading and Extracting Student Lab Files

- For some courses it may be necessary to copy student lab files (in the form of a .zip file) from the instructor's machine (EDUSVR) and extract them to a course folder on your local machine.

1. Add your instructor's EDUSVR machine address to your **hosts** file
   - In the hosts file, uncomment the line: `# edusvr edusvr.tibco.com`
   - Add the EDUSVR ip address (provided by your instructor) to the line
     - *For example*:

       `10.107.132.44 edusvr edusvr.tibco.com`
   - Save the changes
2. Connect to the EDUSVR machine
   - Open a File Browser and under **Places**, select the **EDUSVR Share** bookmark
     - Password: **`TIBCO7222`**
   - You are placed in the instructor's *Share* directory
3. Copy the `<CourseNumber>.zip` file to the /home/tibco directory on your machine
   - Right-click on the zip file and select **Copy**
   - In the **Places** section on the left, select the **tibco** bookmark
   - Right-click anywhere in the view pane on the right and select **Paste**
4. Extract student lab files to your `/home/tibco` directory
   - Right-click on the zip file and select **Extract Here**
   - A directory will be created on your local machine's `/home/tibco` directory corresponding to your course ID. Within this directory are subfolders that contain the files you will need during the course lab exercises.
5. Create a bookmark for the student lab files directory
   - In File Browser, navigate to the `/home/tibco/<course ID>`
   - Verify that the folder location you want to bookmark appears in the location bar

     Example: **/home/tibco/BWEDU**
   - Choose *Bookmarks* > *Add Bookmark*
   - The bookmarks are displayed in the **Places** view of the **Side Pane**.
   - To navigate to the bookmarked directory, double-click the bookmark

Managing Environment Variables

1. View and set the environment variables.

- To ***view*** the initial value for all the environment variables set at the time your session starts, open a Terminal and issue the command `env`
- To filter this list to view only the variables related to TIBCO product installation set for you, issue the command

  `env | grep tib`
- To display the single environment variable by name, use the command `echo`
  - For example, to view the value of the CLASSPATH issue the following:

    `echo $CLASSPATH`
- To ***modify*** the PATH or CLASSPATH *for the current Terminal session*:
  - To add the directory **`new_directory`** to the PATH, issue the command `export`:

    `export PATH=$PATH:<new_directory>`
  - To update the environment variables of the current Terminal session by running a shell script containing several `export` commands, execute

    `source <script_name>`

    OR

    `. <script_name>`

Tip: Executing a script by running a "**`.`**" or a **`source`** command causes the commands from the file to be executed in the current shell context, without spawning a new, nested, shell

- To ***modify*** the PATH or CLASSPATH permanently for the tibco user
  - Edit `/home/tibco/`**`.bashrc`**
  - Add any new variables
  - Note that any variables already defined in the /etc/environment can be overridden for the tibco user by defining them in this .../.bashrc file
    - Example: to prepend a path to the CLASSPATH:
      - Edit /home/tibco/.bashrc
      - Uncommend the # export CLASSPATH=$CLASSPATH line

- Add the new library or jar reference in between the CLASSPATH= and the $CLASSPATH and use a colon at the end.
- To append to the $CLASSPATH instead, add the new reference after $CLASSPATH (starting with a colon).

Working with Java Code

1. To compile Java code:

- Verify the JDK is installed on your computer. To do this, open a Terminal and type **`java -version`**
- Verify `<JAVA_HOME>/bin:` is in your system PATH variable. To do this, in a Terminal issue the command

  `echo $PATH`
- If you do not see the reference to the JDK's bin directory, see *Managing Environment Variables,* page 594 for how to set it.
  - Alternatively, you may have been provided with a **`setup.sh`** file that may set this variable for your Terminal session. If so, run it by using **`source`** command in the current Terminal session that you will subsequently use to compile the code (see *Managing Environment Variables,* page 594)
- Open a Terminal in the directory where your java code (*.java) is and type in the following:

  `javac <code_file_name>.java`

  If no errors are presented, your code has compiled successfully.

Tip: To compile all the java files in a single directory type the following:

`javac *.java`

2. How to run Java code.

- Open the Terminal in the directory that contains your compiled code (the files that end with the .class extension) and type the following:

  `java <code_file_name>`

Setting Up Email

The email server is named **edusvr** and runs on the instructor server. You can send and receive email from your **EDUCLT** client machine using *Mozilla Thunderbird* email client.

1. Create an email account.

- Launch Mozilla Thunderbird
  - Select *Applications* > *Internet* > *Thunderbird Email*
  - At the "Welcome to Thunderbird" screen, select "Skip this and use my existing email."
- In the Mail Account Setup dialog configure your account
  - Your name: **`userXX`**
  - Email address: **`userXX@edusvr`**
  - Password: **`userXX`**
  - Click **Continue**

  Note: *Looking up configuration: Email provider* may take a couple of minutes
  - Click **Done** when **"Configuration found..."** is displayed
  - At the WARNING screen check "I understand the risks"
  - Click **Done**
- Select *userXX@edusvr* in the top left corner
- Select **View settings for this account**
  - Highlight **Outgoing Server (SMTP)**
  - Select **Edit** and set *Authentication Method* to **No authentication**
  - Click **OK** twice to exit the Account Settings windows

Screen Resolution

It is worth taking the time to set the correct screen resolution for your remote desktop session so you can avoid the need to constantly scroll up and down and side to side to see the entire Desktop. See the relevant section below for details.

- For RDP connections
  - From your Windows laptop, choose *Start* > *All Programs* > *Accessories* > ***Remote Desktop Connection*** (if using a Mac, use an RDP program such as the free Microsoft RDP app available in the App store for free)
  - Before connecting to the remote machine, click **Show Options** and in the *Display* tab use the slider to choose the desired display configuration.
  - Click **Connect** and login to the remote machine.

    Tip: If the screen resolution needs further adjustment, exit the remote session and repeat the steps above.

- For Desktop through the browser connections:
  - Point your HTML5-compliant browser to the remote desktop address (e.g. http://ipaddress)
  - Before connecting to the remote machine, click into the **Display tab** to choose the desired display configuration.
  - Click **Connect** and login to the remote machine.
  - If the screen resolution needs further adjustment, exit the remote session and repeat the steps above.

Running Virtual Machines in VMware Player

Launch the Player

- *Start > All Programs > VMware >* ***VMware Player***

Locating and Starting a Virtual Machine

- If the machine you want to run is *not* already listed in the left column under "":
  - From the "Welcome to VMware Player" panel choose **Open a Virtual Machine**
  - Browse to the location of machine, select the .vmx file and choose **Open**
- Select the machine (it should now appear under "") and choose **Play virtual machine**

Logging into a Virtual Machine

- Use the Ctrl+Alt+Insert key combination, or alternatively, from the VMware Player menu bar choose *Virtual Machine >* ***Send Ctrl+Alt+Del***
- Enter the *username* and *password* (admin/TIBCO7222)

Finding the IP Address of Your Virtual Machine

- Once logged into the virtual machine, open a command prompt window and issue command **`/sbin/ifconfig`**

  TIP: look for the eth0 network details

Shutting Down Your Virtual Machine

- From within the virtual machine, open a terminal window and type this command:

  `/sbin/shutdown -h now`
  - When prompted, enter the tibco user password
- When the virtual machine is done gracefully shutting down, VMware Player should automatically close.

Notes