Seamless Integrated Consistent Personalised

enactor

retail systems for a digital world

Enactor Training Course Toolkit Fundamentals

Toolkit Fundamentals

Agenda

- Application Overview
- Application Process Editor
- Page Definition Editor
- Expression Language
- Resource Library
- Process Connection Editor
- Application Deployment Overview

Application Overview

Each Enactor Application is structured into a number of domains:

- Views
  Manage the application lifecycle
- Processes
  Orchestrate the application logic
- Prompts
  Render the User Interface

Application Overview (cont.)

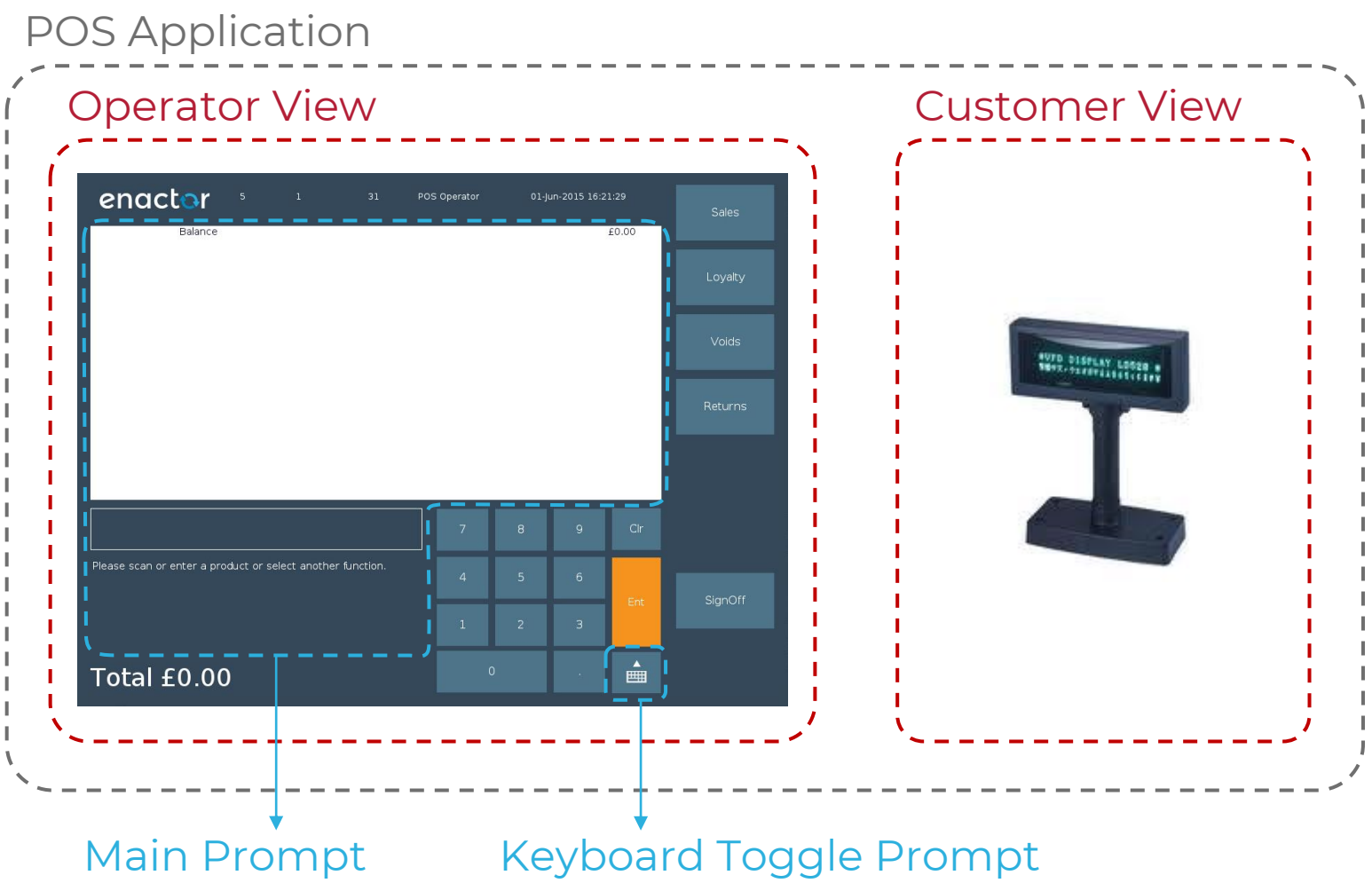
- Views
  - Usually a small number per application
  - Holds the global state for the application and/or ‘session’
  - May support a user interface layer if required

- Application Processes
  - Where the main logic of the application lives
  - Decide what screens to display, what to do with any data captured, etc.
  - Holds the private state for a discrete application operation

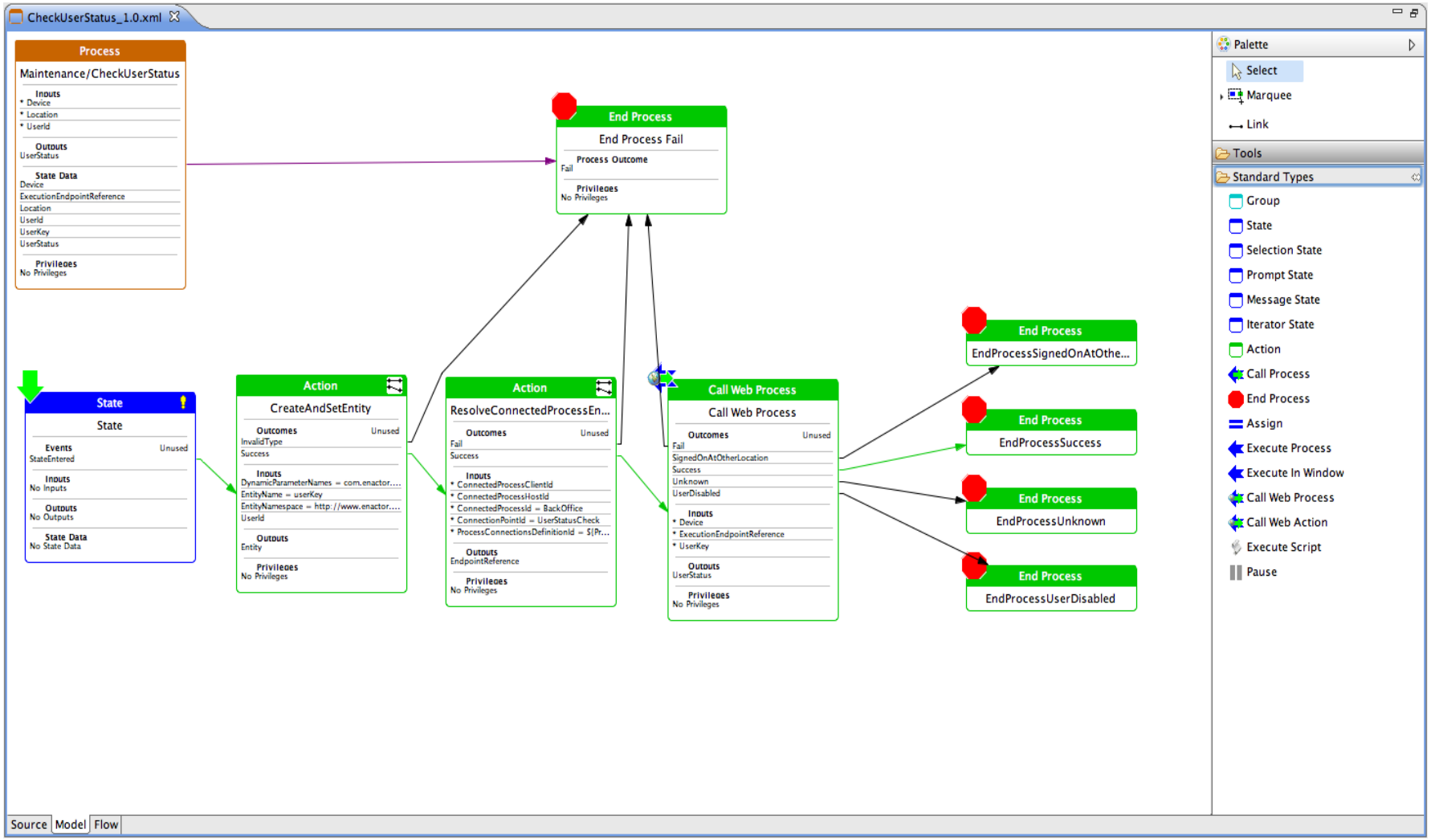
Application Overview (cont.)

- Prompts
  - Are used to control the interaction with the user interface
  - Typically are responsible for controlling the rendering process
    - Some prompts render to XML or JSON which is then sent ‘over-the-wire’ to a remote renderer
  - Raise events back into the Process as a result of user interaction
  - Multiple Prompts may render to the User Interface in parallel

Framework Overview



Application Process Editor



Application Process Editor (cont.)

- Visually represents the Application Process
- Allows you to review how data flows through the application
- Allows you to drill into Prompts and Java Actions
- Can debug deployed applications
  - Both ‘Tracing’ and Breakpoints are supported

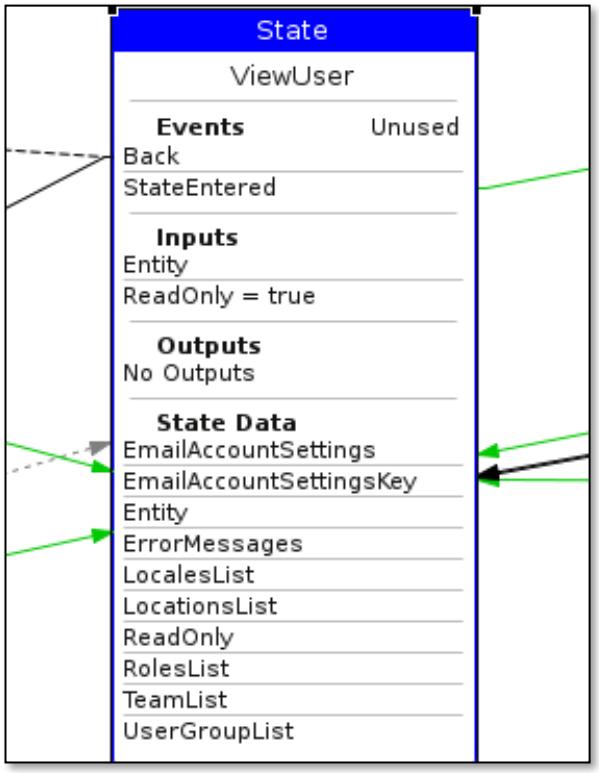
Application Process Editor (cont.)

- Application Processes are formed from States and Actions
- Processes control what happens in response to an Event from a State, or the Outcome of an Action
- Data can be stored at the Process level and is therefore available until the Process is ended
  - Data can be ‘returned’ from a Process to the calling process
  - The Actions UIGetViewData and UISetViewData can also be used to store data at the View level (i.e. globally)

Application Process Editor (cont.)

States

- Points at which a Process waits for an Event to be raised
- Can hold a subset of the data available to a Process
- May be associated with a Prompt (User Interface)
- Data can be transferred from State to State without requiring it to be additionally stored in a Process



Application Process Editor (cont.)

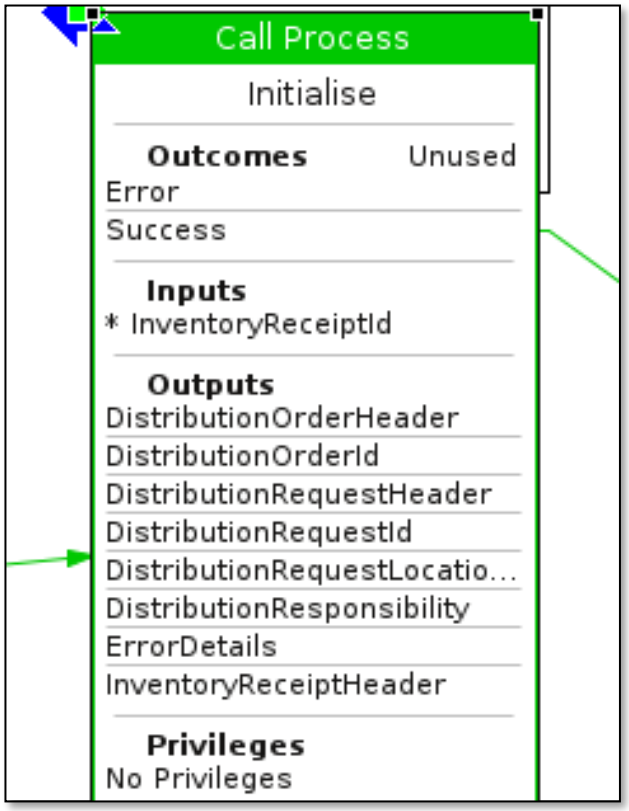
States

- Basic State
  - Used if no user interface, looping or extra functionality is required on the state
- Prompt State
  - Adds support for displaying a user interface while the state is waiting for an Event
- Message State
  - Extends Prompt State for conveniently displaying Messages
- Looping State
  - Provides a mechanism to loop within a Process
- Iterator State
  - Similar to Looping State, but provides a controlled loop over the standard Java Iterable interface.

Application Process Editor (cont.)

Actions

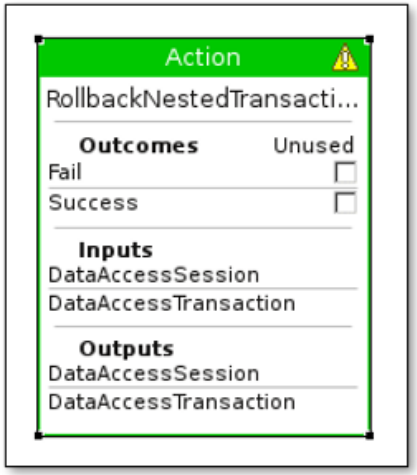
- Control structures that cause some application logic to be executed
- Accept ‘Input Data’ from a Process and/or State
- After execution return ‘Output Data’ and an ‘Outcome’
- The Process will use the Outcome to determine how the application should flow



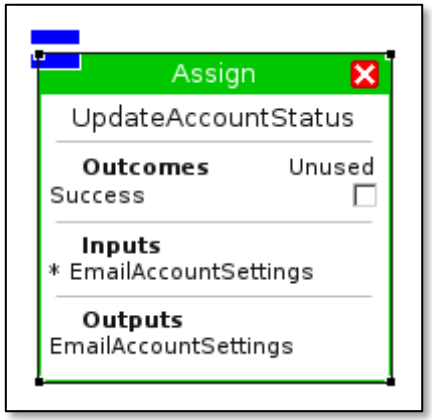
Application Process Editor (cont.)

Actions

- Basic Actions
  - Execute blocks of Java code



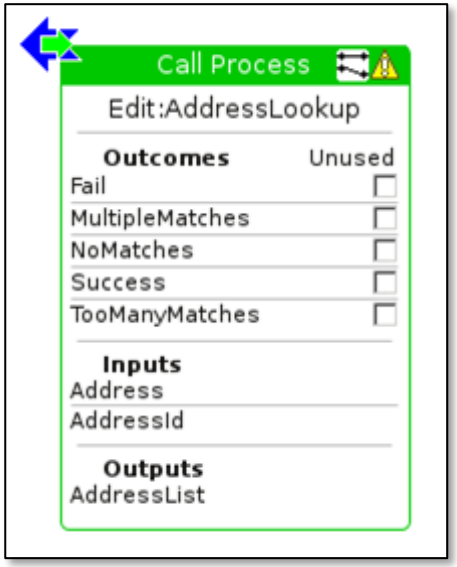
- Assign Action
  - Allow you to embed assignments in the Process
  - Avoids the need to write code
  - Can use complex expressions



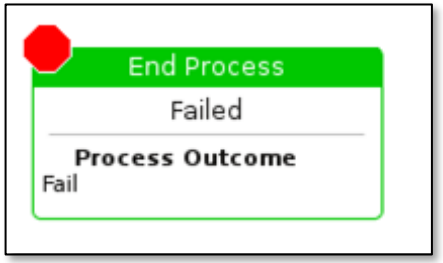
Application Process Editor (cont.)

Actions

- Call / Execute Process Actions
  - Allow you to split your processes up into reusable chunks
  - Also provides the ability to run a Process in its own Thread



- End Process Actions
  - Used to end the current process and return data to the "calling" process



Application Process Editor (cont.)

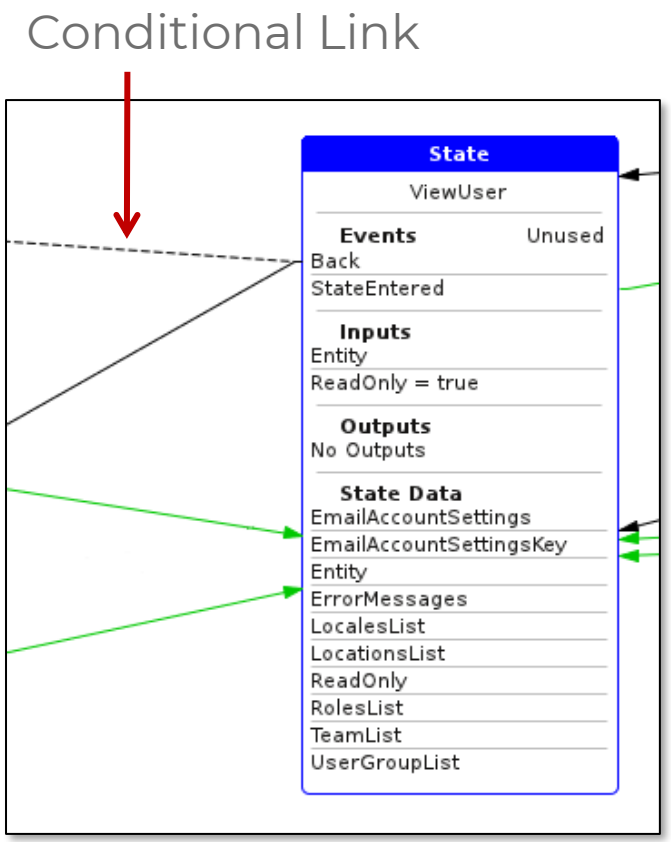
Thread Handling

- The Execute in Background and Execute in Window Actions allow a Process to execute in parallel with the calling Process
- The Outputs and Outcome of these actions indicate if the Process was successfully launched
- A ‘Process Handle’ is returned to the Calling Process
- View Events can be used to communicate between Processes
- Remember that Execute in Background must not attempt to present a user interface.

Application Process Editor (cont.)

Conditional Links

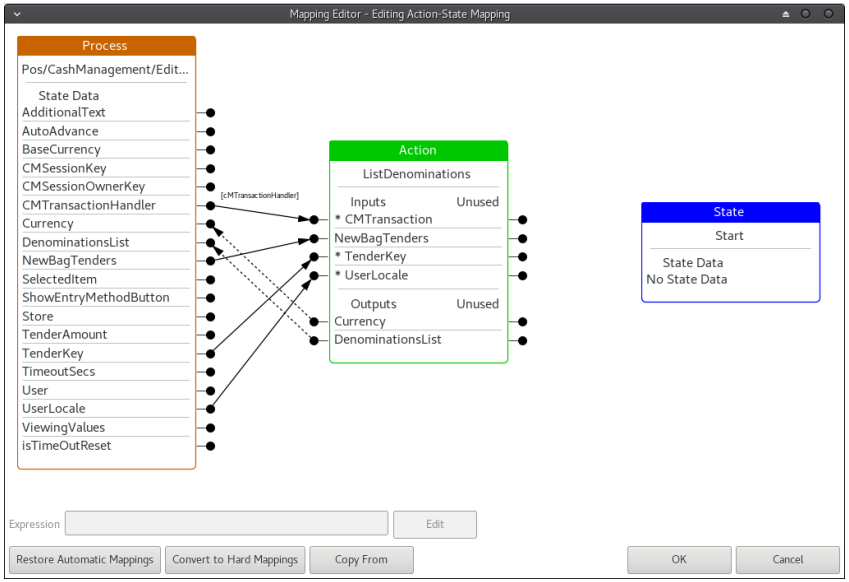
- Allows a Link to only be followed if some condition is met
- Many links can attach to the same Event/Outcome as long as there is only one that is not conditional
  - The order the conditional links are evaluated is not guaranteed, you must therefore ensure any conditions are complementary
  - If there is a link with no condition, it will only be followed if none of the conditional links would be selected.
- Conditions are specified using Expression Language (EL)
  - All data in the current State and Process is available



Application Process Editor (cont.)

Mapping Editor

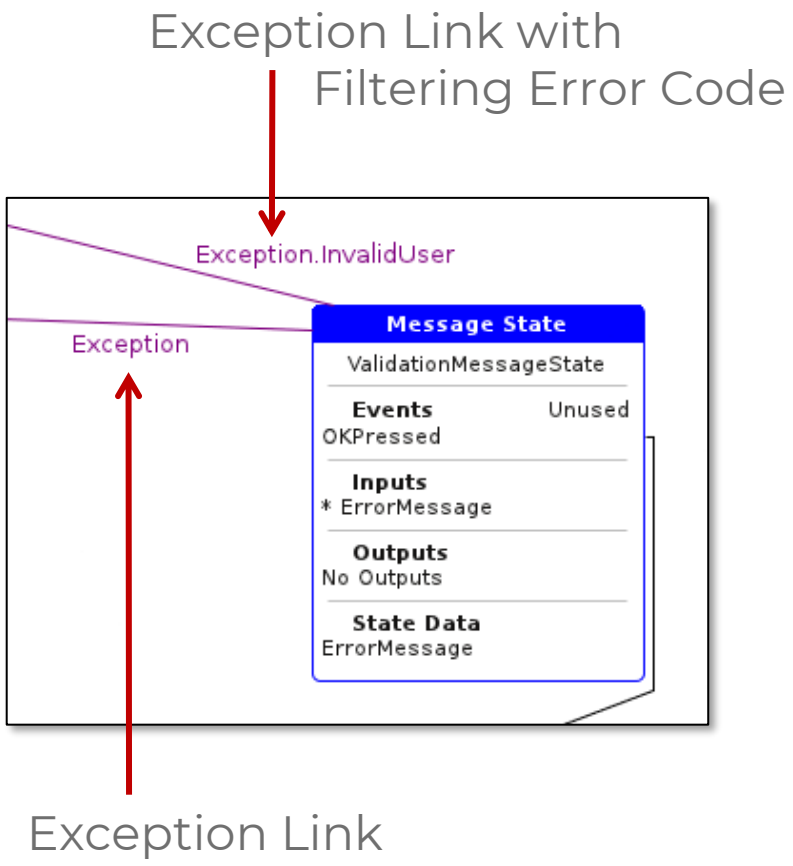
- Allows you to link un-related variables
- Can add expressions to map between part of a source variable
  - Expressions cannot be added to output mappings
- By default any similarly named variable is automatically mapped
  - As soon as you define one ‘Hard’ mapping, all ‘Automatic’ mappings are disabled



Application Process Editor (cont.)

Exception Handling

- Exceptions can be trapped in a Process using Exception Links
- Exceptions can be filtered by their 'ErrorCode'
- Exceptions can be trapped at Action, State or Process level
- If an Exception is not caught in the current Process, it is passed to the calling Process



Application Process Editor (cont.)

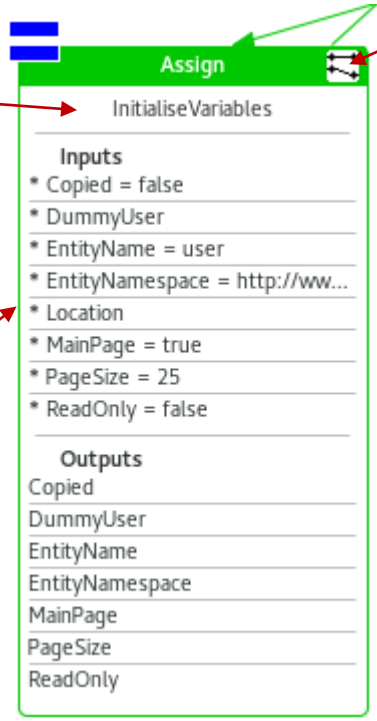
Tips and Tricks – Clickable Shortcuts

Most areas of the figures provide some useful short-cut, when they are double clicked:

Different operation depending on the Action

Double-Click to add/remove Inputs

Shift-Double-Click to add/remove Parameters



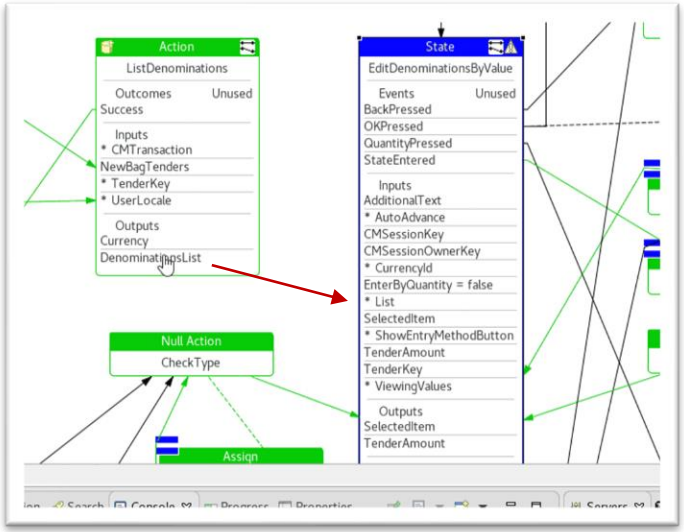
Goes to the Mapping Editor

Double-Click to add/remove Outputs

Application Process Editor (cont.)

Tips and Tricks – Dragging Variables

Drag variable declarations between sections, or between different figures:



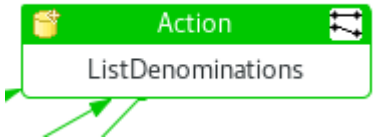
You can also drag variables in the Mapping Editor

Application Process Editor (cont.)

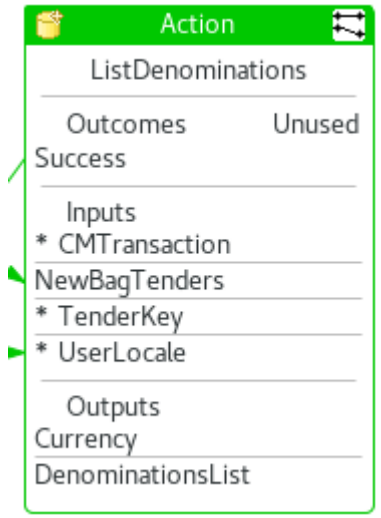
Tips and Tricks – Styling Figures

Using the Style context menu you can change the appearance of the figures:

- Keys Only – a very minimal appearance showing only the Action Type and Name. When in this form, incoming and outgoing links can connect to anywhere on the figure



- Defined Sections – in this form, any section of the figure which has some content is displayed. If Events or Outcomes are shown then any connected links appear next to the correct Event/Outcome



Application Process Editor (cont.)

Tips and Tricks – Synchronizing Actions

- Using the Synchronize context menu you can update the figure to match changes to the underlying Action (or vice-versa)

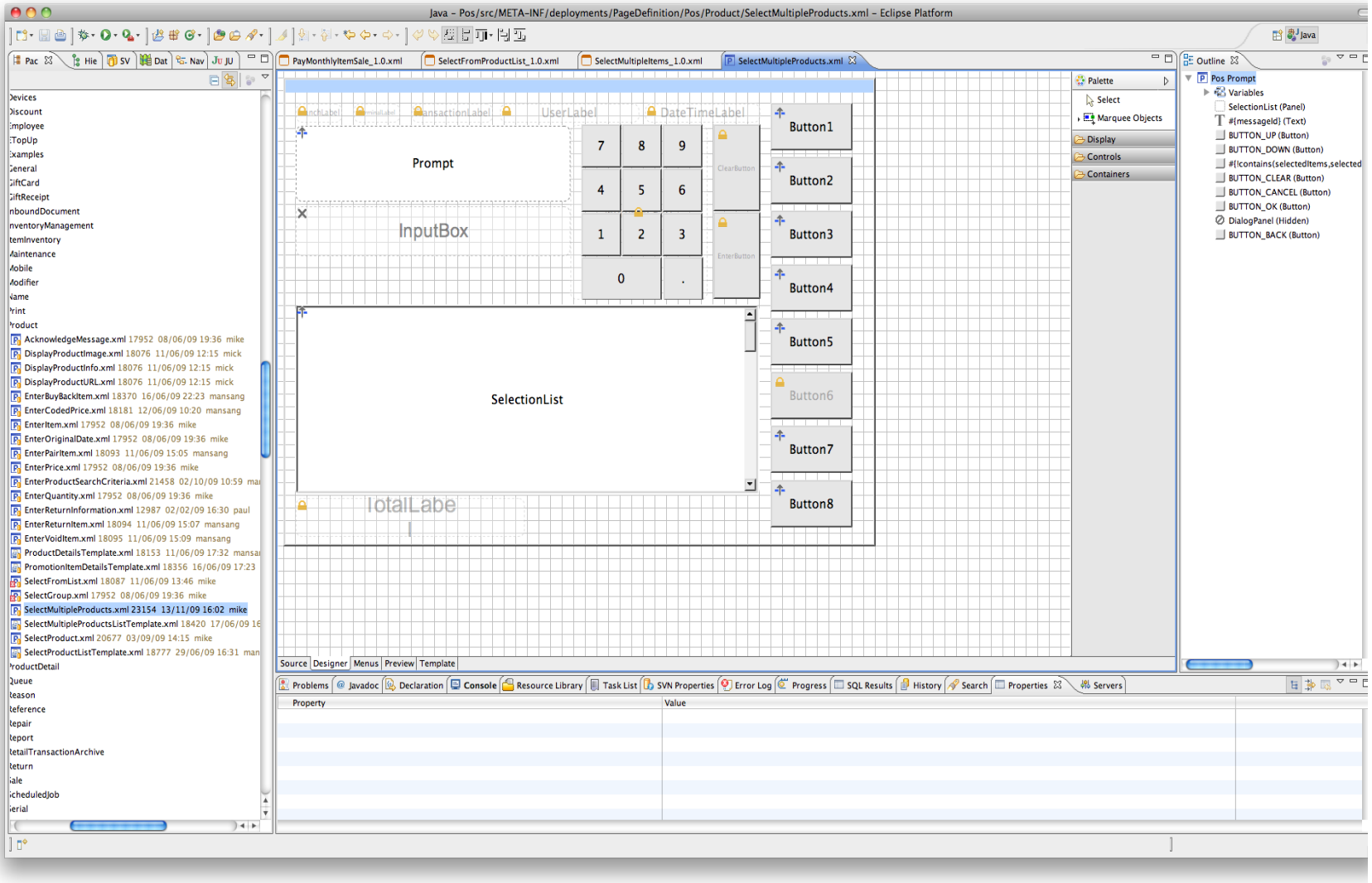
- All Actions include a set of meta-data as Java Annotations that describe:
  - The Inputs an Action expects, including their Data Type and Optionality
  - The Output an Action can return
  - The Outcomes an Action may report

Application Process Editor (cont.)

Tips and Tricks – Synchronizing Actions

- When using the Synchronize option you have the choice to:
  - Update the Figure so that the displayed Inputs, Outputs and Outcomes match those declared in the Java class
  - Update the Java class so that the annotations match the Inputs, Outputs and Outcomes declared on the Figure
- The Synchronize option can also be used to synchronize:
  - A Call Process action with the declared Inputs and Outputs of the Process being called
  - A Prompt State with the variables and Events a Page Definition is using and raising

Page Definition Editor



Page Definition Editor (cont.)

- Documents that control the screen layout
- Can describe any validation that should be performed
- Designed to be client device independent
  - The same Page Definition supports rendering via a Web Browser, a Swing application or via a mobile application (iOS and Android) to produce the same display

Page Definition Editor (cont.)

Layout Constraints

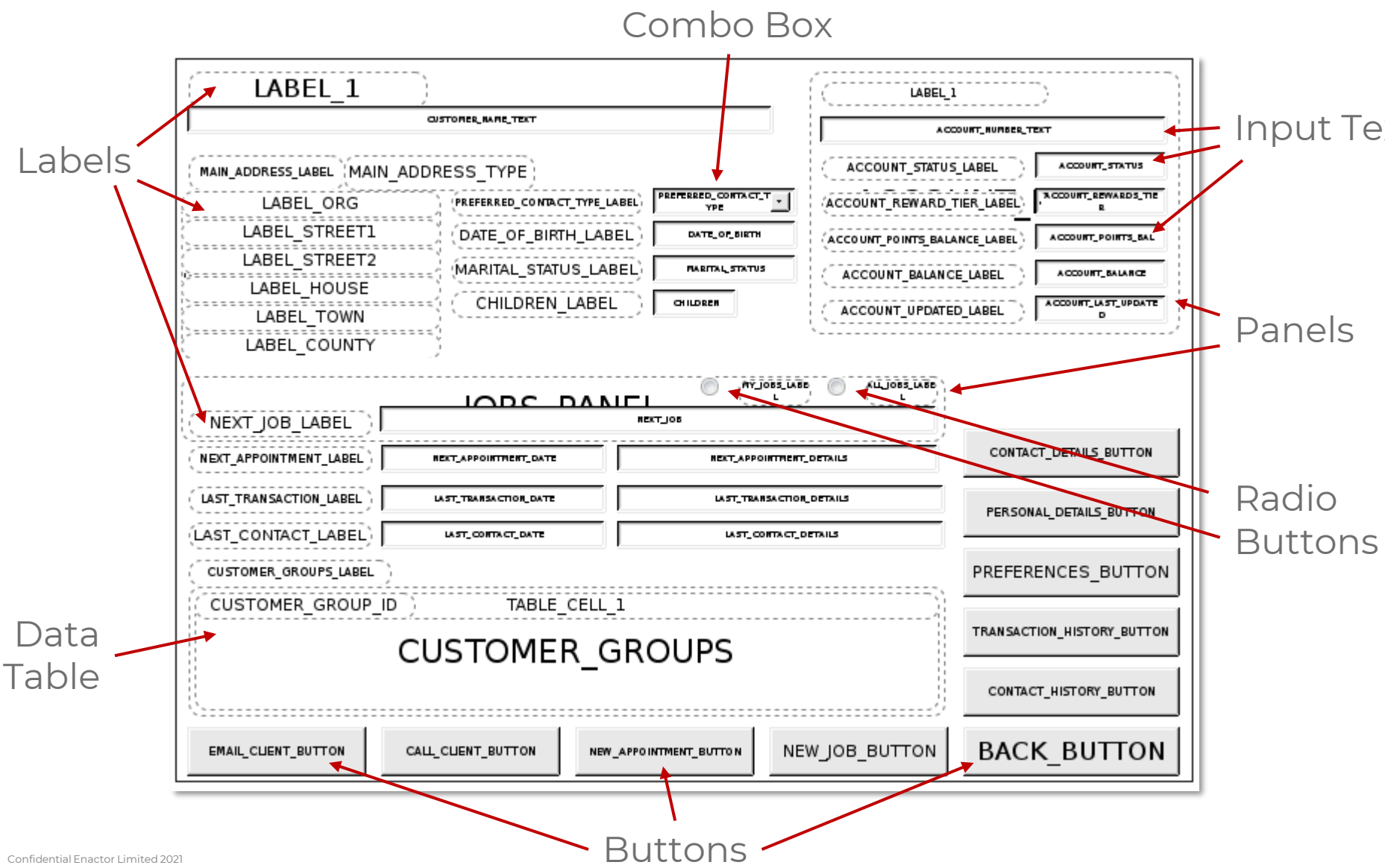
- Supports both Bounded and Flow based position
  - Bounded Positioning
    - Best for Swing applications or screens that are not resizable
    - More control over the final position of the elements of the screen
  - Flow Positioning
    - Best for web based applications or screens that are resizable
    - Less control over the final position, but elements can grow dynamically in size
  - A combination of both styles can be used on the same Page, but not within the same ‘container’

Page Definition Editor (cont.)

Element Types

- Labels and Error Labels
- Images
- Buttons, including Check Box and Radio Buttons
- Text Fields
  - Support for "as you type" validation
- List Boxes
- Tables
  - Both fixed "grid" style, and dynamic tables are supported
- Tabs
- Embedded Browser Control

Page Definition Editor (cont.)



Expression Language (EL)

- Provides a mechanism for embedding code like structures in Processes and Page Definitions
- Can be used to perform tests, extract data or bind a field in a Page Definition to data in a Prompt
- In general will be enclosed with the #{ and } tokens, however in places that expect an expression this is not required
  - An example is Conditional Links – these are always expressions and hence do not require the #{ and } tokens
  - Most places that support EL have an Expression Editor; this will automatically insert the #{ and } tokens as necessary.

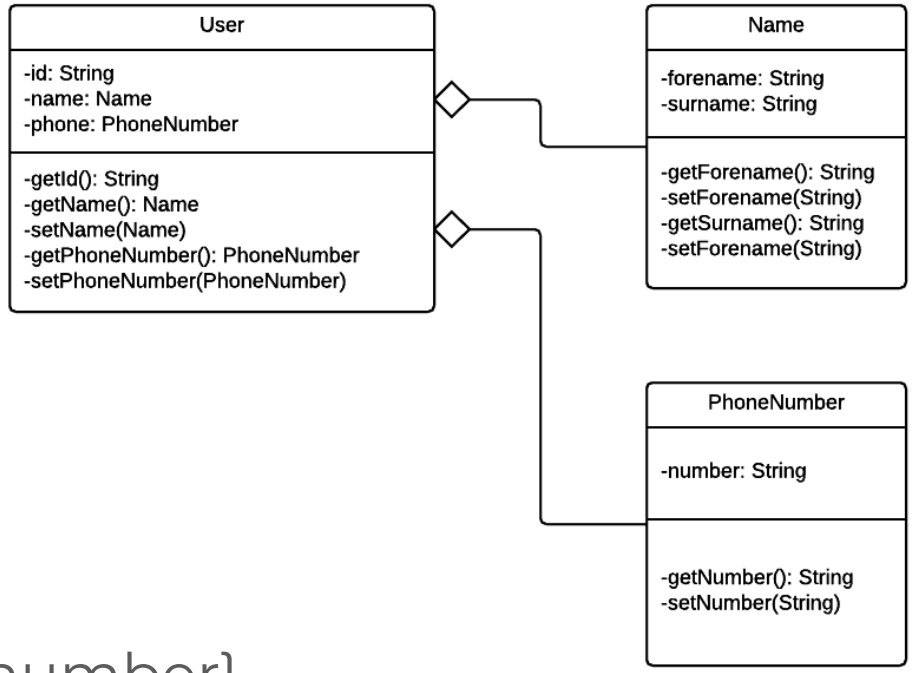
Expression Language (cont.)

- Uses a simple dot-notation to access ‘Java Bean’ enabled properties
- The Expression Editor helps you to build your expressions
  - Allows you to ‘drill-down’ into an object to inspect the available properties
  - Provides a list of the known ‘functions’ that can be called in the expression

Expression Language (cont.)

Examples

- Example 1, Extract the forename from the ‘User’ object:
  - #{user.name.forename}



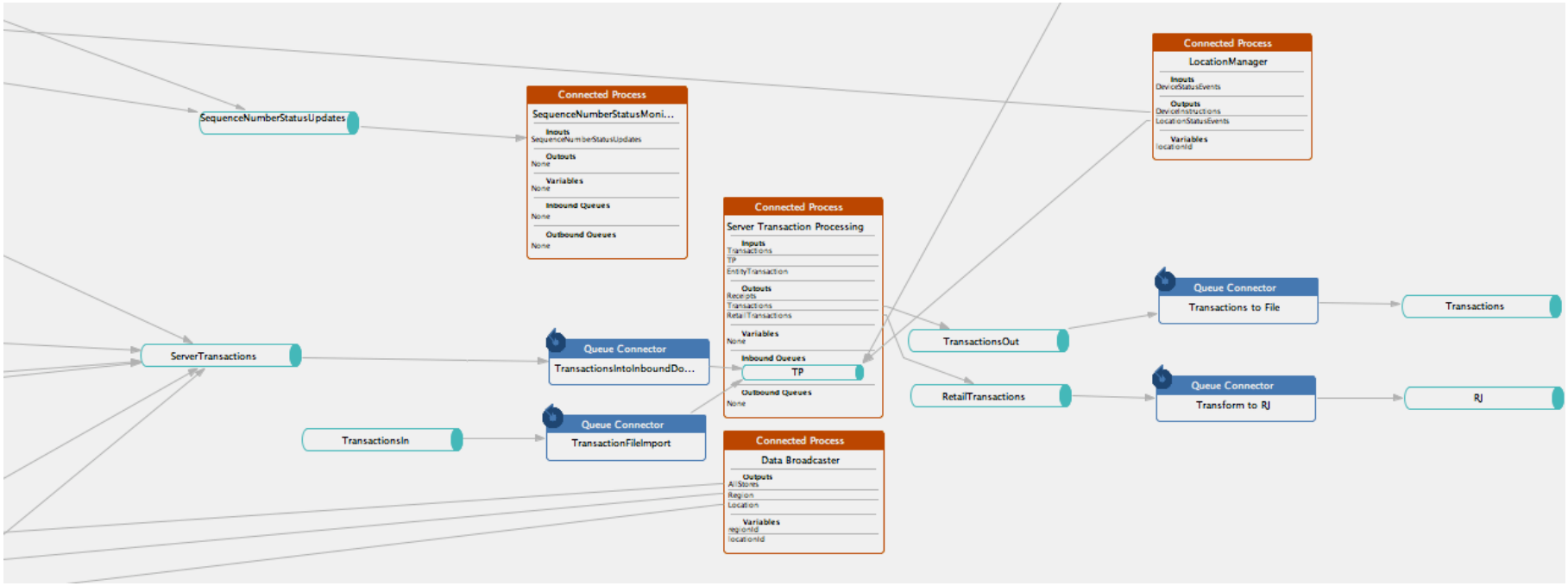
- Example 2, Extract the phone number from the ‘User’ object, and prefix it with ‘Tel: ’ :
  - Tel: #{user.phoneNumber.number}

- Example 3, If the surname is blank, return the User ID, otherwise return the surname:
  - #{empty(user.name.surname) ? user.id : user.name.surname}

Resource Library

- The Resource Library provides the ability to find and reuse existing elements of the Enactor Platform, primarily including Processes, Actions and Entities
- You can drag these element onto a Process and the tool will automatically add the correct reference to the element
  - Processes - Adds a 'Call Process' action to call the process
  - Actions - Adds a call to the Action
  - Entities – Creates actions to create an instance of the Entity and then to populate it
- You can include elements from source you have checked-out, or elements that are present in associated Jar files

Process Connection Editor



Process Connection Editor (cont.)

- Documents that control how the application talks to other tiers of the estate
- Control if a remote call should be done via a Web Service, or over a Queue
- Also allows for remote invocation of Actions and Processes

Process Connection Editor (cont.)

- All inter-application communication is controlled through Process Connections
- Process Connections defines
  - Connected Applications
  - Queue Service Providers (JMS, DB, HTTP, File)
  - Queues
  - Web Services
- Applications have defined connection points which can be linked to queues and web services

Process Connection Editor (cont.)

- Queues can be added, merged, split simply by modifying the diagram
- Queue names are not pre-defined by the application only the connection points
- One diagram defines the entire estate's connections
- Connections are defined by application and host id (Master Pos, Back Office, Estate Manager)
- The connection strategy is defined in the enactor.xml file
  - ProcessConnections.DefinitionId
- XSLT Transforms can be defined for queue and web service connections

Process Connection Editor (cont.)

Queue Connectors

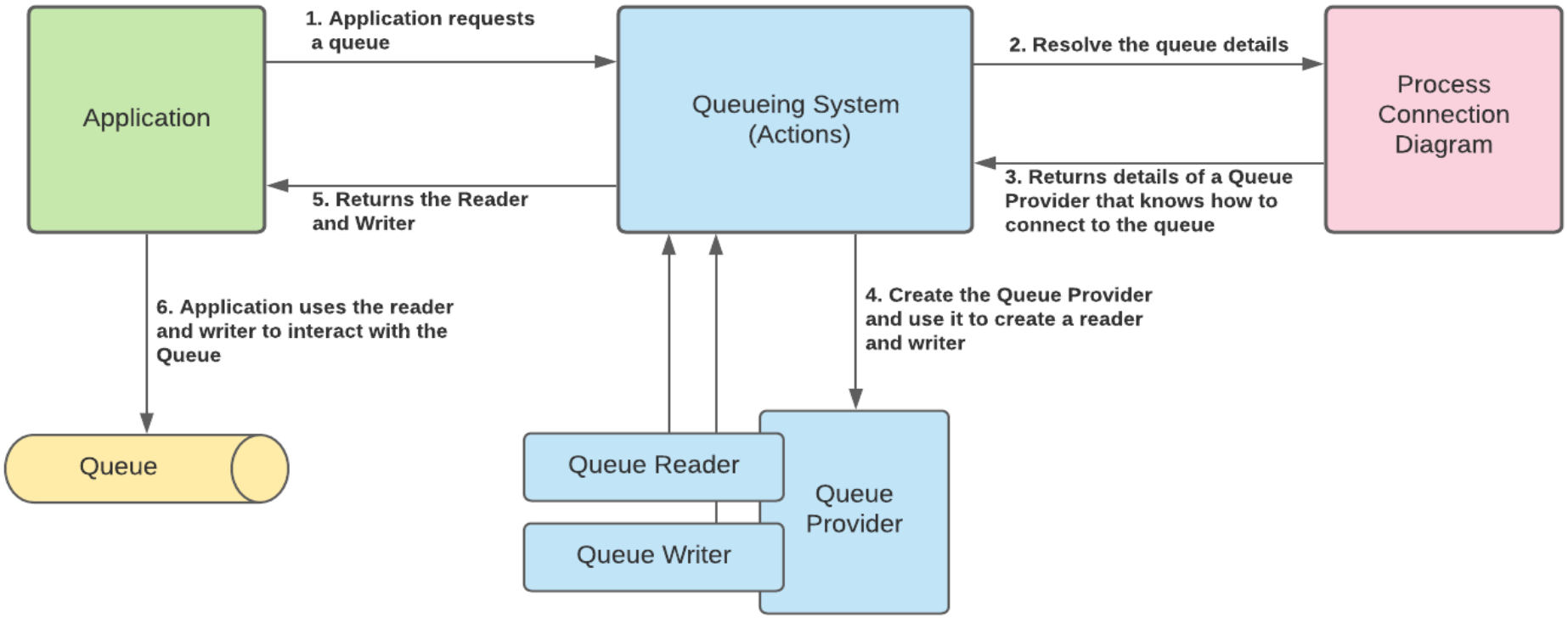
- The Queue Connector is a Service
- It should typically be run on every node in an enterprise which has the Enactor application installed
- It is responsible for all queue transfers regardless of queue provider
- It is multithreaded
- Messages are moved between queues in batches for efficiency

Process Connection Editor (cont.)

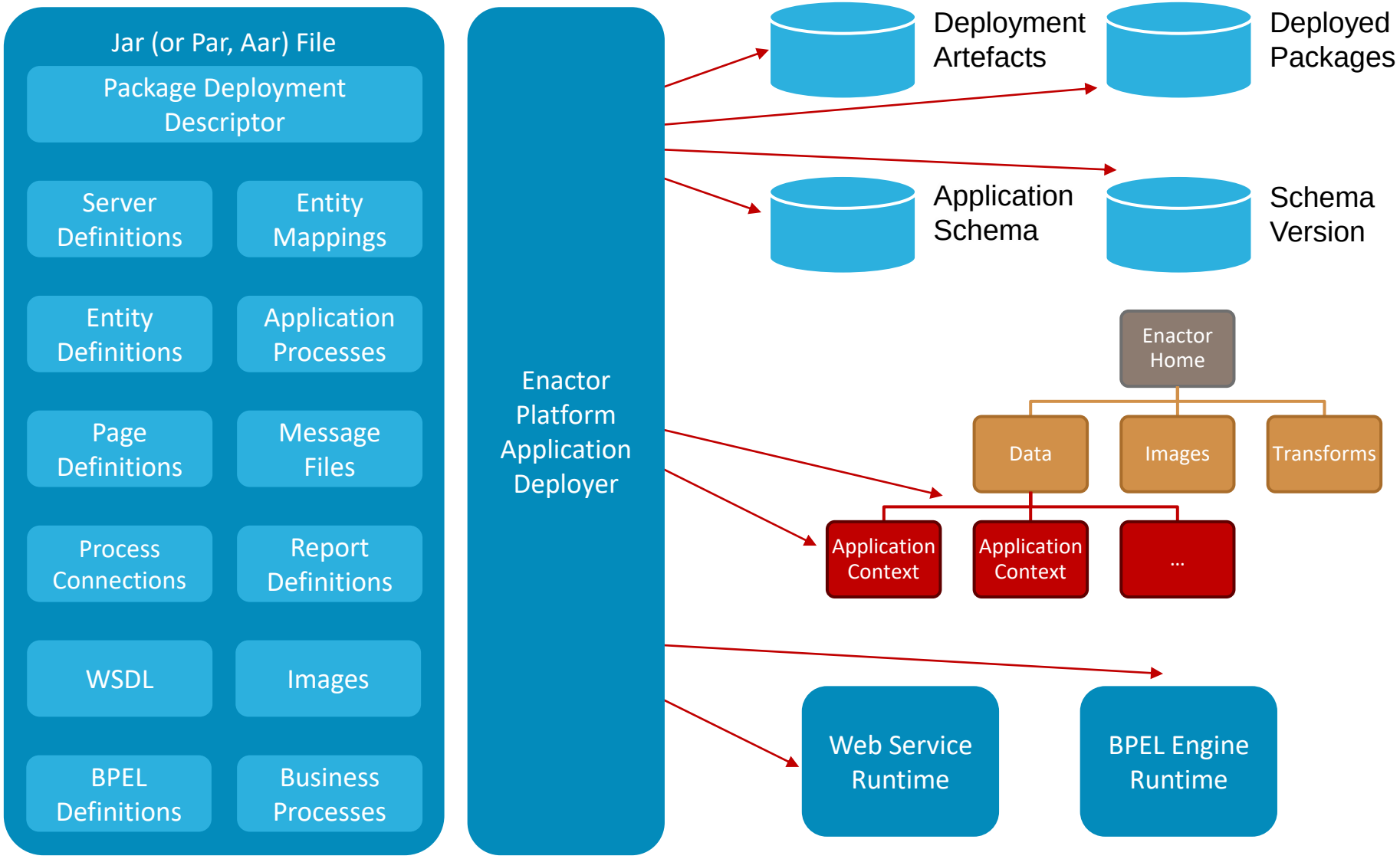
Queue Connectors

- Transforms can be applied as part of the transfer
- File based queue batches can be handled in zip files
- File Queues are located under {Enactor Home}/Queues
- Statistics are available through JMX

Process Connection Editor (cont.)



Application Deployment Overview



Seamless Integrated Consistent Personalised

enactor

retail systems for a digital world

Q & A