Seamless Integrated Consistent Personalised

enactor®

retail systems for a digital world

Enactor Training Course

Customising Enactor

Customising Enactor

Agenda

- Process Sets
- Themes
- Application Extensions
- Hook Processes

Process Sets

- Process Sets provide a mechanism to override standard Enactor Application Processes
- Process Sets are configured using the enactor.xml:
  - Common.ProcessSet
  - Common.ParentProcessSet
- Process Set may also be configured on the command line, or for the mobile devices in the application preferences.
- Are intended to be used for ‘application wide’ overrides
- Unlike other override mechanisms, Process Set only applies to Application Process

Process Sets (cont.)

- When Processes are referenced in the application, the Platform will use the Process Set to locate them
  - For example, if the Process Set was 'MyCompany' and a Process 'Pos/Sale_1.0.xml' was requested, the Platform will look for the Process 'MyCompany/Pos/Sale_1.0.xml'
  - If the Process cannot be found using the Process Set, the Platform will fall-back to using the originally requested name

Process Sets – Hook Processes

- To support extension in earlier versions of Enactor, Hook Processes were supplied
- These are Processes that Enactor will always ensure are empty, providing a place for Customers to inject custom function
- They can be identified by the suffix 'Hook' or 'External' in the Process name
    - For example Pos/Employee/CaptureEmployeeSaleHook
- This is replaced by Extension Points for new overrides, but existing Hook processes will be retained for backwards compatibility

Themes

- Themes provide a more flexible approach to application customisation than Process Set
- The Theme can be configured:
  - Using the enactor.xml:
    - Common.Theme
    - Common.ParentTheme
  - Using the command line, or for the mobile client in the application preferences
  - Using a Pos Terminal associated with the Device

Themes (cont.)

- Themes may be different between different Devices in the same estate
- Themes apply to Application Processes, Page Definitions and Print Documents
- The Theme can be changed during the lifetime of an application

Themes (cont.)

- When resources are resolved using a Theme, the Platform will allow for a bi-directional, hierarchical lookup
  - For example, if the Theme was ‘MyCompany/HighRes’ and the Page Definition ‘Pos/CaptureCustomer’ was requested, the Platform will look for the following resources:
    - MyCompany/HighRes/Pos/CaptureCustomer
    - MyCompany/Pos/CaptureCustomer
    - HighRes/Pos/CaptureCustomer
    - Pos/CaptureCustomer

Application Extensions

- Application Extensions provide a mechanism to ‘automatically’ enable additional functionality
- Application Processes can refer to ‘Extension Points’ by using the UICallExtensionPointProcessAction
- Developers then declare, using the Packages.xml, that they want to implement an Extension Point
- The Platform will discover the implementations and use them automatically as the Application Process is run

Application Extensions (cont.)

- Multiple implementations can be defined for a single Extension Point
  - The developer can specify a priority ordering by referring to the Package ID (from the Packages.xml) that they should be run before or after
  - It is also possible to declare that a given implementation must override an existing implementation, suppressing the original behaviour
  - If no priority ordering is configured, the order the implementations will be applied in is undefined

Application Extensions (cont.)

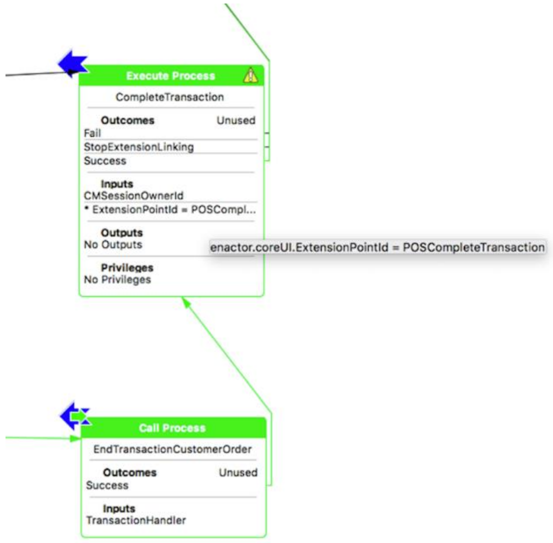
- When multiple implementation are present for an Application Process Extension, they will be invoked using the ordering information in the Packages.xml one after the other
  - Earlier implementations can pass information to following ones using the outputs of the Process
  - Earlier implementations can suppress further implementations by returning a special Outcome on the End Process Action:
    - enactor.action.StopExtensionLinking
  - The Outputs and Outcome from the last extension are returned to the calling Process

enactor

Application Extensions (cont.)

Example – Defining an extension

- First, in the Process, use the UICallExtensionPointProcessAction and configure an Extension Point ID



- Then in the Packages.xml declare the extension with a default implementation:

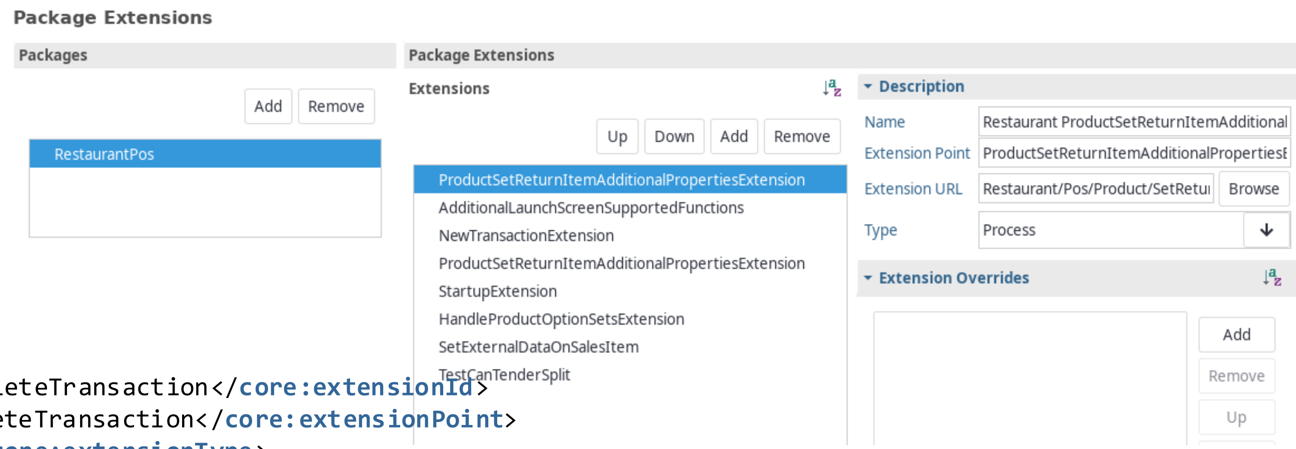
```
<core:extensions>
    <core:packageExtension>
        <core:extensionId>POS CompleteTransaction</core:extensionId>
        <core:extensionPoint>POSCompleteTransaction</core:extensionPoint>
        <core:extensionType>Process</core:extensionType>
        <core:extensionUrl>Pos/EndTransaction</core:extensionUrl>
    </core:packageExtension>
</core:extensions>
```

- You can omit the default implementation if one is not required

Application Extensions (cont.)

Example – Overriding an extension

- Taking the previous example, we simply add a extension point override to our Packages.xml:



Package Extensions

Packages	
RestaurantPos	

Package Extensions — Extensions

ProductSetReturnItemAdditionalPropertiesExtension
AdditionalLaunchScreenSupportedFunctions
NewTransactionExtension
ProductSetReturnItemAdditionalPropertiesExtension
StartupExtension
HandleProductOptionSetsExtension
SetExternalDataOnSalesItem
TestCanTenderSplit

Description

Name	Restaurant ProductSetReturnItemAdditional
Extension Point	ProductSetReturnItemAdditionalPropertiesE
Extension URL	Restaurant/Pos/Product/SetRetui
Type	Process

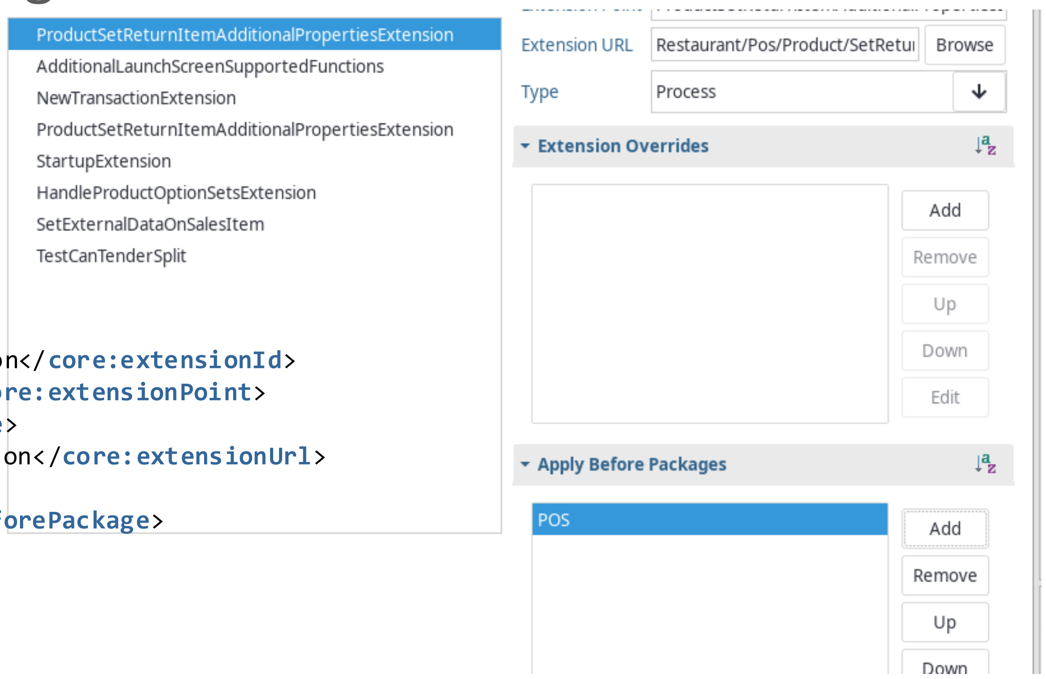
Extension Overrides

```
<core:extensions>
    <core:packageExtension>
        <core:extensionId>Orders CompleteTransaction</core:extensionId>
        <core:extensionPoint>POSCompleteTransaction</core:extensionPoint>
        <core:extensionType>Process</core:extensionType>
        <core:extensionUrl>Orders/Pos/CompleteTransaction</core:extensionUrl>
        <core:extensionOverrides>
            <core:extensionOverride>
                <core:extensionPoint>POSCompleteTransaction</core:extensionPoint>
                <core:packageId>Pos</core:packageId>
            </core:extensionOverride>
        </core:extensionOverrides>
    </core:packageExtension>
</core:extensions>
```

Application Extensions (cont.)

Example – Controlling the ordering

- To control the order extensions are applied, use the following in your Packages.xml:



ProductSetReturnItemAdditionalPropertiesExtension
AdditionalLaunchScreenSupportedFunctions
NewTransactionExtension
ProductSetReturnItemAdditionalPropertiesExtension
StartupExtension
HandleProductOptionSetsExtension
SetExternalDataOnSalesItem
TestCanTenderSplit

Extension URL	Restaurant/Pos/Product/SetRetu
Type	Process

Extension Overrides

Apply Before Packages

POS

```
<core:extensions>
    <core:packageExtension>
        <core:extensionId>Restaurant CompleteTransaction</core:extensionId>
        <core:extensionPoint>POSCompleteTransaction</core:extensionPoint>
        <core:extensionType>Process</core:extensionType>
        <core:extensionUrl>Restaurant/CompleteTransaction</core:extensionUrl>
        <core:applyBeforePackages>
            <core:applyBeforePackage>POS</core:applyBeforePackage>
        </core:applyBeforePackages>
        <core:extensionOverrides/>
    </core:packageExtension>
</core:extensions>
<core:dependencies>
    <core:packageDependency>
        <core:packageId>POS</core:packageId>
    </core:packageDependency>
</core:dependencies>
```

Seamless Integrated Consistent Personalised

retail systems for a digital world

Q & A