

The background of the slide is a scenic photograph of a mountain landscape. In the foreground, two hikers with backpacks are walking away from the camera on a rocky, dirt trail. To the left, a small, calm lake is nestled in a valley. The middle ground shows rolling green hills and rocky patches. In the background, majestic, rugged mountains rise under a blue sky with scattered white clouds. A semi-transparent geometric pattern of white lines and polygons is overlaid on the left side of the image.

*rego*University 2018

# Administration Advanced

Your Guides: Leo D'Souza and Dave Lord

# Introductions

- Take 5 Minutes
- Turn to a Person Near You
- Introduce Yourself
- Business Cards





# Course Outline

## Advanced Objects and Fields

- CA PPM Studio Overview
- Objects
- Attributes and Fields
- Views
- Action Menu

## Basic SQL

- What is SQL?
- Using SQL within CA PPM
- Basic SQL Syntax
- What Else can SQL do

## Portlets, Queries, Lookups

- Lookups
- Queries
- Portlets

## Advanced Basic Processes

- Overview
- Design Basics

## Advanced UI Themes, Menus, Options, Financials

- User Interface Themes
- Menu Configuration
- System Options
- Financial Administration

## XOG

- Introduction
- Requirements
- Installation and Setup
- XOG Files and Procedure
  - Configuration (Query, Portlet, Workflow, etc.)
  - Web Version (13.1 and below)
  - Data
- Best Practices
- Other XOG Methods and XML Creation

# Advanced Objects And Fields: Course Outline

CA PPM Studio Overview

Objects

Attributes and Fields

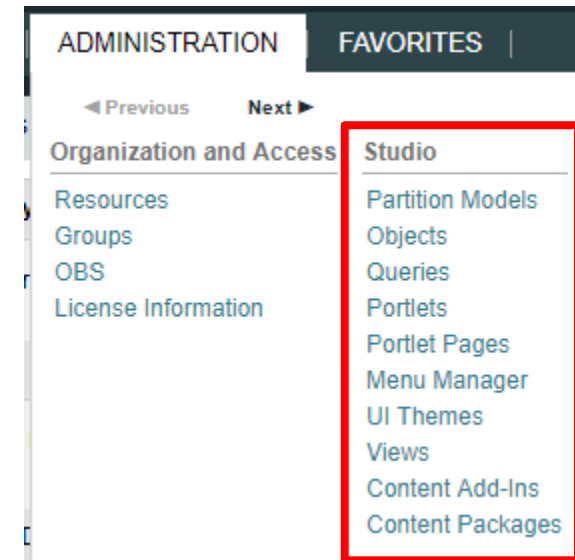
Views

Action Menu

*rego*University 2018



- CA PPM Studio is the interface used to create and deploy portals, dashboards, menus, and objects that can be configured or customized to match organization needs
- An organization must have a CA PPM Studio license to use this functionality
- The user must have “Administration-Studio” access assigned as well as rights to create/edit Objects, Portlets, and Pages



# Objects

- Objects are the major functional components of CA PPM
- Objects define the attributes (fields), subpages (links), page layout, and views that make up your configured instance of CA PPM
- In addition to the stock objects delivered with the system, you can create custom objects. Custom objects are essentially tables inside the database that begin with “ODF\_CA”
- Use the default objects or create custom objects and sub-objects to manage information for specific business needs
- Once you create an object, add attributes, links, and actions and set up the views



# Objects (2)

- Each object has four distinct pieces you can configure
  - Properties
  - Attributes
  - Links
  - Views
- Things to remember
  - You can only delete Custom Attributes
  - Adding more than 100 custom attributes to a single custom object may impact performance
  - A hierarchy with a maximum of three levels of objects can be created, and allow child objects to inherit properties and access rights from parent objects

# The Investment Object

- Allows you to define object attributes used across multiple objects (Project, Idea, Application, Asset, Product, Service, Other Work)
- Streamlines the creation process and ensures consistency across objects
- You may re-label attributes on shared objects if needed (Attribute ID remains the same)
- Attributes defined at the investment level are available to the stock objects noted above but are not required
- You must make updates to Investment attributes at the Investment level

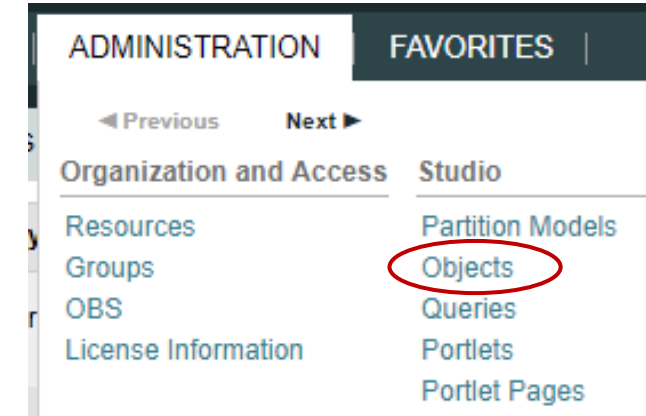


# Objects Types

- Stock Objects
  - Primary Standard Objects
    - Project
    - Task
    - Team
    - Resource
    - Company
    - Application
- Custom Objects
  - Master Objects
  - Sub-Objects

# Exercise #1: Create an Object

- Create a new custom sub-object to the project object
  - Administration -> Objects
  - Click New and fill out the required fields (see next slide)
- Select the following checkboxes if they apply
  - **Event Enabled:** Specifies that the process engine is notified of object instances that are created or updated. (If a process needs to get driven off the object)
  - **Copy Enabled:** Specifies that copies can be made of the object instances.
  - **Export Enabled:** Specifies that object instances can be exported to XML.
  - **View All Enabled:** Specifies that the object instances can have a view containing all properties, sub-object lists, and page portlets that can be personalized on a single page.
- Notice the default fields included in the newly created object





# Exercise #1: Create An Object (2)

Create Object Definition

Object Name: Training Object

Object ID: rego\_training

Content Source: Customer

Description: This object is used for training purposes

Master or Subobject: ☒ Master

Partition Model:

Subobject:

Master Object:

Event Enabled: ☐

Copy Enabled: ☐

Export Enabled: ☐

View All Enabled: ☐

Save Save And Return Return

\* = Required → = Enter Once \* = Unique

**Object Name:** Meaningful name summarizing the object function

**Object ID:** Use a standard naming convention; many start with a customer ID + “\_” (special characters and spaces should not be used in IDs)

**Content Source:** Defaults to “Customer”; categorizes object

**Description:** Detailed explanation of the object’s purpose

→ Master or Subobject ☐ Master

Partition Model

☒ Subobject

→ Master Object

## Master or Subobject

- Choose **Master** if object is to be standalone
- Choose **Subobject** if object is to be accessed via another object (1 to many relationship)
  - Choose the Master object by using the browse button

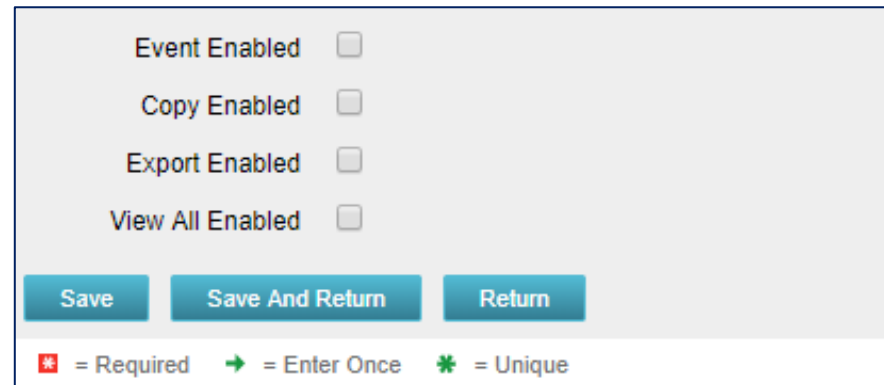
## a. Select the following checkboxes if they apply

- **Event Enabled:** Specifies that the process engine is notified of object instances that are created or updated. (If a process needs to get driven off the object)
- **Copy Enabled:** Specifies that copies can be made of the object instances.
- **Export Enabled:** Specifies that object instances can be exported to XML.
- **View All Enabled:** Specifies that the object instances can have a view containing all properties, sub-object lists, and page portlets that can be personalized on a single page.

➤ Notice the default fields included in the newly created object

# Exercise #1: Create an Object (3)

- Select the following checkboxes if they apply
  - **Event Enabled:** Specifies that the process engine is notified of object instances that are created or updated. (If a process needs to get driven off the object)
  - **Copy Enabled:** Specifies that copies can be made of the object instances.
  - **Export Enabled:** Specifies that object instances can be exported to XML.
  - **View All Enabled:** Specifies that the object instances can have a view containing all properties, sub-object lists, and page portlets that can be personalized on a single page.



The screenshot shows a form with four checkboxes, each with a red asterisk indicating it is required:

- Event Enabled ☐
- Copy Enabled ☐
- Export Enabled ☐
- View All Enabled ☐

Below the checkboxes are three buttons: "Save", "Save And Return", and "Return".

At the bottom of the form, there is a legend:

- \* = Required
- = Enter Once
- \* = Unique

# Exercise #1: Create an Object (4)

- Notice the default fields included in the newly created object

Properties Attributes Linking A

Object: Training Object - Attributes

Attribute Name

Attribute Display All Attributes

Filter Show All Clear

<input type="checkbox"/>	Attribute	Description
	Created By	
	Created Date	
	ID	
	Last Updated Date	
	Name	
	Page Layout	Page Layout
	Partition	Code that identifies the
	Updated By	

**Created By and Created Date:** Keep track of who and when the record was created

**Last Updated By and Updated By:** Keep track of the last person who updated the record and when

**\*\* Note:** Outside of custom objects there are OOTB jobs / processes that will scew the results of the last updated by and date fields, as the application often makes updates to the record

**Page Layout:** Each object defaults to a standard layout with tabs such as Properties, Processes, and Audit. This can be customized by adding a new custom page layout. (Details later on)

**Name and ID:** Used to identify the record; Name can be repeated multiple times while the ID has to be unique. Auto-numbering is often used to force that uniqueness and standardization.

# Objects: Attributes

- Attributes are the fields on any object that store information
- The attributes of each object are available on the Attribute screen within the object
- Many attributes are delivered out-of-the-box, but you can create an unlimited amount of additional attributes using CA PPM Studio
- Once created, you can organize and place attributes on views and portlets and use for reporting
  - Example: “Start Date” is an attribute of the project object

# Objects: Attribute Data Types

- When creating a new attribute, the procedure used depends on the data type selected
- The following data types are available for creating attributes
  - String (2000 character maximum)
  - Large String
  - Number
  - Formula
  - Money (includes currency code)
  - Boolean (checkbox)
  - Date
  - Lookup (related lookup needs to be available / created prior to creating attribute)
  - Multi-Valued lookup (related lookup needs to be available / created prior to creating attribute)
  - Attachment
  - Time-varying
  - URL (Links to actual data)
  - Virtual fields (Not actual data)
- \*\* More in depth descriptions of how to create each field is in the Studio Developer Guide (pg. 25)



# Objects: Calculated Fields

- A Calculated Field is an attribute that displays a dynamically-calculated read-only value
- Values are calculated from existing system values
- Values are calculated when accessing the page
- Values are not stored in the database
- Calculated fields can be only one of the following data types
  - **Number:** Use this data type when a calculated attribute requires a number value like a sum or average
  - **String:** Use this data type when a calculated attribute requires the concatenation of two or more values
    - Example: a concatenation of the value of the attribute “created\_by” and the constant “2007” would produce a result of “ssmith 2007”
  - **Date:** Use this data type when you need to calculate dates using basic arithmetic or to provide the current date

# Objects: Calculated Fields

- You cannot use the following attribute types with calculated attributes
  - Formula
  - Time-varying
  - Attachment
  - Long String
  - Multi-Value Lookups
  - Virtual
- **Note:** You cannot delete source fields for a calculated attribute

# Objects: Auto Numbering

- Often businesses want a meaningful unique identifier within object instances; Auto-numbering using text and numeric values will accomplish this
  - Within an object select an attribute, usually the ID and/or the Name
  - Select the auto-numbering tab
  - Select Scheme -> Edit
  - The default segment type is numeric but this can be modified to include text characters as well
  - Set the counter length

# Objects: Lookups

- A lookup is a field the user can select from a drop-down or pull down a list of pre-defined choices
- Lookup field choices can be static values entered by an administrator, or dynamic values returned from a database query
- Lookups can display as a value or an icon

Lookup Type	Description
Static List	Use this type of lookup when working with a standard set of values. Static list lookups are often used as pull-down lists for fields, reports, and custom forms.
Static Dependent Lists	Use this type of lookup to create a hierarchy of lookups and values. Items that appear on the second and subsequent lists depend upon choices previously made by the user. For example, if the user selects "USA" from a country browse list, then a state list may appear from which the user can select an appropriate state.
Dynamic Queries	Use this type of lookup to capture data from the CA PPM database in real time to populate the drop-down or browse lists. (Using NSQL) These lookups provide the most up-to-date values possible and are often used inside browse windows.

**Note:** You can nest a static lookup inside a dynamic query lookup. You cannot nest a static dependent lookup inside a dynamic query lookup.

# Exercise #2: Objects And Attributes

- Add at least 3 or 4 attributes of different varieties to your new custom sub-object
  - Administration -> Objects -> <New Object> -> Attributes

The screenshot shows the 'Attributes' tab of a configuration interface. At the top, there are tabs for 'Properties', 'Attributes', 'Linking', and 'A'. Below the tabs, the text 'Object: Training Object - Attributes' is displayed. The main area contains a form with two input fields: 'Attribute Name' and 'Attribute Display', with 'All Attributes' selected in the second field. Below these fields are three buttons: 'Filter', 'Show All', and 'Clear'. At the bottom, there is a table with columns for 'Attribute' and 'Description'.

<input type="checkbox"/>	Attribute	Description
<input type="checkbox"/>	Created By	
<input type="checkbox"/>	Created Date	
<input type="checkbox"/>	ID	
<input type="checkbox"/>	Last Updated Date	
<input type="checkbox"/>	Name	
<input type="checkbox"/>	Page Layout	Page Layout
<input type="checkbox"/>	Partition	Code that identifies the
<input type="checkbox"/>	Updated By	



# Exercise #2: Objects And Attributes (2)

- Click New and fill out required fields as well as Data Type of new field


Object: Training Object | Attribute: - Object Attribute

**\* Attribute Name**

**\* → \* Attribute ID**   
( ID must be alphanumeric, underscore is permitted. It must not be a SQL or Clarity reserved word. )

**Description**

**→ Data Type**

**\* → Lookup**  

**Default**   
( Click Save to update this field after selecting a new lookup. )

**Populate Null Values with the Default** ☐

**Value Required** ☐

**Presence Required** ☐

**Read-Only** ☐  
( In order to make an attribute read-only a default must be selected )

\* = Required   → = Enter Once   \* = Unique

**Attribute Name:** Name of attribute (display name can be changed later in fields if need be)

**Attribute ID:** unique ID for attribute (spaces and special characters not allowed)

**Description:** Meaningful explanation for attribute usage

**Data Type:** Type of attribute

**Lookup:** Only shows up for “Lookup” and “Mutli Valued Lookup” types

**Default:** Default value to populate attribute with ( if applicable)

# Exercise #2: Objects And Attributes (3)

Populate Null Values with the Default ☐

Value Required ☐

Presence Required ☐

Read-Only ☐

( In order to make an attribute read-only a default must be selected )

**Save** **Save And Return** **Return**

\* = Required   → = Enter Once   \* = Unique

- Select the following checkboxes if they apply:
  - **Populate Null Values with the Default:** If an attribute is created later on and instances have already been created this will populate the new attribute values with the default value set
  - **Value Required:** Specifies whether a value is required for the attribute.
  - **Presence Required:** Specifies that the attribute always appears in the Edit Properties view.
  - **Read-Only:** Specifies that a user cannot make changes to the value in the attribute

# Exercise #2: Objects And Attributes (4)

- Auto-number the ID attribute
  - Within an object select an attribute, usually the ID and/or the Name
  - Select the auto-numbering tab
    - Select the “Auto-numbered” checkbox and click “Save”
    - Select Scheme -> Edit

Properties Auto-numbering

Object: Training Object | Attribute: ID - Attribute Auto-numbering

General

Auto-numbered ☒

Save Save And Return Return

Schemes

Partition: System ▼

Partition	Runtime Next Number	Scheme
System	00000001	[Edit]

# Exercise #2: Objects And Attributes (5)

- The default segment type is numeric but this can be modified to include text characters as well

Object: Training Object | Attribute: ID | Partition: System - Auto-numbering Scheme

**General Information**

☒ Scheme Name System Default

Partition System

Next Number 00000001

Status Active

**Segments**

<input type="checkbox"/>	Type of Segment	Text Value	Counter Starting Number	Counter Length	Auto-extended
<input type="checkbox"/>	Numeric Counter		00000001	8	✓

New Delete Reorder Save Save And Return Return

☒ = Required

- Select “New” and set Type of Segment = “Text” and type the text value into “Value” field.
- Click Save and Return

**Segment Properties**

Type of Segment Text

Value TRN

Save And Return Return

# Exercise #2 – Objects and Attributes (6)

Object: Training Object | Attribute: ID | Partition: System - Auto-numbering Scheme

**General Information**

✖ Scheme Name System Default

Partition System

Next Number 00000001TRN

Status Active

Note the new numbering scheme

**Segments**

<input type="checkbox"/>	Type of Segment	Text Value	Counter Starting Number	Counter Length	Auto-extended
<input type="checkbox"/>	Numeric Counter		00000001	8	✓
<input type="checkbox"/>	Text	TRN		3	

New Delete Reorder Save Save And Return Return

- The new scheme defaults into the order it was entered. To reorder to have the text in front select “Reorder” and move the segments accordingly.
- Select
- “Save and Return”
- when finished

Reorder Auto-numbering Scheme Segments

Scheme Segments

Text(Next Value: TRN)

Numeric Counter(Next Value: 00000001)

Save Save And Return Return



# Objects: Views

- Object views control what attributes users see / update
- There are four types of views:
  - **Create**: What the user sees when creating a new entry in the object
  - **Edit**: What the user sees when accessing a record in an existing object (for example, accessing a project)
  - **List**: What the user sees when first entering the object (for example, clicking the Projects link on the Home menu)
  - **Filter**: What the user sees within the list view, giving them the capability to further refine results

# Objects: Views (2)

- Within each view separate sub-pages and sections can be created to group attributes together in meaningful ways
  - A sub-page is accessed via a link in the drop down menu and are generally used to drive different functions
  - A section is an area on a page or sub-page that divides the page up into logical groupings
  - Note: A sub-page on a master custom object can also be configured as a tab instead of a link
- The same attribute can be placed on multiple sub-pages within the same object
  - This is normally done so that certain pages can be used for editing but the value will still display on another page

# Objects: Subpage vs. Section (Home)

The screenshot illustrates the difference between a subpage and a section in the Rego software. The top part shows a menu bar with 'Properties' circled in red. A dropdown menu is open, showing 'Main', 'Subprojects', 'Dependencies', 'Baseline', and 'Estimating'. 'Subprojects' is highlighted with a yellow box. A text box explains: 'Subpage shows up in the menu bar as a new page whereas a Section is an area on the same page'. Below this, the 'General' section of the 'Project: 2017 Software Implementation Project' is shown. The 'General' tab is circled in red. The section contains various fields: Project Name (2017 Software Implementation Project), Project ID (PR000009), Project Type (Application Change), Status (Approved), Charge Code ([--Select--]), Progress (Started), Project Manager (Bonham, Jessica), Stage (Initiation), Goal (Infrastructure Improvement), and Priority (25). The 'Description' section is also circled in red.

Project: 2017 Software Implementation Project - Properties - Main - General

**General**

\* Project Name 2017 Software Implementation Project

\* Project ID PR000009

Project Type Application Change

\* Status Approved

Charge Code [--Select--]

\* Progress Started

Project Manager Bonham, Jessica

Stage Initiation

Goal Infrastructure Improvement

Priority 25

**Description**

Description

# Objects: Subpage vs. Section (Home) (2)

Object: Status Report | Partition: System | View: General | Mode: Edit - *Property Layout*

<input type="checkbox"/>	Page > Subpage		Level
<input type="checkbox"/>	- [Edit Status Report Properties]		Page
<input type="checkbox"/>	+ General		Subpage
<div> <div>Create Subpages</div> <div>Return</div> </div>			

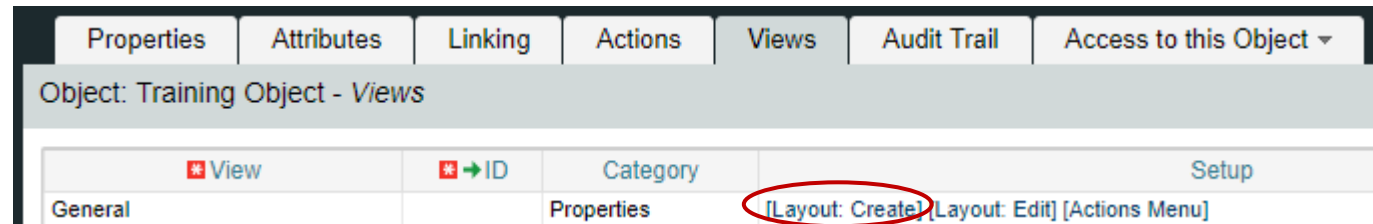
Top	<input type="checkbox"/>	Subpage > Section		Level
	<input type="checkbox"/>	- [General]		Subpage
	<input type="checkbox"/>	+ Status Report		Section
	<input type="checkbox"/>	+ Overall		Section
	<input type="checkbox"/>	+ Schedule		Section
	<input type="checkbox"/>	+ Scope		Section
	<input type="checkbox"/>	+ Cost and Effort		Section
<div> <div>Create Sections</div> <div>✓ Delete</div> <div>Return</div> </div>				

# Objects: Views

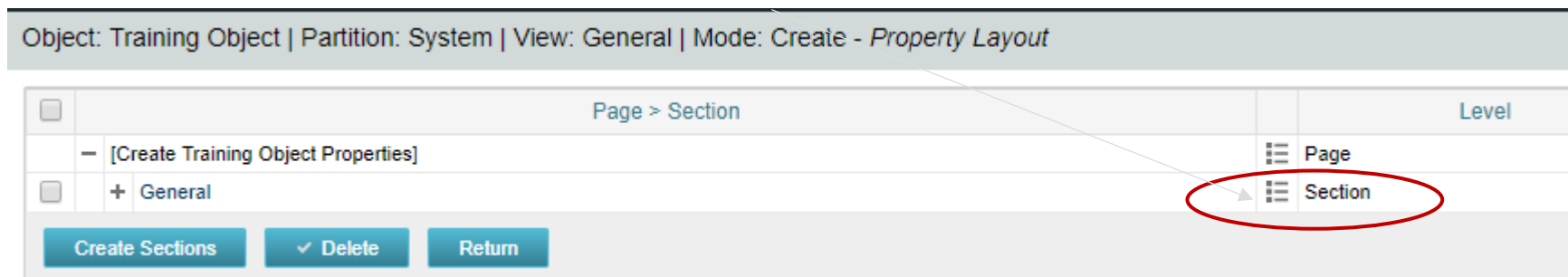
- You can show or hide views based on display conditions or securing the page. The differences in these two are defined below.
- **Display Conditions**
  - Defines a set of conditions that determine when a subpage appears. The expression builder is used to create these conditions and utilizes attributes available to the object or security groups
- **Secured Subpages**
  - Check the “secured subpage” box inside the page properties to create rights to either view or edit the values on this page
  - Secured subpages are not available on the Task object

# Exercise #3: Objects And Views

- Using the new object do the following:
- Modify the create view to add 1 or 2 attributes to General section of General subpage
  - Navigate to Administration -> Objects -> Views -> Layout:Create



- Select the properties icon to the left of the Section to alter the layout





# Exercise #3: Objects And Views (2)

- Move an attribute to the 2nd column using the arrows underneath the columns below

Object: Training Object | Partition: System | View: General | Mode: Create - Section Properties

**Layout**

Required fields are marked with an asterisk (\*)  
Hidden fields are marked with a tilde (~)

Available	Selected (Left Column)	Selected (Right Column)
Created Date Last Updated By Last Updated Date Partition~	Name* ID* Page Layout*	Created By

Add Field →      ← Move Field      ← Move Field

**Title**

Parent Create Training Object Properties

\* Section Name General

Save Save And Return Return

Note: The Name, ID, and Page Layout will show up by default as they are required. They can be hidden, if desired. (In later slides)

# Exercise #3: Objects And Views (3)

- Create a separate subpage within the Edit View
  - Navigate to Administration -> Objects -> Views -> Layout:Edit

View	ID	Category	Setup
General		Properties	[Layout: Create] [Layout: Edit] [Actions Menu]

- Select “Create Subpages” and fill out subpage “Name” and “ID”. Click Save and Return.

Page > Subpage	Page	Subpage
- [Edit Training Object Properties]	Page	
+ General	Subpage	

* Subpage Name	* Subpage ID
Admin	admin

# Exercise #3: Objects And Views (4)

- Add a section within the subpage by clicking on the subpage link first

Page > Subpage

-	[Edit Training Object Properties]	Page
+	General	Subpage
	Admin	Subpage

Create Subpages Delete Return

- Select “Create Subpages” and fill out subpage “Name” and “ID”
- Select “Create Sections” and type the name of each section.
  - You can add up to 5 at time
  - Save and Return

Top

Subpage > Section

[Admin]

Create Sections Return

Parent /Edit Training Object Properties/Admin

\* Section Names Admin

Save And Return Return

\* = Required

# Exercise #3: Objects And Views (5)

- Add an attribute to the new subpage and section by clicking on the section properties link, selecting the field(s) from the Available column and moving to the Left or Right Column(s)

Object: Training Object | Partition: System | View: General | Mode: Edit - *Property Layout*

Top	Subpage > Section	Level
<input type="checkbox"/>	[Admin]	
<input type="checkbox"/>	Admin	

Create Sections   Move   Delete   Return

Object: Training Object | Partition: System | View: General | Mode: Edit - *Section Properties*

Layout

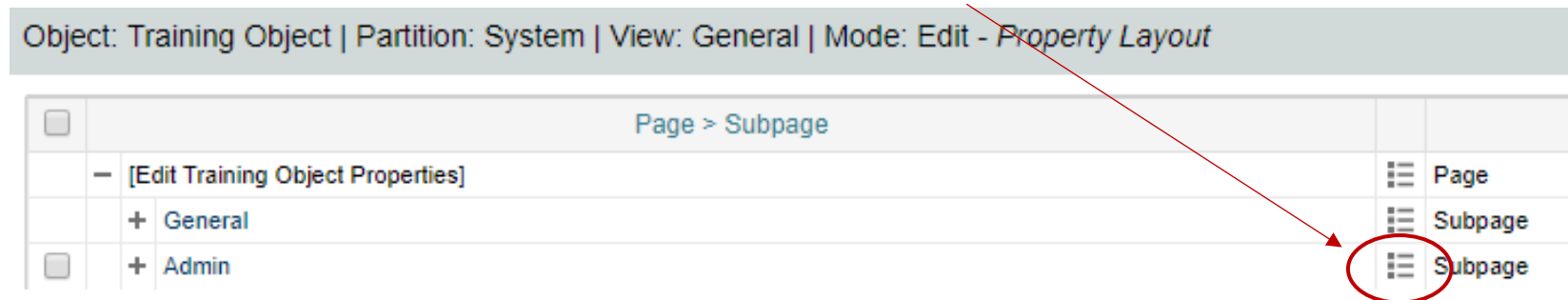
Required fields are marked with an asterisk (\*)  
Hidden fields are marked with a tilde (~)

Available	Selected (Left Column)	Selected (Right Column)
Name Page Layout Partition~ Last Updated By Last Updated Date	ID	Created By Created Date

Add Field →   ← Move Field   ← Move Field

# Exercise #3: Objects And Views (6)

- Create a display condition on the new sub-page
  - Navigate to Administration -> Objects -> Views -> Layout:Edit-> Admin Subpage and select the properties icon



- Select “Define Display Conditions” within the “Display Conditions” section



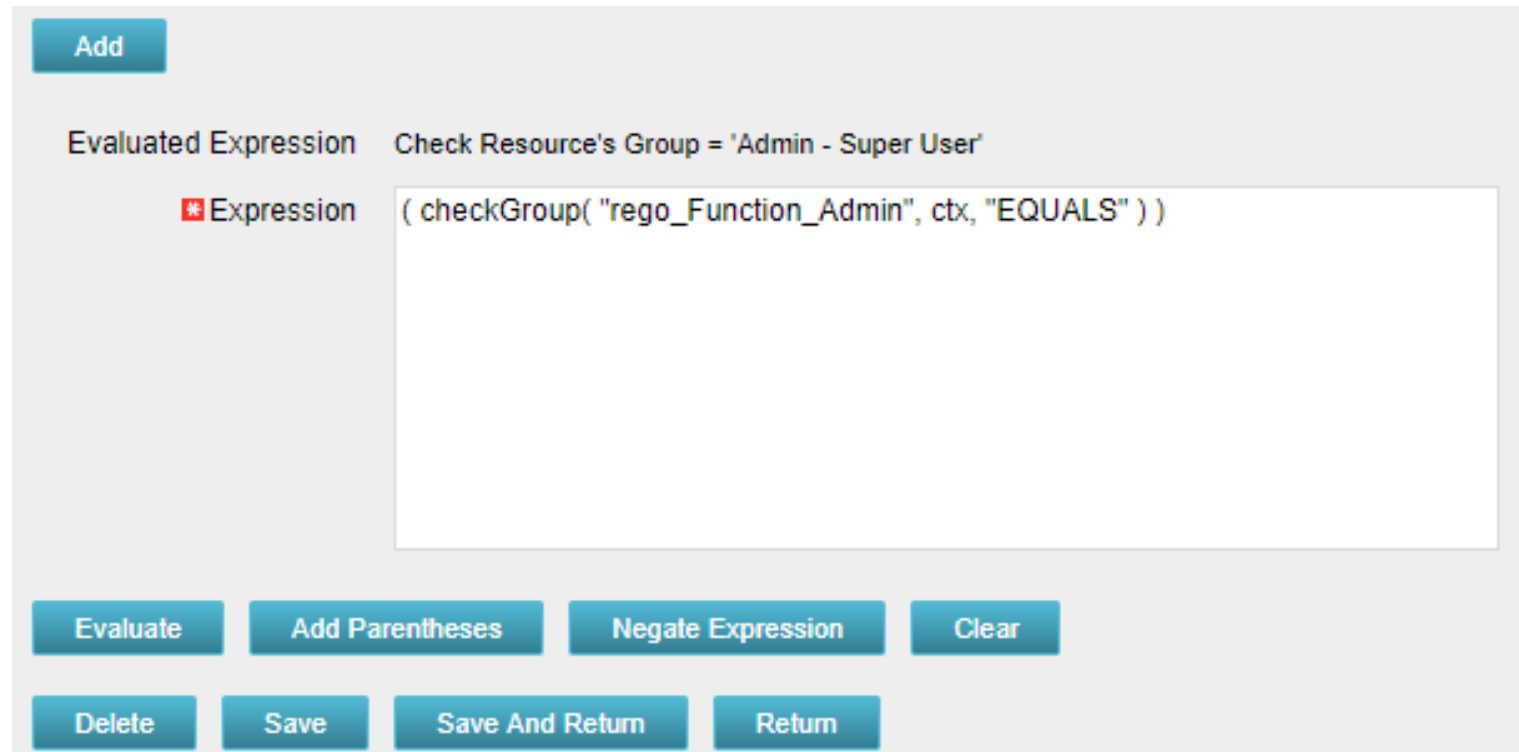
# Exercise #3: Objects And Views (7)

- Select the radio button next to “Operation” and in the “Operation” drop down select “Check Resource’s Group”
- Use the browse within the “Constant” box to choose the group for which the page should only display

The screenshot shows a web-based interface titled "Display Condition Builder". It contains several input fields and controls for building a display condition. On the left, there are labels for "And/Or", "Left", "Operator", and "Right". The "And/Or" field is a dropdown menu. The "Left" and "Right" fields each have a radio button, with the "Right" radio button being selected. The "Operator" field is a dropdown menu showing an equals sign (=). In the center, there is a label "Field" next to a dropdown menu showing "[--Select--]". Below this, there is a label "Operation" next to a dropdown menu showing "Check Resource's Group". At the bottom, there is a label "Constant" next to a text input field. This field has a browse button (a magnifying glass icon) and a list of suggestions, with "Admin - Super User" selected and highlighted.

# Exercise #3: Objects And Views (8)

- Select the “Add” button to build the expression and click “Save and Return”



The screenshot shows a web-based interface for editing Rego expressions. At the top left is a blue button labeled "Add". Below it, the text "Evaluated Expression" is followed by "Check Resource's Group = 'Admin - Super User'". To the left of a large text input area is a red asterisk icon and the label "Expression". The input area contains the code: `( checkGroup( "rego_Function_Admin", ctx, "EQUALS" ) )`. At the bottom, there are two rows of blue buttons. The first row contains "Evaluate", "Add Parentheses", "Negate Expression", and "Clear". The second row contains "Delete", "Save", "Save And Return", and "Return".



# Additional Fields Properties

- Fields are attributes and are available to be placed onto views, however they have some additional / different properties that allow flexibility in view configuration
- Properties of field attributes control how the field appears within the view itself. The following additional properties can be configured within the field:
  - Enter Once: Select this check box to prevent users from changing the attribute's value after it has been entered
  - Required: Select this check box to require that users enter a value
    - \*\* Note: The attribute itself does not have to have the Required checkbox selected

# Additional Fields Properties (2)

- Additional properties can be configured within the field (continued):
  - Hidden: Select this check box to prevent the attribute from displaying on user views. Use hidden attributes to add data that is used in calculations but does not display on the page. You must define a default for hidden attributes.
  - Hints / Tooltips: Enter a message that helps the user. The maximum length for a hint is 512 characters. Hints display as a static value above/below the field. Tooltips appear when the user hovers over the field.
  - Height: Enter the number of lines allowed for a text box. For a notes or description type text field choose “Text Area” from “Display Type” and both the height and width of the text box can be set.

# Exercise #4: Fields And Views

- Using the new object do the following:
  - Modify field “ID” to make read-only
    - Navigate to Administration -> Objects -> <Object Name> -> Attributes
    - Select the ID field and open up. Set a default value. Click Save and Return.

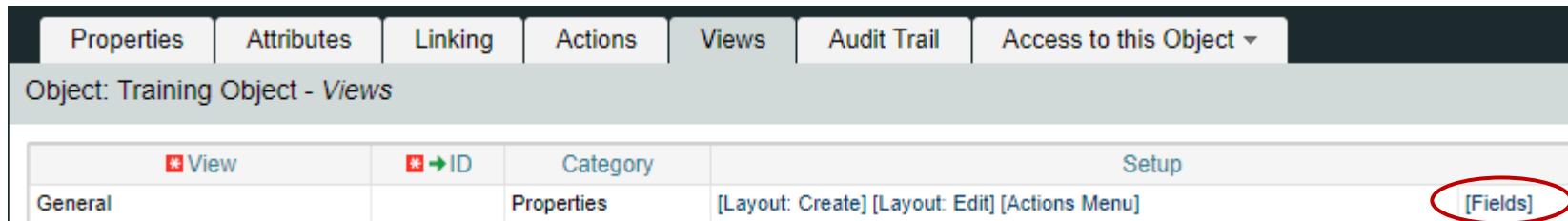
The screenshot shows the 'Attributes' configuration page for the 'Training Object'. The page has two tabs: 'Properties' and 'Auto-numbering'. The 'Properties' tab is active. The page title is 'Object: Training Object | Attribute: ID - Object Attribute'. The main content area displays the configuration for the 'ID' attribute. It includes a red asterisk icon next to 'Attribute Name' with the value 'ID'. Below that, a green asterisk icon with a right arrow is next to 'Attribute ID' with the value 'code'. Further down, the 'Description' field is empty. The 'Data Type' is set to 'String'. The 'Default Value' is set to 'TRN00000', which is circled in red.

Attribute Name	ID
Attribute ID	code
Description	
Data Type	String
Default Value	TRN00000

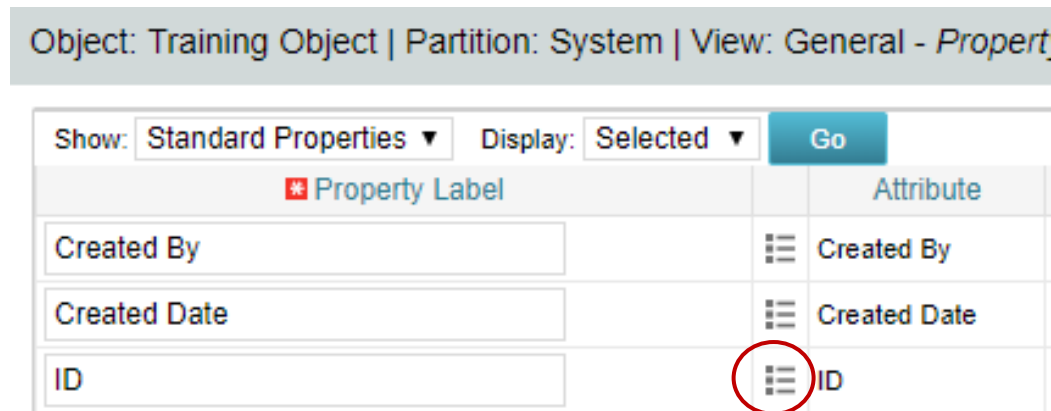
- Select Views

# Exercise #4: Fields And Views (2)

- Using the new object do the following:
  - From the General -> Properties line click on “Fields”




- Click on the properties icon next to the “ID” field



# Exercise #4: Fields And Views (3)

- Using the new object do the following:
  - Select the “Hidden” checkbox

Object: Training Object | Partition: System | View: General - Property Field



Attribute	code
Data Type	String
Property Label	ID 
Display Type	Text Entry ▼
Hint	
Hint Position	Below ▼
Tooltip	
Attribute Default	TRN00000
Override Default Value	
Width	20
Value Required	<input checked="" type="checkbox"/>
Enter Once	<input type="checkbox"/>
Hidden	<input type="checkbox"/>

( In order to make a property field hidden a default must be selected. )

# Exercise #4: Fields And Views (4)

- Make one of the new attributes created “Required” within the field properties
  - Navigate to Administration -> Objects -> <Object Name> -> Attributes
  - Select a field (newly created) and open up properties.
  - Select the “Value Required” checkbox and click “Save”.

Object: Training Object | Attribute: Description - Object Attribute

Attribute Name	Description	 
Attribute ID	v_desc	
Description	<input type="text"/>	
Data Type	String	
Default Value	<input type="text"/>	
Maximum Size	2000	( The maximum size is 2000. For 3 byte Unicode the actual maximum size is 1333. )
Populate Null Values with the Default	<input type="checkbox"/>	
Value Required	<input checked="" type="checkbox"/>	

# Exercise #4: Fields And Views (5)

- Add a hint for entering one of the values and place it below the field
  - Navigate to Administration -> Objects -> <Object Name> -> Views
  - From the General -> Properties line click on “Fields”
  - Choose a field that you want to add some verbiage to help the user when entering information.
  - Add verbiage to the “Hint” text box and choose a position

Object: Training Object | Partition: System | View: General - Property Field

Attribute	name
Data Type	String
* Property Label	<input type="text" value="Name"/>
Display Type	<input type="text" value="Text Entry"/>
Hint	<input type="text" value="Enter the name"/>
Hint Position	<input type="text" value="Below"/>



# Objects: Actions

- Object actions are individual operations that can be selected to be done from either the list or properties view within an object instance
  - Examples of actions are the ability to run a report (only Business Objects), initiate a process instance, copy an object instance, etc.
- Each object has some default actions available to them
- To utilize an action the action menu needs to be configured
  - Within an object this is located within the “Views” tab and by clicking on “Actions Menu” for the specific view being configured
- Both menus and actions can be renamed to fit the business need

# Exercise #5: Action Menu

- Using the new sub-object do the following:
- Add the “New Action Item” action to the “Actions Menu” on the list page
  - Navigate to Administration -> Objects -> <Object Name> -> Views
  - Select “Actions Menu” from the “XXX List” view

Object: Training Object - Views

* View	* → ID	Category	Setup
General		Properties	[Layout: Create] [Layout: Edit] [Actions Menu]
Training Object List		List Column	[Layout] [Options] [Aggregation] [Actions Menu]

- Use the existing “General” menu or add a new menu that represents the action

# Exercise #5: Action Menu (2)

- Using the new sub-object do the following:
- Add the “New Action Item” action to the “Actions Menu” on the list page
  - Move the “New Action Item” option over to the “Selected Actions” column
  - Click Save and Return

Actions Menu - General

Menu Name: General

Menu Code: general

Description: General

Available Actions	Selected Actions
New Add-In Lookup Mapping	New Action Item
New Department	
New Feature	
New GL Account	
New Location	
New Lookup Mapping	
New Project	
New Resource	
New Training Object	
Portfolio	

# Basic SQL

What is SQL?

Using SQL within CA PPM

Basic SQL Syntax

SQL Concepts

What Else Can SQL do?

*rego*University 2018

# What Is SQL?

- SQL stands for Structured Query Language
- SQL is an ANSI (American National Standards Institute) standard
- SQL is semantically easy to understand and learn
- SQL lets you access and manipulate databases and is great for performing the types of analysis and aggregations normally done in Excel
- SQL allows you to traverse much larger datasets and on multiple tables at the same time

# What Is a Database?

- A database is an organized collection of data
- Tables are part of what makes up a database and are similar to the layout of spreadsheets
- Tables are more formalized inside a database with each column having a unique identifier as its heading
- Within databases, tables are organized in schemas
- Schemas are defined by usernames, whereas all of the tables related to that schema will be loaded under it
  - E.g. CA PPM uses NIKU as the default schema in which all of the related tables reside

# SQL Types

- SQL is broken down is as follows:
  - Data Manipulation Language (DML)
    - Used to work with the data stored in the database
    - I.e. Select/Update/Insert/Delete statements
  - Data Definition Language (DDL)
    - Used to build and modify the tables (and other objects) in the database
    - I.e. Create/Drop/Alter/Rename table statements
  - Data Control Language (DCL)
    - Used to administer privileges within the database
    - i.e. Grant / Revoke
  - Transaction Control (TCL)
    - Used to manage the changes made by DML statements
    - i.e. Commit/Rollback

# Using SQL With CA PPM

- SQL is used in multiple facets within CA PPM:
  - To extract ad-hoc data
  - As a basis for NSQL in portlet writing
  - To get data within a process for data manipulation
- CA PPM Data Model
  - Knowing the CA PPM data model is half the battle to grabbing the data you need
  - Three main areas where data is stored:
    - Core Tables (Real Time): Investment, Resource, Timesheet
    - Time Slice Tables: Houses summarize data by daily, weekly, monthly, etc.
    - DataMart Tables: Provides summary and rollup data



# Basic SQL Syntax

- SQL is NOT case sensitive; SELECT is the same as select
- Some database systems (Oracle) require a semicolon at the end of each SQL statement
- There are two required ingredients in any SQL query: a SELECT statement and a FROM statement—and they have to be in that order
  - SELECT indicates which columns you'd like to view, and FROM identifies the table that they live in
- Column names should be separated by commas in the query
- If you want to select every column in a table, you can use \* instead of the column names

# SQL Syntax (2)

- The following statements will extract all rows from a table based on the columns selected:
- `SELECT column_name,column_name  
FROM table_name;`
- `SELECT * FROM table_name;`

# SQL Syntax (3)

- To further limit the results returned from a query use the WHERE clause
  - `SELECT column_name,column_name`  
`FROM table_name`  
`WHERE column_name operator value;`
- Operators in the WHERE clause

Operator	Description
=	Equal
<>	Not equal. <b>Note:</b> In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

# Exercise #1: Basic Query

- Write a select query to pull all the resources from the system and their associated user name

- a. Start with a select from the resource table

```
select r.full_name  
from srm_resources r
```

- b. Add a join to the users table

```
select r.full_name  
from srm_resources r  
join cmn_sec_users u ON u.id = r.user_id
```

- c. Add the user name column from the users table

```
select r.full_name, u.user_name  
from srm_resources r  
join cmn_sec_users u ON u.id = r.user_id
```

# SQL Concepts

- Table Aliases

- Improve readability of SQL queries
  - Use meaningful table aliases
- Allows queries to be easily created/modified
- Improves performance by eliminating the need for the database to search the tables for the reference column.

- E.g.

```
SELECT column_name,column_name  
FROM table_name tbl
```

- In the above example “tbl” is the alias and will be used to reference the table, “table\_name” throughout the rest of the query

# SQL Concepts (2)

- Column Names

- Make your results more presentable by renaming your columns
- The easiest way to rename a column is without spaces using something like an underscore
- If you want to rename a column using spaces you must enclose the name in double quotes (""")
- E.g.

```
SELECT user_name as "User Name", email as "Email Address"  
FROM cmn_sec_users
```

# SQL Concepts (3)

- Joins

- Clauses used to combine rows from two or more tables, based on common fields between them.
- Types
  - INNER JOIN: Returns all rows when there is at least one match in BOTH tables
  - LEFT JOIN: Return all rows from the left table, and the matched rows from the right table
  - RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table
  - FULL JOIN: Return all rows when there is a match in ONE of the tables

# SQL Concepts (4)

- **DISTINCT Statement**

- In a table, a column may contain many duplicate values; and sometimes you only want to list the different (distinct) values
- The SELECT DISTINCT statement is used to return only distinct (different) values.

Syntax:

```
SELECT DISTINCT column_name,column_name  
FROM table_name;
```

- **ORDER BY Statement**

- The ORDER BY keyword is used to sort the result-set

Syntax:

```
SELECT column_name, column_name  
FROM table_name  
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```



# SQL Concepts (5)

- Functions

- Used to perform processing on string and numeric data
- Types
  - Aggregate
    - Return a single value, calculated from values in a column
      - I.e. AVG (Average), COUNT, MAX, MIN, SUM
    - GROUP BY statements are required when using aggregate functions
  - Scalar
    - Return a single value, based on the input value
      - I.e. ROUND, UCASE (Uppercase), LCASE (Lowercase), FORMAT

# SQL Concepts (6)

- Sub-Queries
  - Used to return data that would be used in the main query
  - Nested inside SELECT, INSERT, UPDATE or DELETE statements.
    - Must be enclosed by parentheses

# Exercise #2: Query Using Concepts

- Write a query to pull all active projects starting in 2015 with a count of their tasks.
- Use aliases for table names and rename columns to be meaningful
  - a. Start with a select from the investment table to pull all investments (projects, ideas, other, etc.)

```
select inv.name, inv.code  
from inv_investments inv
```

- b. Add a “WHERE” clause to restrict the result set to active projects only and those that begin in 2015

```
select inv.name, inv.code  
from inv_investments inv  
where inv.odf_object_code = 'project'  
and inv.is_active = 1  
and inv.schedule_start >= to_char('2015-01-01','yyyy-mm-dd')
```

# Exercise #2: Query Using Concepts (2)

- c. Add a join to the investments table to get the tasks

\*\* Note the join will be a left join as we want all projects and just a count of tasks where they exist on the project

```
select inv.name, inv.code, count(t.prid)
from inv_investments inv
left join prtask t ON t.prprojectid = inv.id
where inv.odf_object_code = 'project'
and inv.is_active = 1
and inv.schedule_start >= to_char('2015-01-01','yyyy-mm-dd')
group by inv.name, inv.code
```

# Exercise #2: Query Using Concepts (3)

d. Add column names to make the results meaningful

```
select inv.name as "Project Name", inv.code as "Project ID", count(t.prid) as "Task Count"  
from inv_investments inv  
left join prtask t ON t.prprojectid = inv.id  
where inv.odf_object_code = 'project'  
and inv.is_active = 1  
and inv.schedule_start >= to_char('2015-01-01','yyyy-mm-dd')  
group by inv.name, inv.code
```

# What Else Can SQL Do?

- SQL can insert records into a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables and views in a database
- SQL can create stored procedures in a database
- SQL can set permissions on tables, procedures, and views

# Portlets, Queries, Lookups

*rego*University 2018

# Lookups

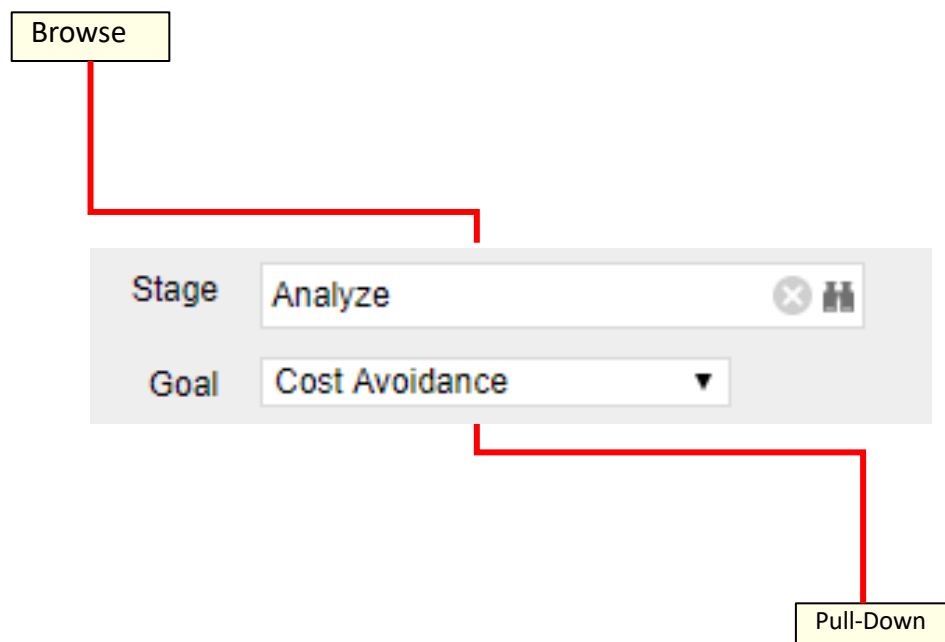
- What is a Lookup?
  - When attached to a field, a lookup allows users to select pre-defined values from a pull-down or browse list.
  - The lookup field's choices can be static values entered by an administrator, or dynamic values returned from a database query
  - Lookups values can be displayed as text or icons
  - 3 Source Types
    - Static List
    - Static Dependent List
    - Dynamic Query



# Lookups (2)

- Static List Lookup

- Use this type of lookup when working with a standard set of values. Static list lookups are often used as pull-down/browse lists for object fields, portlets/reports, and custom forms.



# Exercise #1: Static Lookups

- Create a static list lookup containing the values Yes/No that we can use later during this training.
  - Navigate to Administration → Data Administration → Lookups
  - Create a new static list lookup using the below details
    - Name: Yes and No
    - Source: Static List
    - Values: (corresponding id):
      - No (0)
      - Yes (1)

# Lookups (3)

- Static Dependent Lookup
  - Use this type of lookup to create a hierarchy of lookups and values.
  - Items that appear on the second and subsequent lists depend upon choices previously made by the user.
    - For example, if the user selects “USA” from a country browse list, then a state list may appear from which the user can select an appropriate state.

# Lookups (4)

- Dynamic Query Lookup
  - Use this type of lookup to capture data from the CA PPM database in real time to populate the drop-down or browse lists. Dynamic Queries
  - These lookups provide the most up-to-date values possible and are often used inside browse windows.
  - Dynamic queries should only be written in NSQL
    - NSQL is a version of SQL specific to CA PPM; it is used to designate query segments as metric values, dimensions, dimension properties, or parameters
    - NSQL can only select data from a database; it cannot update data

# Exercise #2: Dynamic Query Lookups

- Create a Dynamic NSQL Query to pull basic information for all investments in the system.
  - Navigate to Administration → Data Administration → Lookups
  - Create a new Query using the below details
    - Name: All Investments
    - Content Source: Dynamic Query
    - Query:

```
SELECT
@SELECT:code:Investment_ID@,
@SELECT:name:Investment_name@
FROM inv_investments
WHERE @FILTER@
```
  - Display Attribute: Investment\_name
  - Hidden Key: Investment\_ID

# Queries

# Queries

- What are Queries?
  - Queries extract data from the system for use in portlets.
  - Similarly to Dynamic Lookup, Queries in CA PPM are also written in NSQL
    - NSQL is a version of SQL specific to CA PPM; it is used to designate query segments as metric values, dimensions, dimension properties, or parameters
    - NSQL can only select data from a database; it cannot update data

# Queries (2)

- The **SELECT** statement retrieves column data from tables
  - NSQL Queries must start with SELECT however for each column a @SELECT@ tag must be used.
- A dimension is a grouping of similar data elements from one or more tables
  - Defining Dimensions
    - <Dimension> is a user-defined name for the dimension
    - <Table.Field> is the table or alias name retrieved in the FROM statement
    - <label> is the name you want to appear in the column list in clarity

## *SELECT*

*@SELECT:DIM:USER\_DEF:IMPLIED:<Dimension>:<Table.Field>:<label>@*

- DIM: Indicates the line is the primary key for the dimension
- There can only be one DIM to each dimension.

*@SELECT:DIM\_PROP:USER\_DEF:IMPLIED:<Dimension>:<Table.Field>:<label>@*

- DIM\_PROP: Indicates columns for the dimension
- There can be many DIM\_PROPS defined to one dimension.



# Queries (3)

- The **FROM** clause is a standard SQL statement which defines which table to gather data from

*FROM <Table>*

- The **WHERE** statement filters data returned by a query to be used on portlets
  - The @FILTER@ statement is required and allows the system to filter the values defined with the @SELECT@ tag

*WHERE <Condition>*

*AND @FILTER@*

- The **GROUP BY** clause is typically used to combine database records with identical values in a specified field into a single record, usually for the purposes of calculating some sort of aggregate function
- The syntax for the **HAVING** statement is @HAVING\_FILTER@ which can be used when a query uses metrics
  - The Developer guide states this is required but it is **NOT**.

# Exercise #3: NSQL Queries

- Create a new Query (nsql) to pull basic investment information
  - a. Navigate to Administration → Studio → Queries
  - b. Create a new Query using the below NSQL statement to extract the data
    - a. Name: Investment Details Query
    - b. Query:
 

```
SELECT
@SELECT:DIM:USER_DEF:IMPLIED:INVESTMENT:code:id@,
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:name:name@,
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:schedule_start :start_date@,
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:schedule_finish :finish_date@,
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:is_active :is_active@

FROM inv_investments

WHERE @filter@
HAVING @HAVING_FILTER@
```
  - c. Associate the two of the attributes with the two lookups created in the earlier exercises.

Attribute Name	Associated Lookup
id	All Investments
Is_active	Yes and No

# Portlets

# Portlets

- What is a Portlet?
  - Portlets are snapshots of CA PPM data and can consist of grids, graphs, or snippets of HTML
  - Portlets do not replace CA PPM reports, but can be considered mini-reports
  - Portlets obtain information and business intelligence from CA PPM, other databases within the enterprise, and external sources available in HTML (for example, business news and network status information)
  - Users can populate Portlets with graphs, tables, workflows, best practices, documents, and forms and have the information update in real-time without running a report

# Portlets (2)

- **Chart Portlet** – A graphical view of CA PPM data (for example, pie and line charts)
- **Grid Portlet** – A list or table of data you can filter in real time. Can be sourced by Objects or Queries.
- **HTML Portlet** – Displays information on a CA PPM page from internal or external web sites formatted as HTML
- **Filter Portlet** – Applies a common filter to all Portlets on a page
- **Interactive Portlet** – Displays visually rich, real-time CA PPM data using imported Xcelsius visualizations

# Portlets (3)

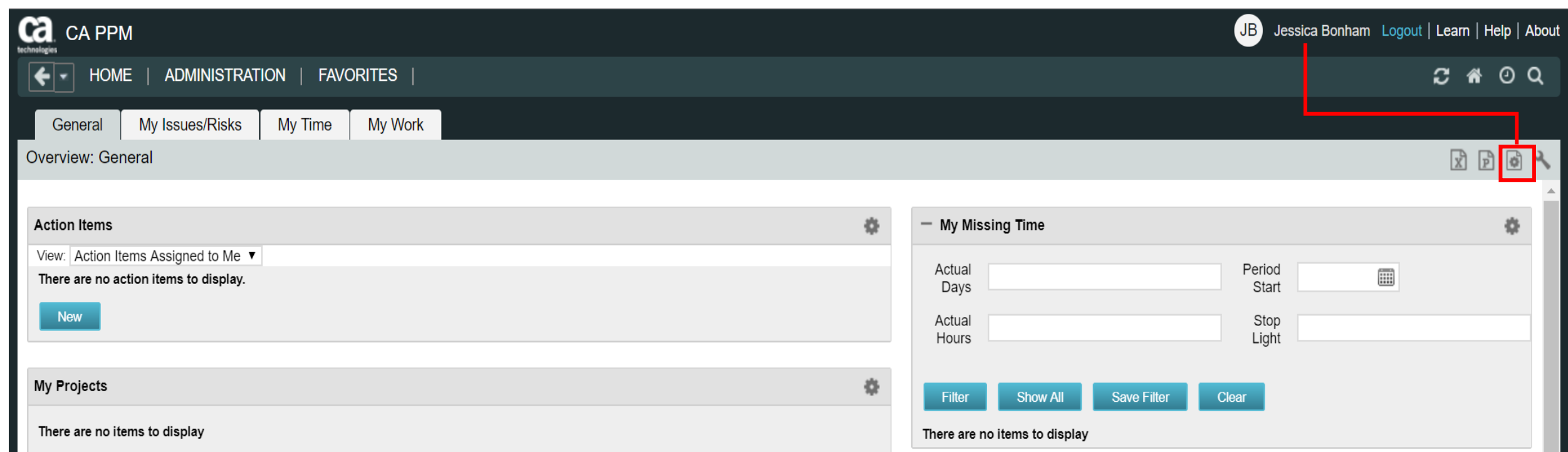
- NSQL
  - Portlets created using queries in Clarity to define the data that can be used.
  - Pros
    - Logic
    - Parameters
    - Security
    - Customizable
    - Matrices
  - Cons
    - Not Dynamic
    - Development Time
    - No In-Line Editing
- Object Based
  - Portlets created using an Object instead of a query to define the data set gathered.
  - Pros
    - Customizable
    - Security
    - In-Line Editing
    - Dynamic
    - Time Scaled Values
  - Cons
    - Multiple Objects Difficulty
    - No Custom Logic

# Exercise #4: Basic Grid Portlets

- Create a portlet that displays basic information for all investments in the system.
  - Navigate to Administration → Studio → Portlets
  - Create a new Grid Portlet using the below details
    - Name: Investment Details
    - Data Provider: Investments Details Query
    - List Layout:
      - ID, Name, Start Date, Finish Date
    - Filter Layout:
      - ID, Is Active?

# Adding a Portlet

- Use the Add button on the toolbar at the far right to add additional content to your page.



The screenshot displays the CA PPM user interface. At the top, the logo 'ca technologies' and 'CA PPM' are visible on the left, and the user 'JB Jessica Bonham' with links for 'Logout', 'Learn', 'Help', and 'About' is on the right. Below the header is a navigation bar with 'HOME', 'ADMINISTRATION', and 'FAVORITES'. A secondary bar contains tabs for 'General', 'My Issues/Risks', 'My Time', and 'My Work'. The main content area is titled 'Overview: General'. On the right side of this area, a toolbar contains icons for document, print, and a circled 'Add' icon (a square with a plus sign). A red line originates from the 'Add' button and points to the 'My Missing Time' portlet. The 'Action Items' portlet on the left shows 'View: Action Items Assigned to Me' and 'There are no action items to display.' with a 'New' button. The 'My Projects' portlet at the bottom left also shows 'There are no items to display'. The 'My Missing Time' portlet on the right includes input fields for 'Actual Days', 'Actual Hours', 'Period Start', and 'Stop Light', along with 'Filter', 'Show All', 'Save Filter', and 'Clear' buttons. It also displays 'There are no items to display.'



# Adding a Portlet (2)

CA PPM technologies

JB Jessica Bonham Logout | Learn | Help | About

HOME | ADMINISTRATION | FAVORITES

Page Properties | **Content** | Page Filters | Layout

Page: Overview | Tab: General - Page Content

<input type="checkbox"/>	Title ▲	Content	Column	Category	Description	Active
<input type="checkbox"/>	Action Items	Action Items	Left	Collaboration	Action items assigned to or created by you	✓
<input type="checkbox"/>	Communications Portlet - Simple	Communications Portlet - Simple	Right	Business Intelligence	Simple Communications Portlet	✓
<input type="checkbox"/>	Missing Time by Project Manager	Missing Time by Project Manager	Right	Business Intelligence		✓
<input type="checkbox"/>	Missing Time by Resource Manager	Missing Time by Resource Manager	Right	Business Intelligence		✓
<input type="checkbox"/>	My Missing Time	My Missing Time	Right	Business Intelligence		✓
<input type="checkbox"/>	My Projects	My Projects	Left	Project	Favorite projects	✓
<input type="checkbox"/>	Site Links	Site Links	Right	Collaboration	Site Links	✓

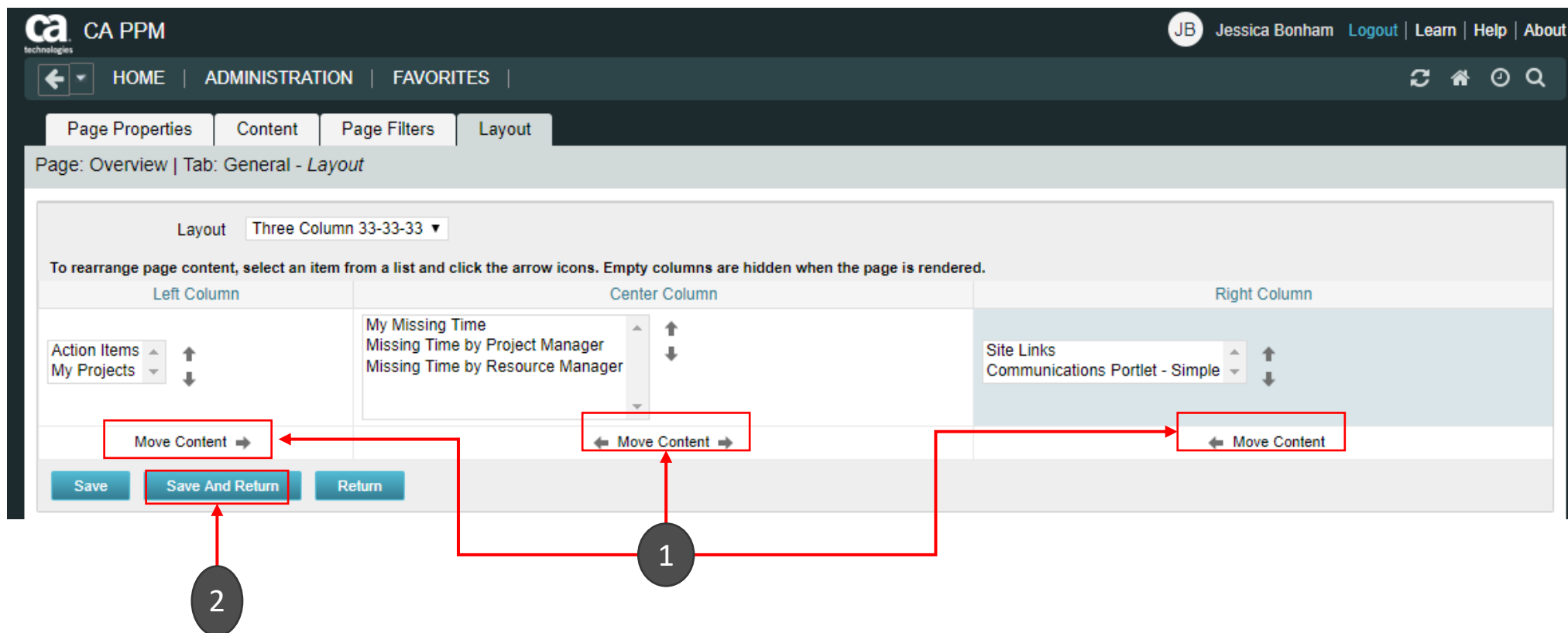
Displaying 1 - 7 of 7

**Add** | ✓ Remove | **Continue** | Restore Defaults | Return

Click the tab where you want the portlet to appear

- 1 Click the *Content* tab
- 2 Click *Add* and browse for the desired portlet
- 3 Click *Continue*

# Adding a Portlet (3)



- 1 Use the *Move Content* arrows to position the portlet on the page
- 2 Click *Save and Return*

# Advanced Basic Processes

*rego*University 2018

# Agenda

---

- Overview
- Design Basics
- Example/Exercise
- Questions

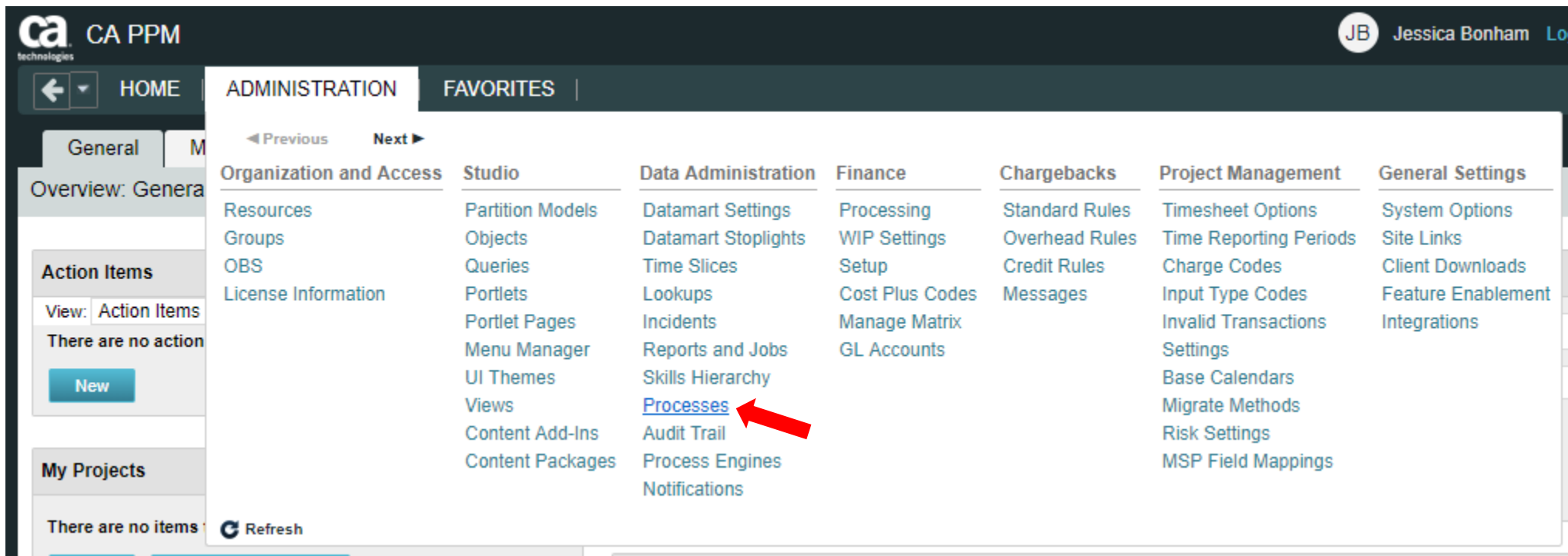
# Overview

# Processes: Overview

- What is a process?
- Processes automate repetitive steps that you would otherwise perform manually through the user interface
  - To accurately reproduce a user action, the process impersonates the process initiator to perform the process steps
  - A process includes a series of steps that result in an end
  - Each step performs one or more actions that move the process toward completion
  - All processes have a start and finish step
  - Processes use pre and post conditions to connect the steps
- CA PPM provides stock processes that you can use to
  - Approve documents
  - Approve timesheets
  - Approve ideas

# Processes – Overview (2)

- Where to access process definitions
  - Administration > Data Administration > Processes



# Design Basics



# Processes: Design Basics

## 1. Create

- a. Define Process Properties
- b. Add Process Objects (optional)
- c. Start Options (object related)
- d. Define Steps
  - 1) Pre-condition(s)
  - 2) Action(s)
  - 3) Post-condition(s)

## 2. Validate / Activate

# Processes: Design Basics (2)

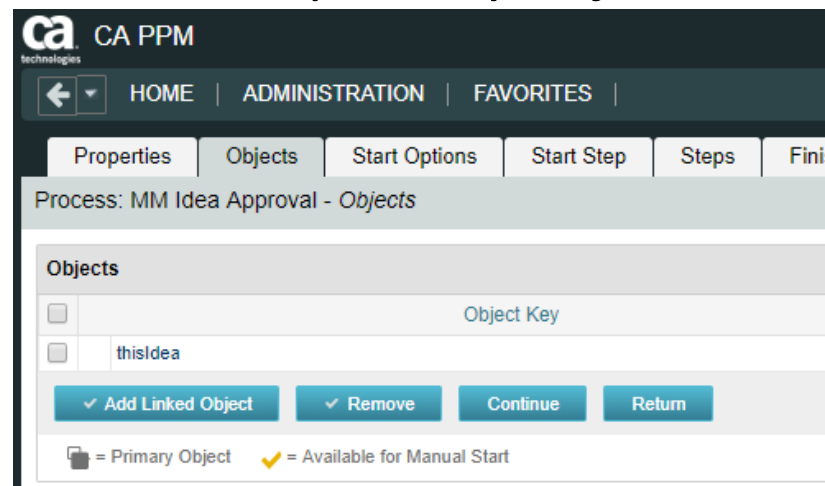
- Define Process Properties
  - Set the name and ID of the process
  - By default the process will be created in “Draft” mode

The screenshot shows the 'Process Definition: Properties' form in the Rego system. The form has a tabbed interface with tabs for Properties, Objects, Start Options, Start Step, Steps, Finish Step, Escalation Defaults, Notifications, Validation, and Process Flow. The 'Properties' tab is active, showing the 'General' section. The form contains the following fields and options:

- Process Name:** MM Idea Approval
- Process ID:** mm\_idea\_approval
- Content Source:** Customer (dropdown menu)
- Description:** (empty text area)
- Status:** Not Validated
- Mode:** Draft (selected), Active, On Hold
- Buttons:** Save and Continue, Save, Save And Return, Return
- Legend:** [Red square] = Required, [Green arrow] = Enter Once, [Green star] = Unique

# Processes: Design Basics (3)

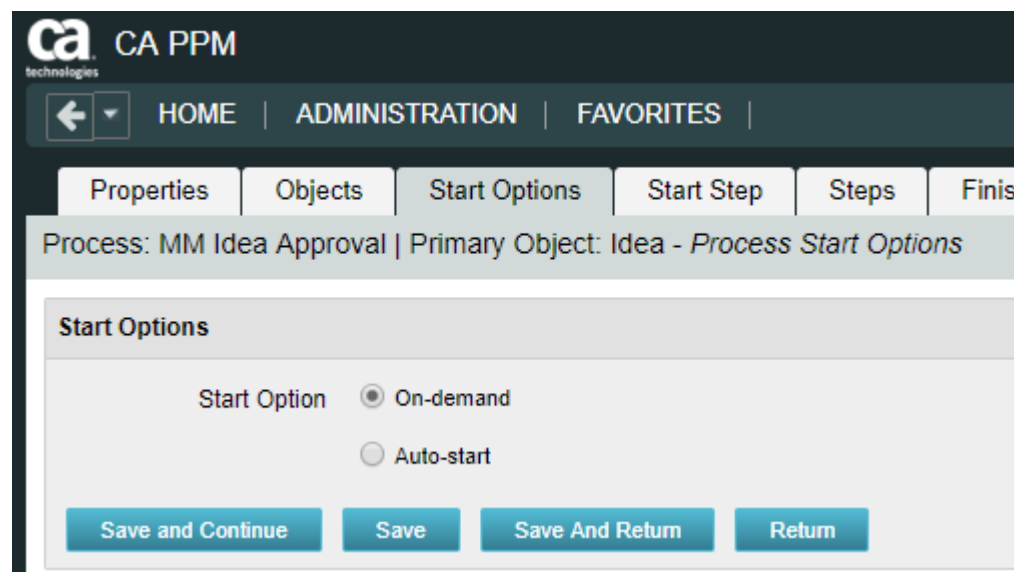
- Add Process Objects
  - Add an object to your process only if ..
    - The process needs to be triggered on a create or update event
    - The process is using manual actions, system actions, or sub-processes within your steps
  - You can add the following types of objects to your processes
    - A primary object
    - One or more linked objects
  - An example of an associated primary object looks like the below



# Processes: Design Basics (4)

- Start Options

- On Demand: A process will only be kicked off manually or by another process calling that process
  - Note: Only a process that is NOT associated with an object can be scheduled by a job
- Auto-Start: A process will be started based off a create or an update event on the object instance



The screenshot shows the 'CA PPM' web interface. The top navigation bar includes a back arrow, 'HOME', 'ADMINISTRATION', and 'FAVORITES'. Below this is a tabbed interface with 'Properties', 'Objects', 'Start Options' (selected), 'Start Step', 'Steps', and 'Finish'. The main content area displays 'Process: MM Idea Approval | Primary Object: Idea - Process Start Options'. Under the 'Start Options' heading, there are two radio buttons: 'On-demand' (selected) and 'Auto-start'. At the bottom, there are four buttons: 'Save and Continue', 'Save', 'Save And Return', and 'Return'.

# Processes: Design Basics (5)

- Define Process Steps
  - Steps comprise the groups of actions that take place inside a process to accomplish a set of activities
  - Process logic is the Pre-Condition or Post-Condition of each step
    - When defining a pre-condition to a step, you can use attributes from multiple objects added to the process
      - Pre-conditions will determine whether the step should begin
  - After defining the pre-conditions that trigger a step, you must define post-conditions that connect this step to the next step or the end step
    - Post-conditions will determine where and whether the step will move
  - Examples of pre and post-conditions include:
    - Checking the status of action items
    - Checking between object attributes values (except for MVL attributes)
    - Waiting for a sub-process to complete before joining the master

# Processes: Design Basics (6)

- Define Process Steps
  - Steps comprise the groups of actions that take place inside a process to accomplish a set of activities
  - Process logic is the Pre-Condition or Post-Condition of each step
    - When defining a pre-condition to a step, you can use attributes from multiple objects added to the process
      - Pre-conditions will determine whether the step should begin
  - After defining the pre-conditions that trigger a step, you must define post-conditions that connect this step to the next step or the end step
    - Post-conditions will determine where and whether the step will move

# Processes: Design Basics (7)

- Examples of pre and post-conditions include:
  - Checking the status of action items
  - Checking between object attributes values (except for MVL attributes)
  - Waiting for a sub-process to complete before joining the master

The screenshot shows the configuration interface for the 'Process: MM Idea Approval - Start Step'. The interface is divided into several tabs: Properties, Objects, Start Options, Start Step, Steps, Finish Step, Escalation Defaults, Notifications, Validation, Process Flow Diagram, and Access to this Process. The 'Start Step' tab is currently selected.

The 'General' section contains fields for Step Name (Start), Step ID (Start), Status (Not Validated), Description, Raise a Warning After (dropdown), and Milestone (checkbox). The 'Referring Steps' section shows 'There are no referring steps to display.' The 'Pre-conditions' section is highlighted with a red box and two red arrows. It shows 'Join Type: None' and 'There is no precondition to display.' with a 'New' button. The 'Post-conditions' section shows 'Split Type: Serial' and 'Post-conditions will be checked in the order they are listed below.' with a table for conditions. The 'Notifications' section includes 'Send Notification' (checkboxes for When step is started, completed, or in error), 'Enter Recipients' (text field), 'Send Notification To' (dropdown), and 'Notify Owner' (checkbox). The 'Escalation' section shows 'There is no escalation rule setup to display.' with a 'New' button. The 'Actions' section shows 'Actions are not set.' with a 'New' button.

# Processes: Design Basics (8)

- Splits / Joins determine the direction of the process flow
  - A Split branches processing into multiple directions
  - A Join consolidates the process flow

## Post Conditions Split

- Serial
- Parallel
- Decision Point
- Multi choice

## Preconditions or Joins

- Rendezvous (AND)
- Merge (XOR)
- Wait and Merge
- Multi-thread
- First in Line



# Processes: Design Basics (9)

- A **Step Action** is a task which a process executes and can be defined as follows:
  - Manual Action
    - Example(s): Actions Items
    - With manual actions you can associate variables with the subject line and message
  - System Action
    - Example(s): Lock/Unlock Attributes, Set Attribute Values, and certain System Operations (copy a financial plan from a template)
  - Running of a Job
    - Example(s): Post Timesheets
  - Custom Script (GEL)
    - Example(s): Notification Scripts
  - Sub-process
    - Sub processes are invoked as embedded processes within the context of the current process

# Processes: Design Basics (10)

103

Properties Objects Start Options Start Step Steps Finish Step Escalation Defaults Notifications Validation Process Flow Diagram Access to this Process

Process: MM Idea Approval - Start Step

**General**

Step Name: Start

Step ID: Start

Status: Not Validated

Description:

Raise a Warning After: [--Select--]

Milestone: ☐

**Referring Steps**

There are no referring steps to display.

**Pre-conditions**

Join Type: None

There is no precondition to display.

**Post-conditions**

Split Type: Serial

Post-conditions will be checked in the order they are listed below.

	If ...	Then Go To	Description
<input type="checkbox"/>	[Build Conditions]	[Select Step]	

**Notifications**

Send Notification: ☐ When step is started  
☐ When step is completed  
☐ When step is in error

Enter Recipients: [Quick Add Recipients]

Send Notification To: [Add Recipients]

Notify Owner: ☐

**Escalation**

There is no escalation rule setup to display.

**Actions**

Actions are not set.

**Buttons:** New, Delete

Process: MM Idea Approval | Step: Start - Select Action Type

	Action Type	Description
<input checked="" type="radio"/>	Manual Action	Send a Action Item to a resource, group, role or profile
<input type="radio"/>	System Action	Perform an operation on a PPM Object
<input type="radio"/>	Run Job	Run a PPM job
<input type="radio"/>	Custom Script	Execute a custom script
<input type="radio"/>	Subprocess	Invoke another process within the context of the current process
<div>NextCancel</div>		

# Processes: Design Basics (11)

- Validate and Activate Process

- Use the process validations page to monitor the latest validation statuses and errors at the step and process level
- Validation rules are used to validate steps and conditions and the structure of a process

Process: MM Idea Approval - Process Validation

Status Re-validation Required

Mode Draft

Validations

	Validation Object	Status
-	Steps	
<input type="checkbox"/>	Start	◆
<input type="checkbox"/>	Finish	◆
<input type="checkbox"/>	Process	◆

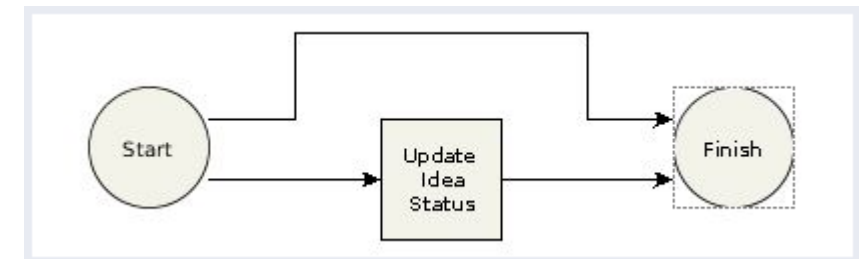
✓ Validate Validate All and Activate Continue Return

◆ = Validated ◆ = Errors Encountered ◆ = Re-validation Required ◆ = Not Validated

# Exercise

# Process: Exercise

1. Create a new process using your initials as an identifier
2. Link the process to the idea object
3. Make the process auto-start on update, with status set to “Submitted for Approval” and prior status not set to “Submitted for Approval”
4. Create one step in addition to the Start and Finish steps
5. Add an action to the new step that sets the idea status to Approved
6. Add a second action to the new step that sends an action item / notification to the idea owner that the idea has been approved
7. Once all steps are created, create links between steps
8. Ensure the conditions point to the correct step
9. Validate and Activate the process
10. Now TRY IT OUT!! 😊



# Advanced UI Themes, Menus, Options, Financials

*rego*University 2018

# Agenda

- User Interface Themes
- Menu Configuration
- System Options
- Financial Administration

# User Interface (UI) Themes

- What is a UI theme?
  - Determines the look and feel of CA PPM
  - Can be adjusted or extended
  - Can be different per partition
  - Created using cascading style sheets (CSS)
- Create your own theme:
  1. Copy CSS from stock theme
  2. Modify
    - Main page background color (3366AA)
    - Logo
      - Convert to base64 string – free tools on web
      - Replace string on CCS for The logo in the upper left hand corner...
    - Remove a button from a custom object view



# UI Demonstration / Exercise

- Modify the UI Theme to change the background color
  - Navigate to Administration -> Studio -> UI Themes
  - Pick an existing UI theme to copy from
    - Open and copy CSS
  - Select “New” and give your UI theme a name and ID
  - Paste the CSS into the CSS field
  - Change the following snippet to the hexadecimal # representing the color you wish to change it to

```
/* The main page background color */  
.ppm_page_bg {  
  background: #3366AA;  
  background: -webkit-gradient(radial,50% 10, 1, 50% 100, 600, from(#052E5F), to(#3366AA ));  
  background: -moz-radial-gradient(50% 10% 0deg,ellipse contain, #052E5F, #3366AA);  
}
```

# UI Demonstration / Exercise (2)

- Apply new UI Themes
  - Navigate to Administration -> Studio -> Partition Models
  - Click the name of the partition you want to view
  - Select the “Partitions” tab
  - Click on the properties icon next to the name of the partition to which you want to set a UI theme
  - In the UI Theme field select the new UI Theme created

# Menu

- Menus are utilized to assist in simplifying the organization and accessibility of information for users, based on their organizational role
- Two Main Menu Areas
  - Home (Application )
  - Administration
- Consists of:
  - Links to a Page or Action
  - Sections
- Menus are setup without regard to security
  - User will only see what they have access to
- Changes to the menu require a refresh or logout of the existing session

# Menu Demonstration / Exercise

- Modify the Home menu by adding a new section called “Rego U”
  - Navigate to Administration -> Studio -> Menu Manager
  - Select Application Menu
  - Click on the “Add” button and select “Section”
  - Name the section “Rego U-<your initials>” with an ID = “regou\_<your initials>”
  - Click Save and Return

# Menu Demonstration / Exercise (2)

- Move the XXX link from the X section to the new “Rego U” section
  - Navigate to Administration -> Studio -> Menu Manager
  - Select Application Menu
  - Click on the checkbox next to the link you want to move
  - Select “Move”
  - Select the radio button next to the new section created
  - Click “Save and Return”

# System Options

- The following areas within the Administration menu should be set up and configured to fit your organization's needs
  - Data Administration
    - Datamart Settings
      - Setup is required for successful execution of the Datamart extraction job; prerequisites for this setup include defining an entity and a department and location OBS
    - Time Slices
      - A time slice is a flat table that contains data derived from a sliced binary large object (BLOB)
      - Configure a time slice request to specify the objects and fields, the slicing frequency, and the granularity to store the data
      - The Time Slicing job extracts data and displays a readable flat table based upon the criteria set
  - Finance
    - Processing: Standardize functionality or business logic by setting up basic rules for financial transaction processing

# System Options (2)

- Project Management
  - Timesheet Options: Complete in order for staff members to track time and for managers to view and approve time
    - Choose included fields, range of days in which to populate timesheets, whether to use indirect time, etc.
  - Settings
    - Set options like work effort display unit, allowing of effort task creation, automatically opening team members for time entry, etc.
  - Risk Settings
    - Set probabilities based on risk impact
- General – System Options
  - Apply to CA PPM as a whole
  - Used to set things like password rules, logout timeout, chart colors, company name, etc.

# System Options Discussion

Discuss the system options



# Financials Best Practices

- Keep as simple as possible
- Do invest time to develop the right architecture
  - Include PMO, finance, PM's
  - Start with desired outputs and work backwards to build architecture
- CA PPM is not the financial system of record
- CA PPM is not an accounting tool
- Provide enough financial information to make decisions

# Financials Basic Requirements

- Entity
  - Financial reference for investments
- Fiscal Time Periods
  - Weekly, Semi-Monthly, Monthly, Quarterly, Annually, 13 Periods
  - Different from timesheet periods (usually)
- Department OBS – A department represents units in the organizational structure of the company; Keep simple
- Location OBS – A location represents a geographical location or department where a company conducts its business; they are linked to departments
- Rate Matrix – Used to determine cost and billing during financial processing
  - Assigned columns identify the criteria used to match rates and costs to transactions
- Jobs – OOTB jobs are used to process transactions

# Financials Classifications

- The financial output you desire drives the CA PPM classifications you will use:
  - Resource Classes: Categorize resources by organizational, geographical, or skill level
  - Company Classes: Categorize clients or lines of business within your organization
    - E.g. Industry: Government, Education, Technology
  - Work In Process (WIP) Classes: Categorize companies and investments
  - Investment Classes: Categorize work logically within an organization
  - Transaction Classes: User defined values that group transaction types
    - Categorize labor transactions using values such as Development, Consulting, Sales
  - Input Type Codes: Represents a breakdown of work associated with resources
    - E.g. Regular time vs. Overtime
  - Cost Type: Set at investment and task level; choose from Capital or Operating
  - Charge Code: Represents a breakdown of work associated with investments
    - E.g. Capital, Expense or non-project time

# Financial Jobs

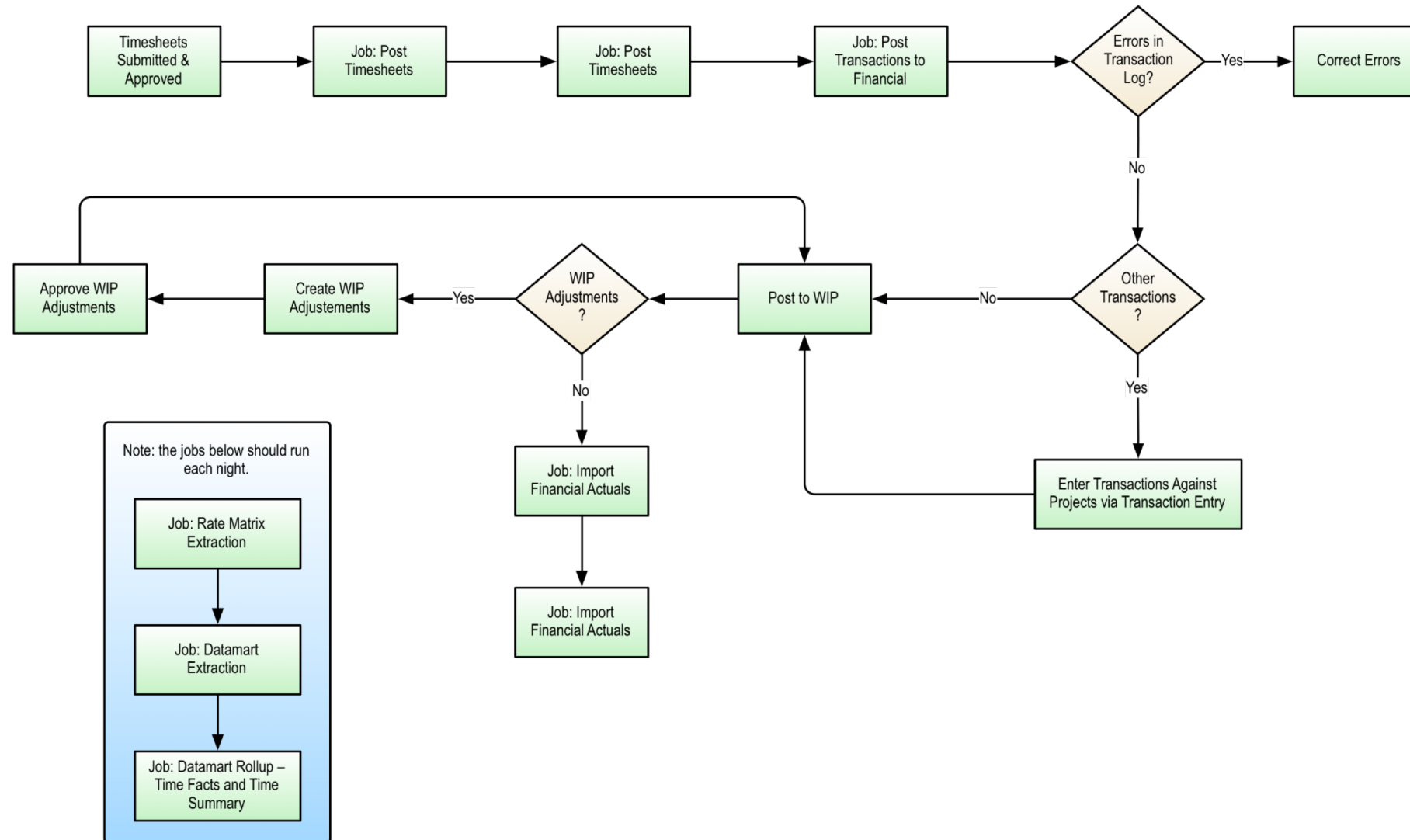
1. Post Timesheet
2. Post Transactions to Financials
  - Review and fix invalids
3. Post to WIP
4. Import Financial Actuals

## Overnight Jobs

- Rate Matrix Extraction
- Datamart Extraction
- Datamart Rollup – Time Facts and Time Summary

# Financial Flow

122



# Rate Matrix

- Determines rates for transactions based on the financial classifications you want to use
- CA PPM supports three levels of matrices and will look at the levels in order to determine a rate:
  1. Investment
  2. Entity
  3. System
- Rate Matrix Setup:
  1. Create matrix
  2. Assigned classifications (columns)
  3. Add rate criteria

# Rate Matrix Demonstration / Exercise

- Create a new rate matrix and add the role, resource, and resource class fields
  - Navigate to Administration -> Finance -> Manage Matrix
  - Click on “New” and give the matrix a name
  - Click on “Save and Continue”
  - Select the Role, Resource, and Resource class fields and move to “Selected” column
  - Click “Save and Continue”
  - The matrix is now ready to have rows entered

# Manual Transaction

- There are two steps for creating manual transactions
  1. Create a voucher
    - Vouchers can be for Expense or Other (which includes all other transaction types)
    - You can only use expense resources on an expense voucher
    - You can link vouchers to a PO number or a specific vendor
  2. Create transactions
    - There can be multiple transactions per voucher, but all must be the same type (Expense or Other)



# Manual Transaction Demo / Exercise

- Enter a manual labor transaction for a 10 hour quantity and a rate of 120
  - Navigate to Home -> Financial Management -> Transaction Entry
  - Click on “New”
  - Create the voucher by selecting “Voucher Other” in the Entry type and enter an Entry Number
  - Click “Save”
  - Click “New” to enter a transaction and fill out the required fields and any other necessary fields desired
    - Required fields for a transaction are Transaction Date, Investment ID, Task, Charge Code, Resource ID, Transaction Class, Input Type Code, and Quantity

# Manual Transaction Demo / Exercise

- Enter 10 into the Quantity field
- Enter 120 into the Cost and Rate fields

Transaction Details

☒ Transaction Date

☒ Investment ID PR000001

☒ Task Database Development

Cost Type Capital

☒ Charge Code Develop

☒ Resource ID jbonham

Role

☒ Transaction Class Person

☒ Input Type Code Regular

User Value 1

User Value 2

Expense Type [--Select--]

Notes

Preserve General Information ☐ Select this to preserve general information when you click Submit and Create New.

Transaction Data

☒ Quantity 10

Cost 120  USD

Rate 120  USD

Chargeable ☒

☒ = Required



XOG

*rego*University 2018

# XOG Overview

- Introduction
- Requirements
- Installation and Setup
- XOG Files and Procedure
  - Configuration – (Query, Portlet, Workflow, etc)
  - Web version (13.1 and below)
  - Data
- Best Practices
- Other XOG Methods and XML Creation

# Introduction

- CA PPM has a Web Services interface called XML Open Gateway (XOG)
- What XOG Does
  - Export data and configuration
  - Import data and configuration
- When to Use XOG
  - Move configuration or data from one environment to another
  - Handle data imports via batch processing

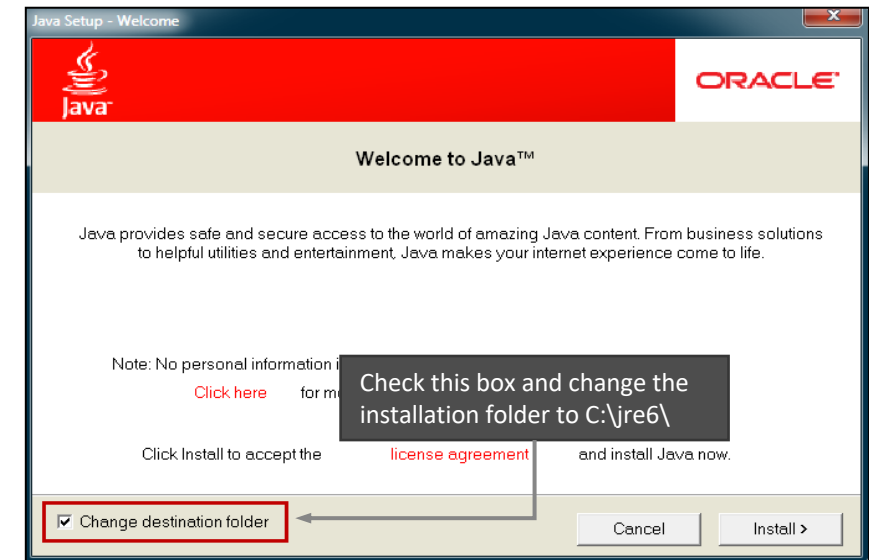
# Requirements

- Operating System
  - Microsoft Windows XP Pro or later (32 or 64 bit)
  - Mac OS X 10.4 or later
  - Linux
- Java Runtime Environment (JRE)
  - Oracle Java 6 Runtime Environment version 1.6.0\_15
  - Oracle Java 7 Runtime Environment version 1.7.0\_25
- CA PPM
- XOG client version that matching the CA PPM
- CA PPM user with XOG Global Rights
  - Administration – XOG
  - Administration – Access
  - XOG rights for individual objects (for example, Resource, Project, OBS)

# Installation and Setup

# Installation

- Download compatible JRE from <http://www.java.com/en/>
  - Installation folder: C:\jre6\
- Set the environment variables:
  - **JAVA\_HOME**=C:\jre6
  - **PATH**=;%JAVA\_HOME%\bin
- Test for Java using a command prompt
  - java -version



```
C:\>Administrator: C:\Windows\system32\cmd.exe

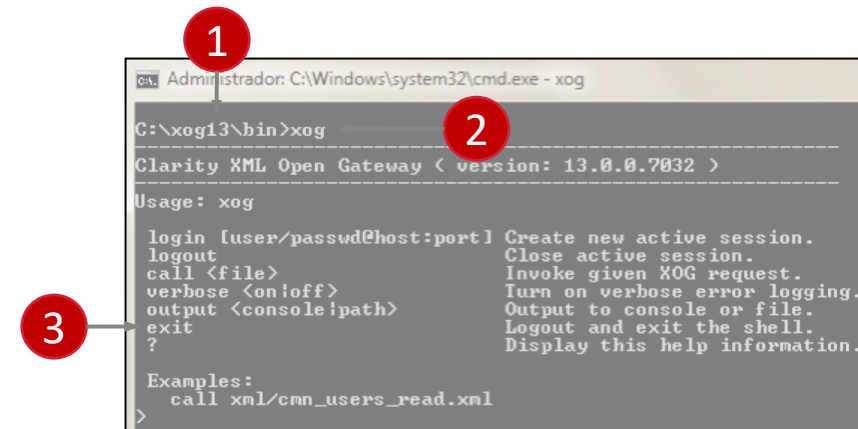
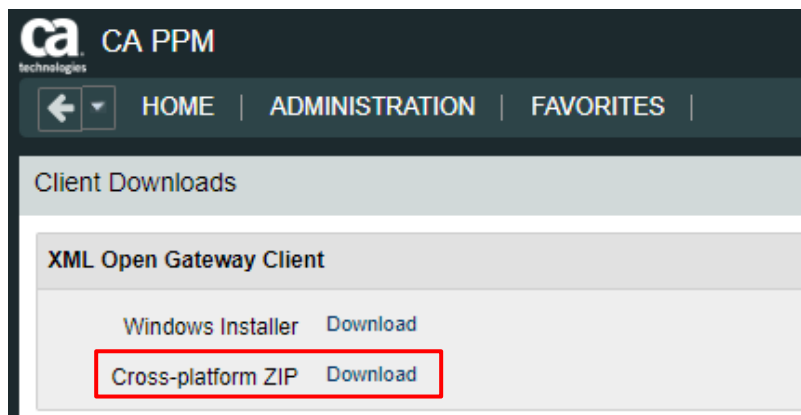
C:\>java -version
java version "1.6.0_31"
Java(TM) SE Runtime Environment (build 1.6.0_31-b05)
Java HotSpot(TM) Client VM (build 20.6-b01, mixed mode, sharing)

C:\>
```



# Installation Cont.

- Download the XOG client from the Admin tool
  - Download the cross-platform client
  - Extract the ZIP file to C:\xog13\
- Test for XOG client using a command prompt
  1. Access C:\xog13\bin\ folder
  2. Type xog and press enter
  3. Type exit to quit



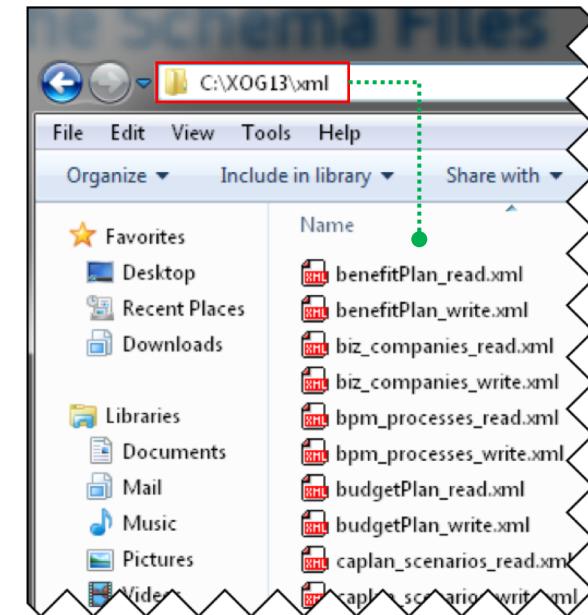
# Verify Connectivity

- Use XOG client shell commands to test the connection between the XOG client and CA PPM server as follows:
  1. Open a command prompt
  2. Type `cd C:\xog13\bin\` and press *enter*
  3. Type `xog (xog -sslenabled true` if your connection is using HTTPS) and press *enter*
  4. Type `login <username>/<mypassword>@<myserver>:<port>` and press *enter*  
For example, if your username was rodmi03, your password was Niku2000, and your CA PPM instance was `https://cppm1234-dev.ondemand.ca.com/niku` on port 443, you would type:  
`login rodmi03/Niku2000@cppm1234-dev.ondemand.ca.com:443`
  5. Verify login succeeded

# XOG Files and Procedure

# XML Files

- The XML files are valid examples of read and write requests that can run using the XOG client
- There are XML files for each CA PPM object you can manipulate with XOG (for example, Resource, Project, Group)
- XML files come in pairs
  - Read (Export)
  - Write (Import)
- Access the XML files from the XOG client installation folder (C:\xog13\xml)



# The XML Read Files

- Use the XML Read files to export a specific item from CA PPM
- Each Read XML file contains the following structure:
  - **Header:** Supported CA PPM version, Operation (Read), and the object (Resource, Project, etc.)
  - **Arguments:** The type of information associated to the object to be included in the export (for example, include tasks and team members for projects)
  - **Query filters:** Limit the export data to (for example, Export only Project-A23 and Project-B89)

# The XML Read Files (2)

The Query Filter section supports criteria values to limit the scope of the export and accepts EQUALS, OR, BETWEEN, AFTER, BEFORE

```
<Header version="6.0.11" action="read" objectType="project" externalSource="NIKU">  
  <args name="include_tasks" value="false"/>  
  <args name="include_resources" value="false"/>  
</Header>  
<Query>  
  <Filter name="projectId" criteria="EQUALS">prj123</Filter>  
  <Filter name="projectId" criteria="OR">prj123,prj244</Filter>  
  <Filter name="start" criteria="BETWEEN">2007-01-07,2009-01-15</Filter>  
</Query>
```

# The XML Read Files (3)

- Use the % character as a wildcard

```
<Filter name="projectID" criteria="EQUALS">prj1%</Filter>
```

- Alternatively, you can filter objects based on custom attributes created using Clarity Studio

```
<FilterByCustomInfo name="attribute_id" criteria="EQUALS">prj1</FilterByCustomInfo>
```

- The regular criteria values apply to the Query Filter By Custom section (EQUALS, OR, BETWEEN, AFTER, BEFORE)

# The Properties File

- You can submit a XOG request using XOG client shell commands:

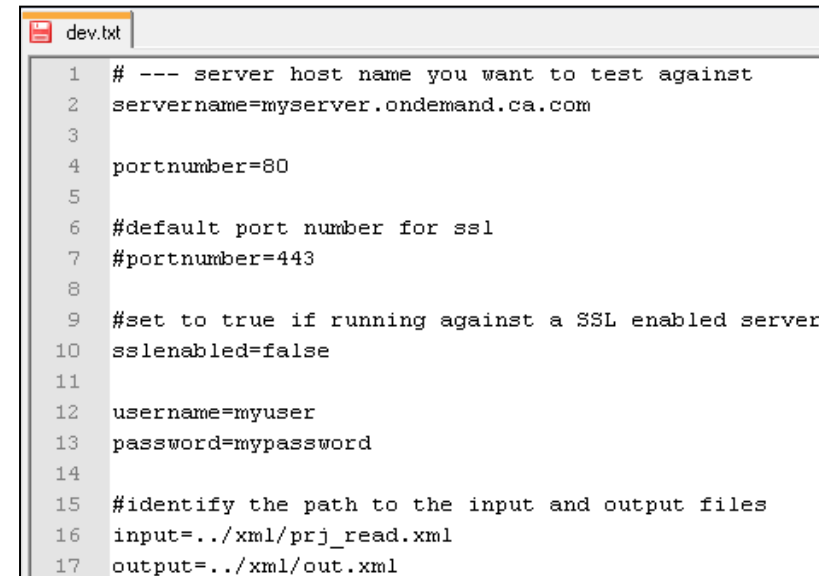
```
xog -servername <host> -portnumber <port>  
-username <username> -password <password>  
-input <input filepath> -output <output filepath>
```

- You can create a .properties file to store the parameters for common XOG requests
  - Use the example .properties file provided with the XOG Client (**C:\xog13\bin\test.properties**)
  - Store the new .properties file in the bin directory
  - Name the file whatever you want ( for example, dev.txt, test.txt, prod.txt)
  - Use a simple text editor like MS Notepad



# The Properties File (2)

- The following properties are required to make a XOG request
  - **servername**=myserver
  - **portnumber**=80 | 443
  - **sslenabled**=false | true
  - **username**=myuser
  - **password**=mypassword
  - **input**=../xml/prj\_read.xml
  - **output**=../xml/out.xml



```
1 # --- server host name you want to test against
2 servername=myserver.ondemand.ca.com
3
4 portnumber=80
5
6 #default port number for ssl
7 #portnumber=443
8
9 #set to true if running against a SSL enabled server
10 sslenabled=false
11
12 username=myuser
13 password=mypassword
14
15 #identify the path to the input and output files
16 input=../xml/prj_read.xml
17 output=../xml/out.xml
```

# Exercise #1: Export Data

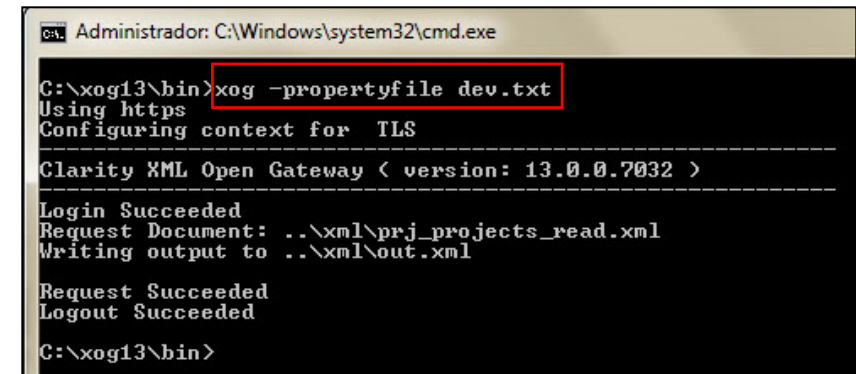
- Submit a XOG Read request using the XOG client as follows
  - Create a .properties file with the default values for the XOG parameters
  - Create an input XML file with the necessary header information, arguments, and query filters
  - Navigate to the “bin” folder under the XOG client installation folder by typing `cd C:\xog13\bin\` and pressing enter
  - Type **xog -propertyfile** *<properties.txt>*
  - Verify the operation succeeded and check the output file

# Exercise #1: Export Data (2)

## Properties File

- # --- server host name you want to test against
- servername=myserver.ondemand.ca.com
- #portnumber=80
- #default port number for ssl
- portnumber=443
- #set to true if running against a SSL enabled server
- sslenabled=true
- username=myuser
- password=mypassword
- #identify the path to the input and output files
- input=../xml/prj\_projects\_read.xml
- output=../xml/out.xml

## XOG Commands



```
Administrator: C:\Windows\system32\cmd.exe
C:\xog13\bin>xog -propertyfile dev.txt
Using https
Configuring context for TLS
-----
Clarity XML Open Gateway < version: 13.0.0.7032 >
-----
Login Succeeded
Request Document: ../xml/prj_projects_read.xml
Writing output to ../xml/out.xml

Request Succeeded
Logout Succeeded
C:\xog13\bin>
```

# Exercise #1: Export Data (3)

## Input File

```
<?xml version="1.0" encoding="UTF-8"?>

<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../../../xsd/nikuxog_read.xsd">

  <Header version="6.0.11" action="read" objectType="project" externalSource="NIKU">

    <!-- you change the order by simply swap 1 and 2 number in the name attribute -->

    <args name="order_by_1" value="name"/>
    <args name="order_by_2" value="projectID"/>
    <args name="include_tasks" value="true"/>
    <args name="include_dependencies" value="false"/>
    <args name="include_subprojects" value="false"/>
    <args name="include_resources" value="false"/>
    <args name="include_baselines" value="false"/>
    <args name="include_allocations" value="false"/>
    <args name="include_estimates" value="false"/>
    <args name="include_actuals" value="false"/>
    <args name="include_custom" value="false"/>
    <args name="include_burdening" value="false"/>

  </Header>

  <Query>

    <Filter name="projectID" criteria="EQUALS">csk.%</Filter>

  </Query>

</NikuDataBus>
```

# Exercise #1: Export Data (4)

## Output File

```
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
  <Header action="write" externalSource="NIKU" objectType="project" version="13.0.0.7032"/>
  <Projects>
    <Project active="true" approved="false" approvedForBilling="1" assignPool="0" billingCurrencyCode="U
      equipmentExchangeRateType="AVERAGE" evCalcMethod="0" expenseExchangeRateType="AVERAGE" fin
      2012-04-26T03:00:01" materialExchangeRateType="AVERAGE" name="Application Change Template" open
      csk.appChange" requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007
      useSystemDefinedTotalCostOfCapital="true">
    <Project active="true" approved="false" approvedForBilling="1" assignPool="0" billingCurrencyCode="U
      expenseExchangeRateType="AVERAGE" financialStatus="O" finish="2007-04-26T10:40:00" format="11" fu
      materialExchangeRateType="AVERAGE" name="Application COTS Template" openForTimeEntry="true" par
      requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-04-09T08:00:00
      ="true">
    <Project active="true" approved="false" approvedForBilling="1" assignPool="0" billingCurrencyCode="U
      expenseExchangeRateType="AVERAGE" financialStatus="O" finish="2007-06-25T17:00:00" format="11" ge
      AVERAGE" name="IT Infrastructure Deployment Template" openForTimeEntry="true" pageLayoutCode="da
      requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-05-01T08:00:00
      ="true">
    <Project active="true" approved="false" approvedForBilling="1" asOf="2007-03-15T00:00:00" assignPool
      equipmentExchangeRateType="AVERAGE" evCalcMethod="0" expenseExchangeRateType="AVERAGE" fin
      materialExchangeRateType="AVERAGE" name="New IT Project" openForTimeEntry="true" pageLayoutCode
      requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-03-12T08:00:00"
      ="true">
  </Projects>
  <XOGOutput>
    <Object type="project"/>
    <Status state="SUCCESS"/>
    <Statistics failureRecords="0" insertedRecords="0" totalNumberOfRecords="4" updatedRecords="0"/>
    <Records/>
  </XOGOutput>
</NikuDataBus>
```

# The XML Write Files

- Use the XML Write files to import a specific object into CA PPM
- Each Write XML file contains the following structure:
  - **Header:** Supported CA PPM version, Operation (Write), object type (Resource, Project, etc.)
  - **Body:** Data to import
- You can create XML write files manually or by modifying the XML Write file examples provided with the XOG client or by using the output of an XML read request
- The output file from a Read response becomes the input file when moving data from one system to another

# The XSD Files

- The XSD files are the XML Schema Definition files
- Each XSD file is used to validate the structure and content of the XML write file
- XSD files contain constraints such as required fields, field lengths, accepted values, etc.
- A valid XML editor can be used to validate an XML Write file against it's schema definition prior to loading to CA PPM
  - If the file is not valid an error will be thrown with the validation error in the output XML file during the write

# Exercise #2: Import Data

- Submit a XOG Write request using the XOG client as follows
  - Create a .properties file with the default values for the XOG parameters
  - Create an input XML file with the necessary header and import data
  - Navigate to the XOG client installation “bin” folder by typing `cd C:\xog13\bin\` and pressing enter
  - Type **xog -propertyfile** *<properties.txt>* and press enter
  - Verify the operation succeeded and check the output file



# Exercise #2: Import Data (2)

## Properties File

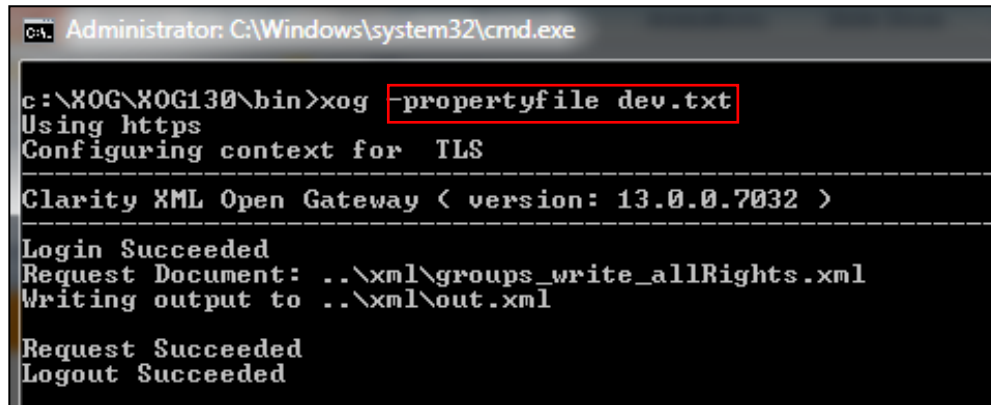
- # --- server host name you want to test against
- servername=myserver.ondemand.ca.com
- #portnumber=80
- #default port number for ssl
- portnumber=443
- #set to true if running against a SSL enabled server
- sslenabled=true
- username=myuser
- password=mypassword
- #identify the path to the input and output files
- input=../xml/groups\_write\_allRights.xml
- output=../xml/out.xml

## Input File

- `<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_group.xsd">`
- `<Header action="write" externalSource="NIKU" objectType="group" version="7.5"/>`
- `<groups>`
  - `<group code="rodmi03.AllRights" isActive="true">`
    - `<nls languageCode="en" name="All Rights"/>`
    - `<members>`
      - `<resource userName="admin"/>`
    - `</members>`
    - `<rights>`
      - `<GlobalRights/>`
      - `<InstanceRights/>`
      - `<InstanceOBSRights/>`
    - `</rights>`
    - `</group>`
  - `</groups>`
  - `</NikuDataBus>`

# Exercise #2: Import Data (3)

## XOG Commands



```

Administrator: C:\Windows\system32\cmd.exe

c:\XOG\XOG130\bin>xog -propertyfile dev.txt
Using https
Configuring context for TLS
-----
Clarity XML Open Gateway < version: 13.0.0.7032 >
-----
Login Succeeded
Request Document: ..\xml\groups_write_allRights.xml
Writing output to ..\xml\out.xml

Request Succeeded
Logout Succeeded
  
```

## Output File

- `<XOGOutput xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="..\xsd/status.xsd">`
- `<Object type="group"/>`
- `<Status elapsedTime="1.079 seconds" state="SUCCESS"/>`
- `<Statistics failureRecords="0" insertedRecords="1" totalNumberOfRecords="1" updatedRecords="0"/>`
- `<Records/>`
- `</XOGOutput>`

# Common Errors

Error	Description
Login Failed	Verify username and password are correct by accessing CA PPM with the same credentials.
No valid input file specified	Verify the file and directory indicated in the input parameter are valid.
Unexpected end of file from server	Check to see if the connection uses SSL (CA PPM URL begins with https://). If the connection uses SSL, set the sslenabled parameter to true.
Failed to retrieve response document java.io.FileNotFoundException:	Verify the directory indicated in the output parameter exists.
HTTP Error: Status-Code 504: Gateway Timeout	The XOG Client cannot connect to the Clarity server. Verify the connection port and test connectivity.
[Fatal Error] :5:47: The entity name must immediately follow the '&' in the entity reference.	Make sure you have escaped all special characters.
[Fatal Error] :6:14:	Verify there are no syntax errors in the XML file.

# Best Practices

# Best Practices

- Use an XML Editor that supports color syntax highlighting, UNICODE, verification, and validation of XML files
  - Altova XMLSpy (License)
  - Notepad ++ (Open Source)
  - XML Pad (Freeware)
  - XML Copy Editor (Open Source)
- Use the XML files from the installation folder of the XOG client as a baseline to create your own XML files
- Verify XML file syntax is correct and validate the XML files against the object schema before running XOG
- Make sure the CA PPM server and XOG client versions match

# Other XOG Methods and XML Creation

# Clarity UI Client

- The CA PPM UI has an unsupported XOG client located at:  
**`http://<servername>/niku/app?action=xog.client`**
- On Demand clients have the above option automatically turned off
  - There is a portlet that is available that mimics the xog.client functionality and can be utilized by adding it to a page
- Both the xog.client and portlet should be used sparingly on smaller XOG files. This uses the application memory (JVM) and can affect performance if the XOG is too large.

# XML Creation Methods

- In addition to manually creating XML for loading to CA PPM other methods can be utilized to generate XML
  - GEL scripting (both locally and in a process)
  - Perl scripting
  - Mail Merge
  - 3rd party tools such as Kettle, etc.

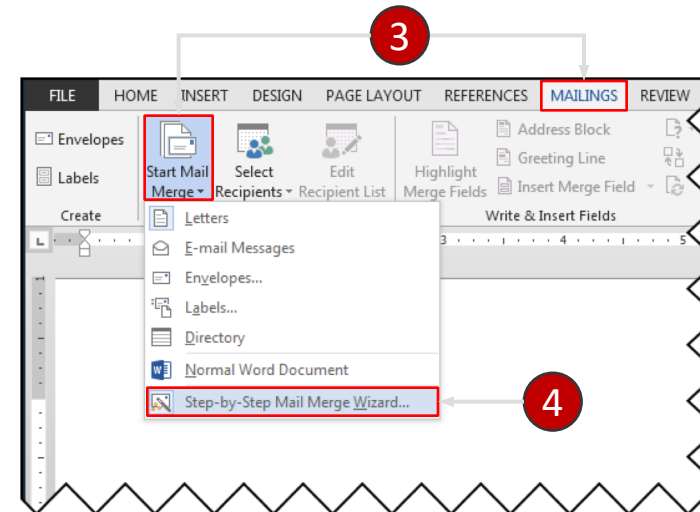


# Import Resources Into CA PPM: Mail Merge

- The purpose of this section is to help the consultant load in existing resources into Clarity
  1. Complete the MS Excel template with the resource data
  2. Open the MS Word template
  3. On the Mailings menu, point to Start Mail Merge
  4. Click Step by Step Mail Merge Wizard



Note: There cannot be blank rows or columns within the list.



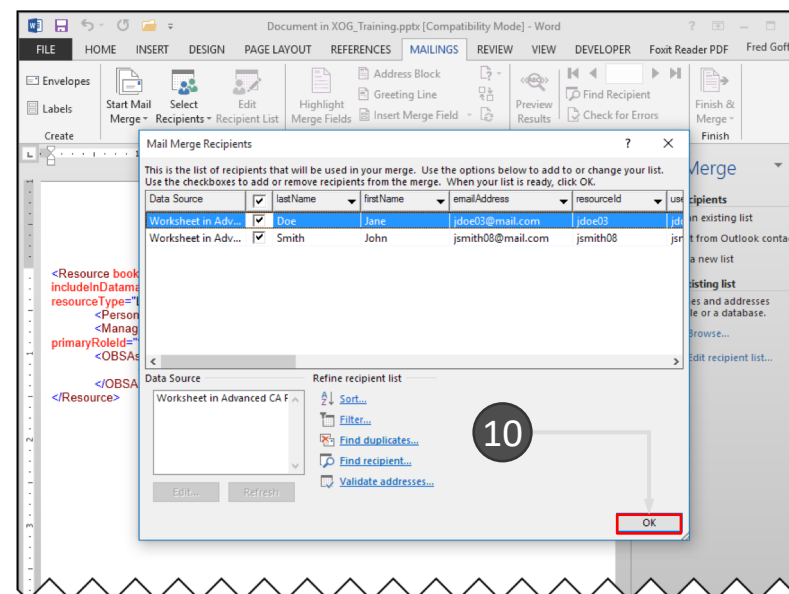
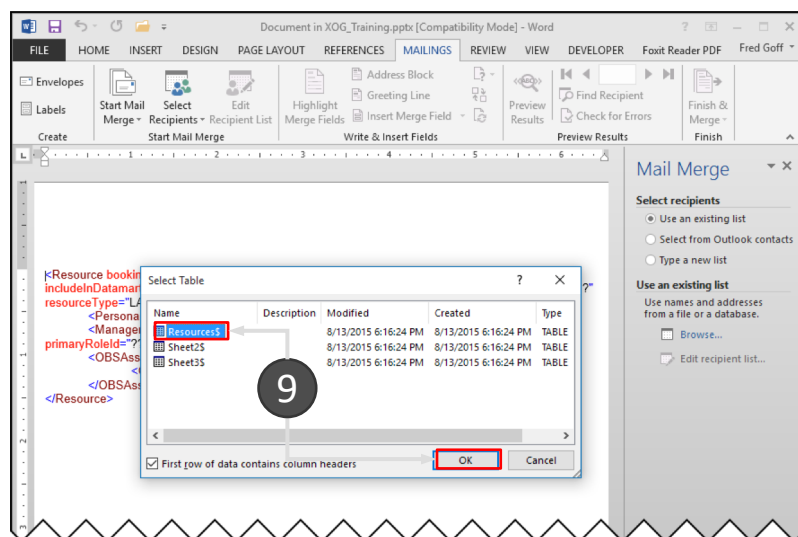
# Import Resources Into CA PPM: Mail Merge (2)

5. Select *Letters* as the document type and click *next*
6. Select *Use the current document* and click *next*
7. Select recipients *Use an existing list* and locate the workbook you created in step 1
8. In the Look in list, click the folder in which you saved the workbook with your data, click the workbook, and then click Open

# Import Resources Into CA PPM: Mail Merge (3)

9. In the *Select Table* dialog box, select your list and click *OK*

10. In the Mail Merge Recipients dialog box, click *OK* to select all records



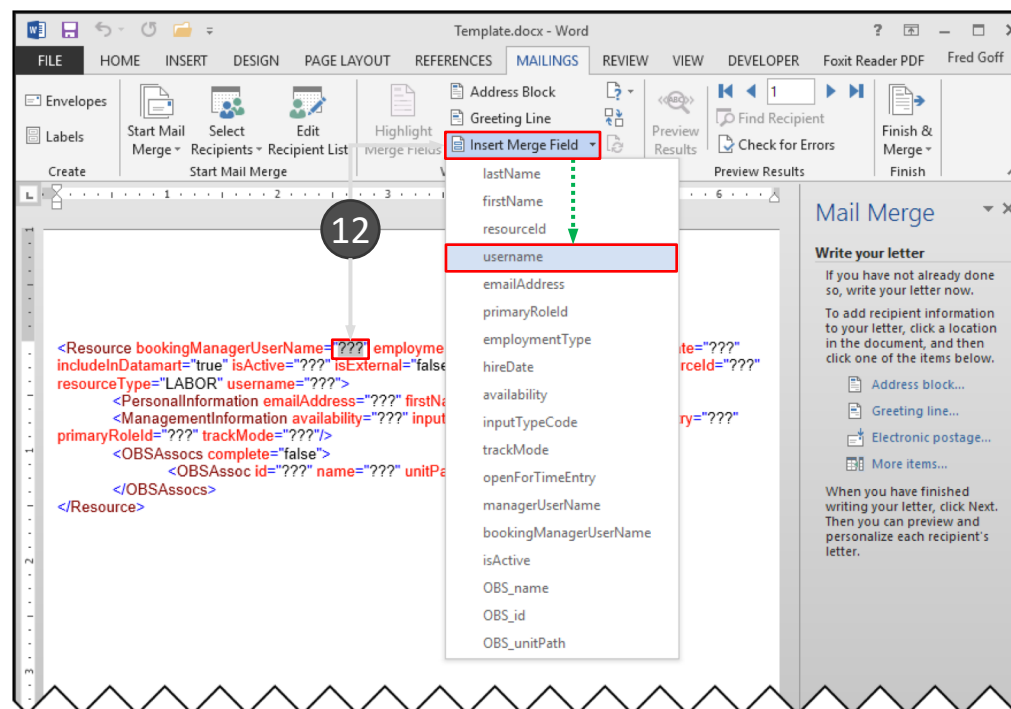
# Import Resources Into CA PPM: Mail Merge (4)

11. Click *Next*

12. Highlight the question marks in the template, and replace with the corresponding merge field using the *Insert Merge Field* drop-down

13. Click *Next*

14. Review the data and click *Complete the merge*



# Import Resources Into CA PPM: Mail Merge (5)

15. Click Edit Individual Letters

16. On the first page enter the XML Write header information

```
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../xsd/nikuxog_resource.xsd">
  <Header action="write" externalSource="NIKU" objectType="resource"
version="13.0.0.7032"/>
  <Resources>
```

17. On the last page enter the closing tags

```
  </Resources>
</NikuDataBus>
```

# Import Resources Into CA PPM: Mail Merge (6)

18. Save the file as a new plain text file
19. Use the Windows default settings for the file conversion
20. Change the extension of the file from *\*.txt* to *\*.xml*
21. Verify the XML file syntax
22. Validate the XML Write file against the resource schema  
(C:\xog13\xsd\nikuxog\_resource.xsd)

# Clean Up XML Write Files

- XML predefines the following five entity references for special characters that need to be escaped (to prevent them from being considered part of the markup)

Name	Character	Escaped
Ampersand	&	&amp;
Left angle bracket	<	&lt;
Right angle bracket	>	&gt;
Straight quotation mark	"	&quot;
Apostrophe	'	&apos;&apos;

- Use CDATA ( `<![CDATA[ ..... ]]>` ) to escape special characters like SQL code

# Cleaning Up XML Write Files Escaping Special Characters

```

<Action code="setInitiator" synchronized="true" type="BPM_SAT_CUSTOM">
  <nls languageCode="en" name="Set Process &amp; Initiator"/>
  <customScript languageCode="gel">
    <scriptText>
      <gel:script xmlns:core="jelly:core"
xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary" xmlns:sql="jelly:sql"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <gel:setDataSource dbId="Niku" var="dataSource"/>
        <sql:update dataSource="${dataSource}"><![CDATA[
          SELECT
            gg_totcost_currency GTC,
            CODE,
            FROM
            ODF_CA_GG_ECOACTIVITY
            WHERE
            id=${gel_objectInstanceId}]]></sql:update>
        </gel:script>
      </scriptText>
    </customScript>
  </Action>

```



# Questions?



*rego*University 2018

Let Rego be your guide.

# Thank You For Attending regoUniversity

## Instructions for PMI credits

- Access your account at pmi.org
- Click on **Certifications**
- Click on **Maintain My Certification**
- Click on **Visit CCR's** button under the **Report PDU's**
- Click on **Report PDU's**
- Click on **Course or Training**
- Class Name = **regoUniversity**
- Course Number = **Session Number**
- Date Started = **Today's Date**
- Date Completed = **Today's Date**
- Hours Completed = **1 PDU per hour of class time**
- Training classes = **Technical**
- Click on **I agree** and **Submit**



Let us know how we can improve!  
Don't forget to fill out the class survey.



### Phone

888.813.0444



### Email

[info@regouniversity.com](mailto:info@regouniversity.com)



### Website

[www.regouniversity.com](http://www.regouniversity.com)