



regoUniversity

KANSAS CITY • 2024

Sponsored by
ValueOps
by Broadcom

Clarity
by Broadcom

Rally
by Broadcom

ConnectALL
by Broadcom

Insights
by Broadcom

Administrator | Advanced

Your Guides:

Chris Ciavarella , Rajini Mamidi

Agenda

- Introduction
- Objects, Attributes and Views/Fields
- Studio and Modern UX
- Introduction to SQL
- Lookups, Queries and Portlets
- Introduction to Workflows (Processes)
- XOG (XML Open Gateway)
- REST API

Introduction

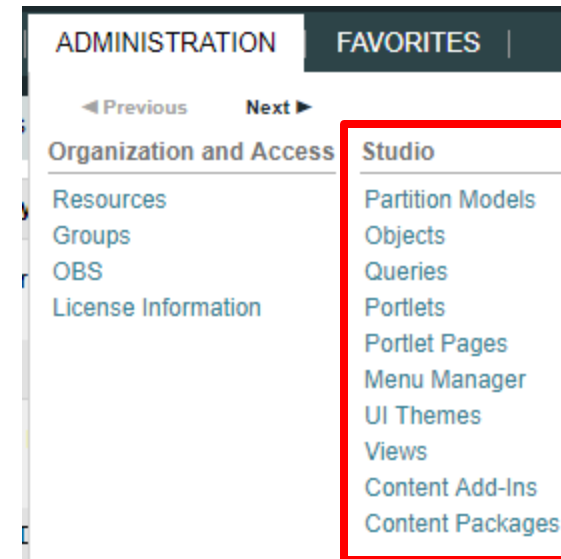


Objects, Attributes and Views/Fields



Clarity Studio

- Studio is the interface used to create new or modify existing components and customize the UI to meet the needs of your organization.
- A full license is required to use this functionality, and users must have a variety of related security rights (Administration-Studio, Create/Edit Objects, Portlets, Pages, etc.).
- Studio is accessed via the Administration menu.



Objects

- Objects are the major functional components of Clarity.
- The properties of objects are defined by the attributes (fields).
- Each object that is enabled for modern user experience has a standard blueprint associated.
- Object subpages (links), page layout, views make up your configured instance of Clarity in Classic. Whereas blueprints, views, and flyout configurations make up the configuration in modern user experience.
- 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”
- Custom Investment Types (CITs) of objects can also be created. These are an extension of the investment object and have additional capabilities such as Cost Plans, Roadmaps, and Hierarchies.
- Use the default objects or create custom objects and sub-objects to manage information for specific business needs.

Objects (continued)

- Each object has distinct pieces you can configure
 - Attributes
 - Screen configuration – layouts (Classic UI) or blueprints (Modern UX)
 - Modules (tabs) on blueprints
 - Views
 - Flyout Configurations
- Things to remember
 - You can only delete Custom Attributes
 - Attributes need an API Attribute ID to show in the Modern UX
 - A CIT (investment extension objects) cannot be deleted
 - 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

Objects: Types

- Stock Objects
 - Primary Standard Objects
 - Investment
 - Idea
 - Project
 - Task
 - Team
 - Resource
 - Company
 - Application
- Custom Objects
 - Master Objects
 - Custom Investment Types (CIT) - Investment Object Extension
 - Sub-Objects

Objects: Investment Object

- The Investment object is a special component that provides the ability to define attributes one time and share them across select OOTB objects.
- These objects “inherit” attributes from the Investment object:
 - Project, Idea, CITs, Other Work, Application, Asset, Product, Service
- Streamlines the creation process and ensures consistency across objects.
- You may re-label attributes on shared objects if needed (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.**

Attributes

- Attributes are the fields on an 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 additional attributes using Studio.
- Once created, you can organize and place attributes:
 - Classic UI: on views and portlets and use for reporting
 - Modern UX: Views and Blueprints
 - Example: “Start Date” is an attribute of the project object
- Details on the various data types can be found in the Studio Developer Guide.

Attribute Data Types

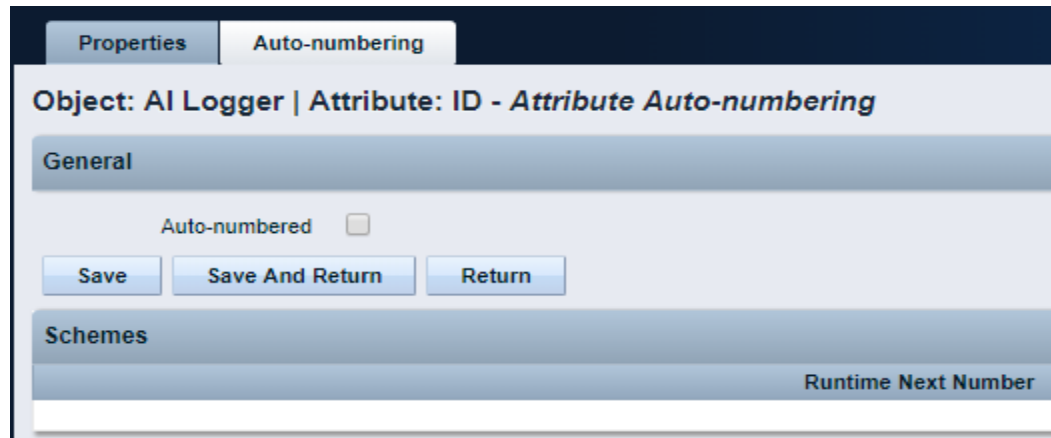
- When creating attributes the following data types are available:
 - String (2000-character maximum)
 - Large String – Plain Text (stored as a CLOB in the database)
 - Large String – Rich Text (stored as HTML and has a Plain Text counterpart for reporting)
*(Only Available in MUX)
 - Number
 - Calculated
 - Money (includes currency code)
 - Boolean (checkbox)
 - Date
 - Lookup (related lookup needs to be available - create prior to creating attribute)
 - Multi-Valued lookup (same note above applies)
 - Attachment
 - Time-varying
 - URL (Links to actual data)

Calculated Attributes

- A Calculated attribute displays a dynamic, read-only value.
- Values are calculated from other existing attribute values.
- Values are calculated every time the user accesses or refreshes the page.
- These values **are not stored** in the database.
- Calculated attributes can only read the following data types :
 - **Number**: Use this data type to calculate a number value like a sum or average
 - **String**: Use this data type to concatenate two or more text values
 - Example: concatenate the value of the attribute “created_by” and the constant “2007” to 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
- **NOTE** – You cannot delete source attributes used in a calculated attribute!

Attributes: Auto-Numbering

- Clarity provides the ability to create your own numbering/naming scheme for object instances (PRJnnnnn, APPnnnnn, etc.).
- The scheme can be numeric or a mixture of characters and numbers.
- Two out-of-the-box attributes that are commonly auto-numbered are “Name” and “ID”
- Configuration is done via the Auto-numbering tab on the attribute detail.



Attributes: Exercise

- Create a new custom sub-object to the project object.
 - Administration -> Objects
 - Click New and fill out the required fields

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)

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

Create Object Definition

Create Object Definition

OBJECT NAME

OBJECT ID

CONTENT SOURCE
Customer

DESCRIPTION

MASTER OR SUBOBJECT
 Master
 Subobject

PARTITION MODEL
(Changing will deactivate all partitioned attributes. Clear to use the System partition.)
 Subobject
 MASTER OBJECT

EVENT ENABLED

COPY ENABLED

EXPORT ENABLED

VIEW ALL ENABLED

API ENABLED

(Once the value is enabled, it cannot be disabled.)

TEMPLATE ENABLED

(Once the value is enabled, it cannot be disabled.)

OBJECT EXTENSION
(An object created with an extension cannot be deleted.)

SAVE SAVE AND RETURN RETURN

- Select the following checkboxes if they apply:
 - **Copy Enabled:** Specifies that copies can be made of the object instances.
 - **Export Enabled:** Specifies that object instances can be exported to XML.
 - **API Enabled:** Allows object to show in the New UX.
 - **Template Enabled:** Allows for this object to have templates defined
 - **Include in Datawarehouse:** Enabled the object to be available in the DWH.
 - **Object Extension:** Used to create custom Investments, by associating to the Investment Object.
 - **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.
 - **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)

Attributes: Exercise (cont.)

- Notice the default fields included in the newly created object.

Properties | **Attributes** | Linking | A

Object: Training Object - Attributes

Attribute Name

Attribute Display

<input type="checkbox"/>	Attribute	Descript
	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.

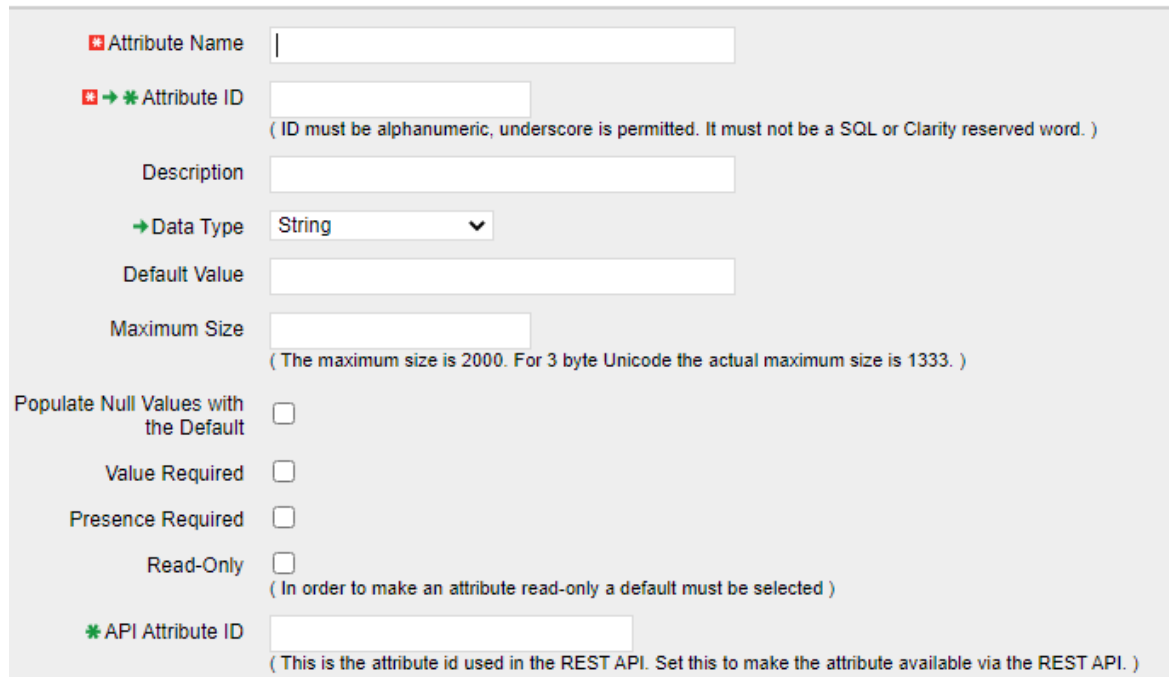
** Note: Outside of custom objects there are OOTB jobs / processes that will skew the results of the last updated by and date fields, as the application often makes updates to the record.

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.

Attributes: Exercise (cont.)

- Add 3 or 4 attributes of different varieties to your new custom object.
 - Click New and fill out required fields as well as Data Type



The screenshot shows a form for creating a new attribute. The fields and their states are as follows:

- Attribute Name:** A text input field with a red asterisk icon.
- Attribute ID:** A text input field with a red asterisk icon and a green plus icon. A note below it states: "(ID must be alphanumeric, underscore is permitted. It must not be a SQL or Clarity reserved word.)"
- Description:** A text input field.
- Data Type:** A dropdown menu currently set to "String" with a green plus icon.
- Default Value:** A text input field.
- Maximum Size:** A text input field. A note below it states: "(The maximum size is 2000. For 3 byte Unicode the actual maximum size is 1333.)"
- Populate Null Values with the Default:** A checkbox.
- Value Required:** A checkbox.
- Presence Required:** A checkbox.
- Read-Only:** A checkbox. A note below it states: "(In order to make an attribute read-only a default must be selected)"
- API Attribute ID:** A text input field with a green asterisk icon. A note below it states: "(This is the attribute id used in the REST API. Set this to make the attribute available via the REST API.)"

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)

API Attribute ID: Required for field to show in New UX. Only available if Object is API Enabled

Attributes: Exercise (cont.)

- Select the following checkboxes if they apply:
 - **Populate Null Values with the Default:** If an attribute is added after instances have already been created, this will populate the existing records with the default value
 - **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
 - **Include in the Datawarehouse:** Include the field in the DWH schema
 - **Attribute API Alias:** Alias to enable the attribute through REST API



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

Attributes: Exercise (cont.)

- Auto-number the ID attribute:
 - Select the auto-numbering tab
 - Check the “Auto-numbered” box 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]

Attributes: Exercise (cont.)

- 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	<input checked="" type="checkbox"/>

= 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

Attributes: Exercise (cont.)

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

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

New Delete Reorder Save Save And Return Return

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

Reorder Auto-numbering Scheme Segments


Scheme Segments: Text(Next Value: TRN)

Numeric Counter(Next Value: 00000001)

Save Save And Return Return

Fields and Views

- Fields are representations of the available attributes which are placed onto screens and list views.
- Field names can be different than the attribute (reabeled).
- They provide additional options and allow flexibility in view configuration:
 - Browse vs Pull-Down (for lookups)
 - Hints and Tooltips
 - Size of the field on the screen and text boxes
 - Default value
 - Required (even if NOT required on the attribute definition)
 - List alignment
 - List column width
 - List word wrapping



Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Project - Views						
View	Category	Setup				
Project Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Program Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Project List	List Column	[Layout]	[Options]	[Aggregation]	[Actions Menu]	[Fields]

Fields: Adding to a Screen

- Fields can be added or removed on the Create and Edit screens.

Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object
Object: Project - Views						
View	Category	Setup				
Project Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Program Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Project List	List Column	[Layout]	[Options]	[Aggregation]	[Actions Menu]	[Fields]

Page > Section		Level
<input type="checkbox"/>	[Create Asset Properties]	Page
<input type="checkbox"/>	+ General	Section
<input type="checkbox"/>	+ Detail	Section
<input type="checkbox"/>	+ Attachements	Section
<input type="checkbox"/>	+ OBS	Section

Create Sections Delete Return

Page > Subpage		Level
<input type="checkbox"/>	[Edit Asset Properties]	Page
<input type="checkbox"/>	+ Asset Summary	Subpage
<input type="checkbox"/>	+ Schedule & Performance	Subpage
<input type="checkbox"/>	+ Alignment & Risk	Subpage
<input type="checkbox"/>	+ Financial Summary	Subpage
<input type="checkbox"/>	+ Compliance	Subpage
<input type="checkbox"/>	+ Settings	Subpage

Create Subpages Delete Return

Fields: Adding to a List

- Fields can also be added or removed on the List view.

Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Project - Views						
View	Category	Setup				
Project Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Program Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Project List	List Column	[Layout]	Options]	[Aggregation]	[Actions Menu]	[Fields]

Column Layout

Available Columns	Selected Columns
Actuals for Labor Resources	ID
Actuals Sum for Labor Resources	Compliance
Alignment	Value Metrics
Architectural Fit	Gray Bar
Billing Currency Code	Related Idea
Blueprint	Approved Flag
Blueprint Active ID	Status
Board Plan	Progress
BTM Integration	Start
Budgeted Benefit	Finish

Fields: List Options

- Consider the following List view options to improve the user experience:
 - Rows per Page
 - Allow Configuration
 - Filter results

Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Project - Views						
View	Category	Setup				
Project Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Program Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Project List	List Column	[Layout: Options]	[Aggregation]	[Actions Menu]	[Fields]	

Display Options

Secondary Value Display Mouseover only
 Mouseover and redline text
(Used when any list column field displays a secondary value)
 Show Null Secondary Values

Filter Automatically show results
 Do not show results until I filter

Rows per Page 50 ▾

Highlight Row by Attribute Approved Flag
(A row will be highlighted when this attribute is not zero)

Display Currency Code in Column
(Applies when only a single currency is active)

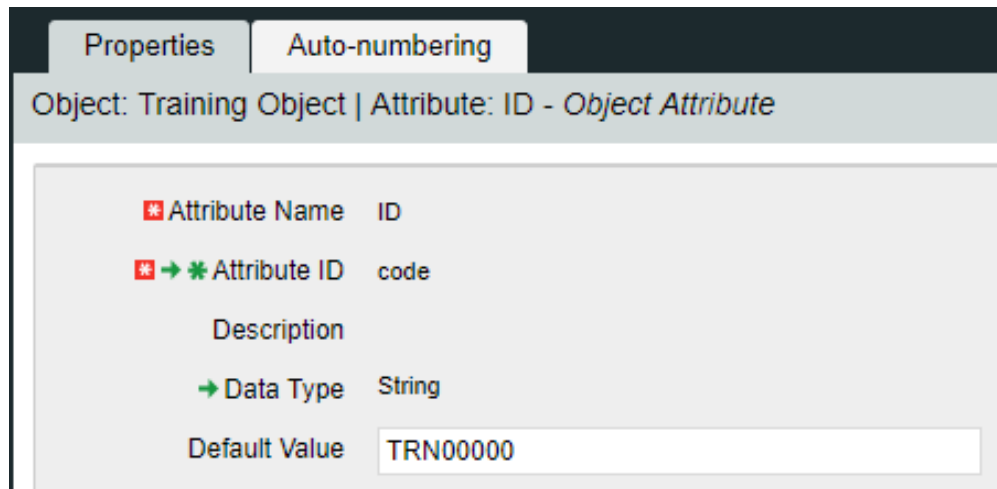
Allow Configuration

Allow Label Configuration

Attribute Value Protection Use display conditions and secured subpages to protect attribute values on this list
 Use only secured subpages to protect attribute values on this list
 Display all attribute values on this list

Fields: Exercise

- 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. Set a default value. Click Save and Return.

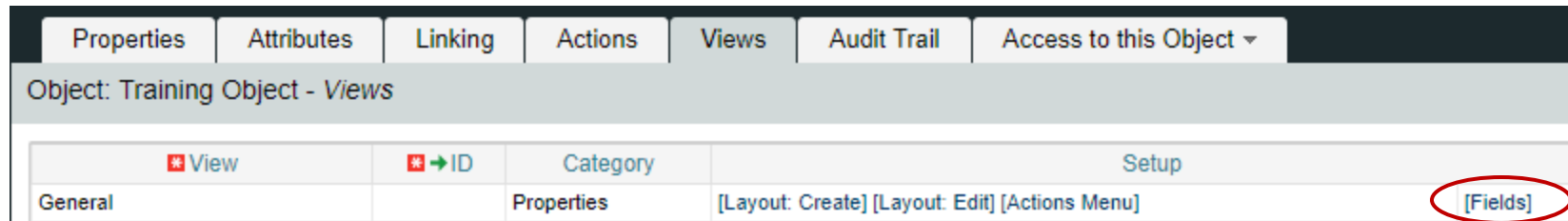


The screenshot shows a configuration window with two tabs: 'Properties' and 'Auto-numbering'. The 'Auto-numbering' tab is active. The window title is 'Object: Training Object | Attribute: ID - Object Attribute'. The main content area displays the following fields:

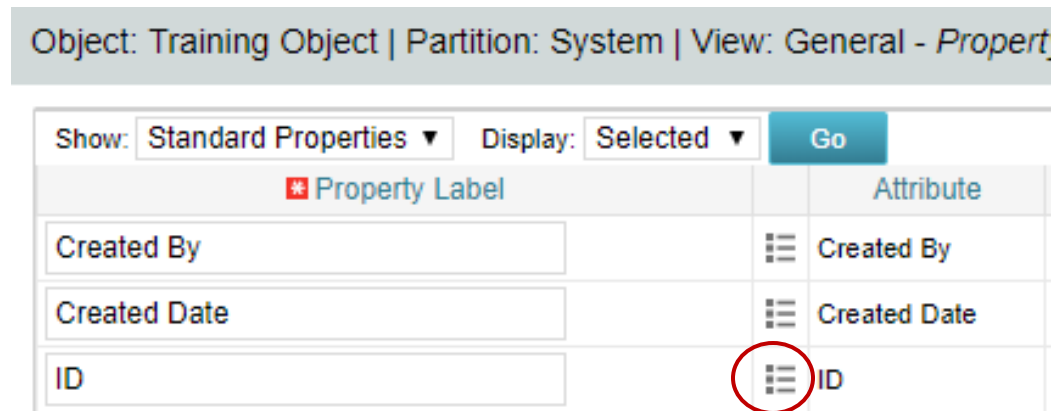
* Attribute Name	ID
* → * Attribute ID	code
Description	
→ Data Type	String
Default Value	<input type="text" value="TRN00000"/>

Fields: Exercise (cont.)

- Select “Views” tab
- On the General -> Properties line click the “Fields” link



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



Fields: Exercise (cont.)

- Select the “Hidden” checkbox

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

Attribute	code
Data Type	String
* Property Label	<input type="text" value="ID"/>
Display Type	<input type="text" value="Text Entry"/>
Hint	<input type="text"/>
Hint Position	<input type="text" value="Below"/>
Tooltip	<input type="text"/>
Attribute Default	TRN00000
Override Default Value	<input type="text"/>
Width	<input type="text" value="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.)

Fields: Exercise (cont.)

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

Object: Training Object | Attribute: Description - *Object Attribute*

Attribute Name	Description
Attribute ID	v_desc
Description	
Data Type	String
Default Value	
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"/>

Fields: Exercise (cont.)

- 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"/>

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
- Actions and the associated menus can be renamed as needed.

Studio and Modern UX



Administration in the Modern UX

- The Modern UX Administration is more functional than technical. It can be managed by any user with a good functional understanding of Clarity.
- While some administration activities are done in the Classic Clarity UX, others are available only in the Modern UX.
- Administrators must have a good understanding of what they want to expose and to whom they wish to make functionality available to make best use of the Classic and New functionality.
- Studio Objects and Attributes are available in the Modern UX, they only need an API Field ID to be configured.

Enabling Attributes for the Modern UX

- For custom attributes to be able to be viewable/editable in the Modern UX, their API Attribute ID must be populated.
- Access your custom attributes on the object to which they belong, and populate this value to ensure the attributes may be utilized in the Modern UX.
 - Go to Administration -> Objects -> Select the appropriate object
 - Select *Attributes* Tab
 - Select Desired Attribute
 - Populate *API Attribute ID* and **Save**

The screenshot shows the 'Clarity PPM' Administration interface. The breadcrumb trail is 'Home > Administration > Favorites'. The current page is titled 'Object: Project | Attribute: Project Kickoff Meeting Complete - Object Attribute'. The 'General' tab is active, showing the following configuration:

- Attribute Name: Project Kickoff Meeting Complete
- Attribute ID: huss_pk
- Description: (empty field)
- Data Type: Lookup - Number
- Lookup: Yes or No
- Default: (empty field)
- 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)
- API Attribute ID: PKMeeting** (This is the attribute id used in the REST API. Set this to make the attribute available via the REST API.)
- Include in the Data Warehouse:
- Include in the Data Warehouse Trending:

Note: You must API-Enable OBS attributes for them to be utilized in filters, views, etc.

Basic Components Modern UX

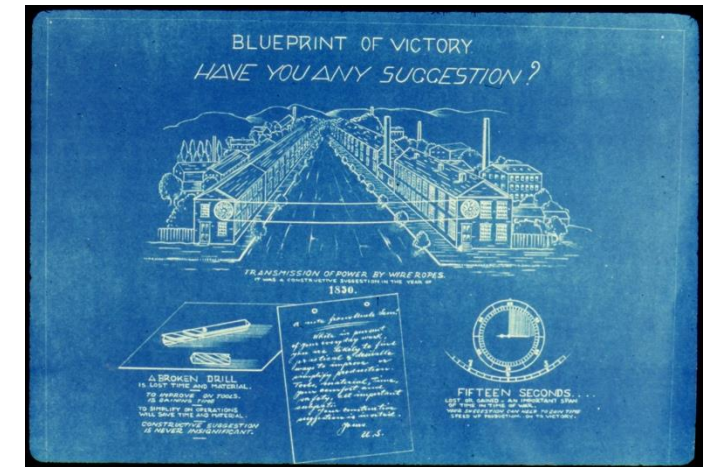
- Blueprints
 - Modules
 - Views
 - Channels
 - Actions
 - Business Rules
- Attributes
- Field Level Security
- Pages (Dashboards)
- User Personalization
 - Views
 - Picklists

Configure Modules

- Blueprints
 - Properties
 - Visuals
 - Modules
 - Rules
 - Actions
 - Channels
 - Canvas
 - Create from Template
- Views
- Flyouts

Blueprints Overview

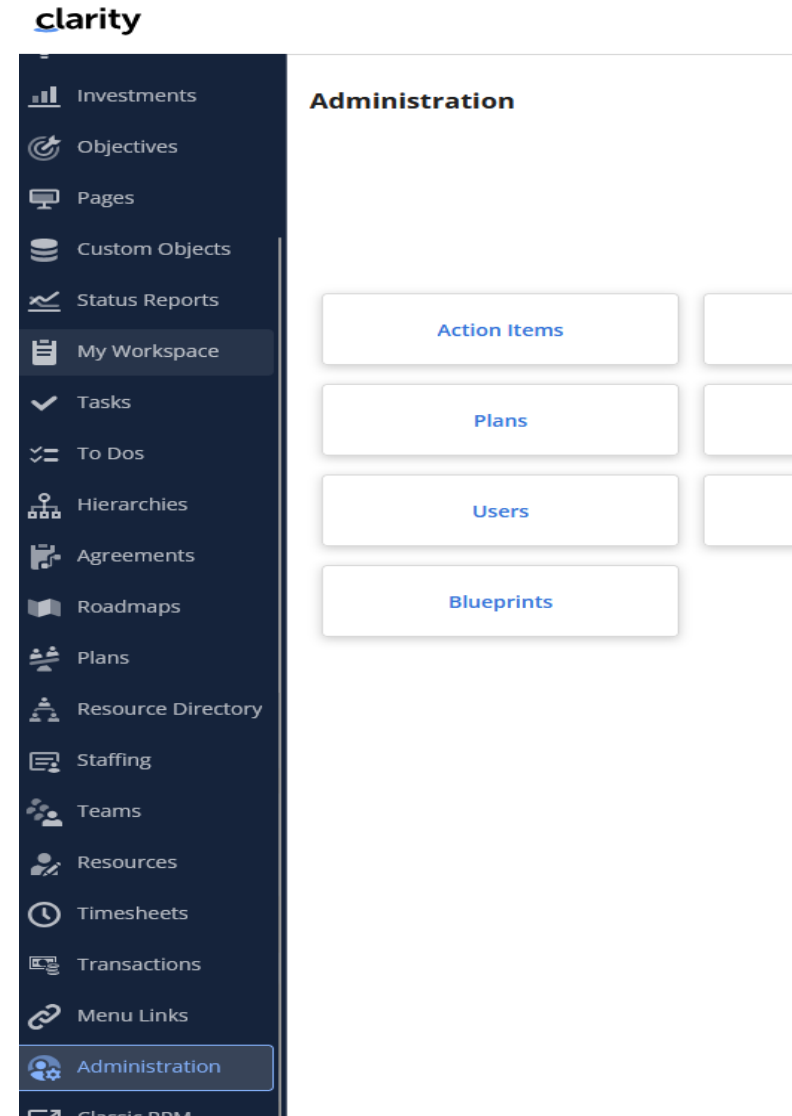
- Blueprints are configurable layouts in the New User Experience.
- Blueprints can be used (might vary depending on version):
 - Projects
 - Ideas
 - Custom Investments
 - Custom Objects
 - Hierarchy
 - Roadmaps
 - Agreements
 - Pages
- Depending on the Blueprint type options might vary, but in general Blueprints can:
 - Allow admins to customize the look and feel of Edit Views, Add Modules, Add Channels
 - Impact how certain investments are created from templates
 - Implement Rules to enforce validations like required fields in Objects and Subobjects (16.0.3)
 - Configure Actions to run processes within MUX
 - Produce attribute updates from user inputs
 - Generate action items from user inputs



Configuring Blueprints

Navigation

- To have the ability to view and update Blueprints a user requires the following security rights:
 - Blueprint – Create Copy, Blueprint - Delete – All, Blueprint – Edit – All, and/or Blueprint – View – All
1. Once in the New User Experience, Click on the ‘Administration’ Icon.
 2. Click on the ‘Blueprints’ tile:
 - Here you will see the list of Blueprints create in the system



Blueprint List View

- Here you will see a list of the Blueprints that have been created in the system:
 - The list is filterable; can filter by Blueprint type (e.g., Idea vs. Project vs. Custom Investments)
- There are multiple options for a Blueprint:
 - Copy – Create a new Blueprint which is a one for one copy
 - Rename – Rename the Blueprint
 - Delete – Delete the Blueprint
 - Make Default
 - Any newly created Idea or Custom Investment will inherit this Blueprint.
 - Projects not created from a template will inherit this Blueprint

Administration ▾

Blueprints

↓

Match Filters **All** Any Add filter groups

▼ Type = Project × + Add filter Remove all

Select all Deselect all

Group By	
Name * ↑	Type * ↓
<input type="checkbox"/> Standard Project	Project
<input type="checkbox"/> Standard Project - Demo	Project
<input type="checkbox"/> New Project Blueprint--with Smartsheet	Project
<input type="checkbox"/> Standard Project - Copy TTM	Project
<input type="checkbox"/> New Project Blueprint - Test	
<input type="checkbox"/> Standard Project - Blueprinttesting	
<input type="checkbox"/> Project Gates Blueprint	

- Open Details
- Star
- Make default
- Copy
- Delete Row
- Chart Range (beta) >

Blueprint List View cont.

- "Has Draft" indicates if there are unpublished changes to the blueprint.

Select all Deselect all

Group By		
Name * ↑	Has Draft	Type
<input type="checkbox"/> Standard Project		PI
<input type="checkbox"/> Standard Project - Demo	✓	PI
<input type="checkbox"/> New Project Blueprint--with Smartsheet		PI
<input type="checkbox"/> Standard Project - Copy TTM		PI
<input type="checkbox"/> New Project Blueprint - Test	✓	PI
<input type="checkbox"/> Standard Project - Blueprinttesting	✓	PI
<input type="checkbox"/> Project Gates Blueprint		PI

Properties

- On Project Blueprints, following main features can be defined.
 - **Properties**– This is the main details or properties page for the Project/Custom Investment Type. Here you can add or remove fields and sections. You can also move and resize fields by dragging and dropping them.
 - **Visuals** – These are the icons on the Project Tiles. There can be a maximum of 3, but there is a minimum of 1 required. Currently, these are only available for the Project Blueprints.
 - **Modules** – These are the supporting “pages” that can be added or removed from the Project. The modules include functionality like Financials, Teams, Risk, Issues, Changes, etc. Currently, these are only available for Project Blueprints.
 - **Rules** – These are business rules that can govern the data and its visibility.
 - **Actions** – These are links to trigger active processes associated with the object.
 - **Create from Template** – This lets users ensure all required data is captured before project is created.

Administration > Blueprint >

Custom Project Blueprint Published: Aug 28, 2024 [Edit](#)

[Properties](#) [Visuals](#) [Modules](#) [Rules](#) [Actions](#) [Create from Template](#)

[Collapse all](#)

PROJECT SUMMARY

Project Name

Project ID

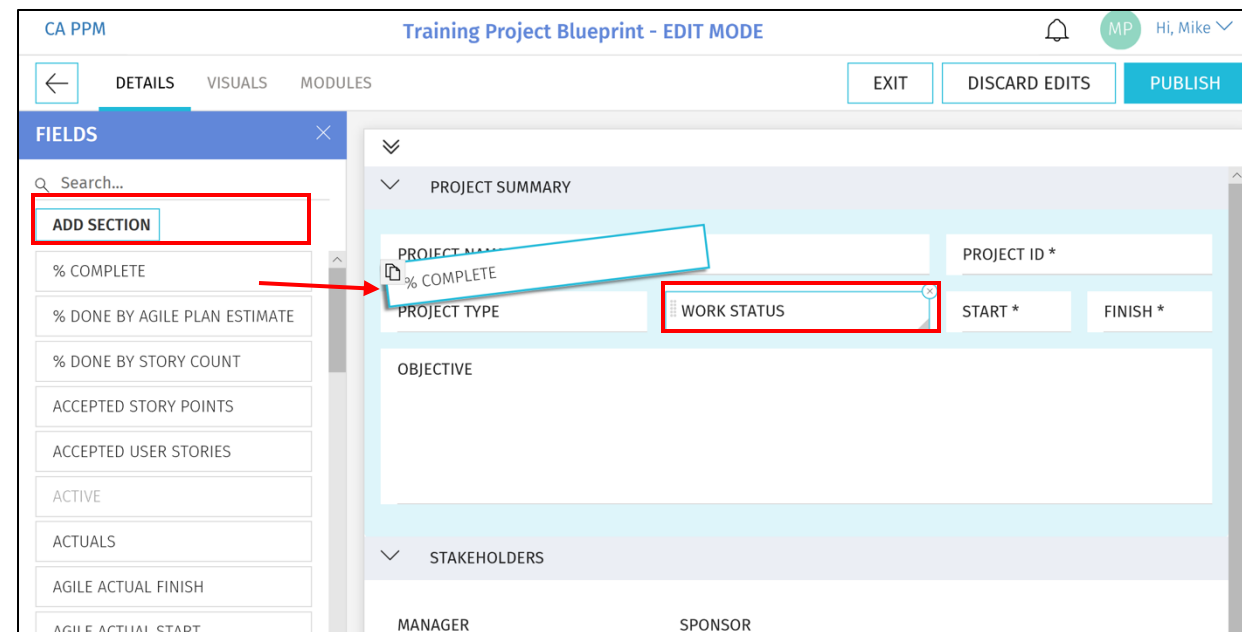
Properties

• Fields Pane

- Location of the fields you can add to the sections of your Details
 - By default, it contains a list of out-of-the-box Project and Investment fields
 - Custom fields and sub-objects can be added and will be covered later
- Fields that already exist on your Blueprint are greyed out
- Click the Add Section button to add a new section to the Details

• Fields

- Add or Move a field by simply dragging and dropping the field into a section
- Remove a field by clicking the X in the top right-hand corner of the field
- Resize the field by dragging the bottom right-hand corner of the field



Properties cont.

- Details Options:
 - **Exit** - Allows you to save your changes without Publishing the new view to Users
 - **Discard Edits** – Removes the change you have made
 - **Publish** – This Publishes the new view the users for the Projects associated to this Blueprint

CA PPM Training Project Blueprint - EDIT MODE

← DETAILS VISUALS MODULES

EXIT DISCARD EDITS PUBLISH

FIELDS

Q Search...

ADD SECTION

% COMPLETE

% DONE BY AGILE PLAN ESTIMATE

% DONE BY STORY COUNT

ACCEPTED STORY POINTS

ACCEPTED USER STORIES

ACTIVE

ACTUALS

AGILE ACTUAL FINISH

AGILE ACTUAL START

PROJECT SUMMARY

PROJECT ID *

PROJECT TYPE WORK STATUS START * FINISH *

OBJECTIVE

STAKEHOLDERS

MANAGER SPONSOR

Visuals (Projects Only)

- Visuals are displayed on the Project Tiles.
- Currently, there are 10 out-of-the-box Visuals to choose from.
- Add or Move a Visuals by simply dragging and dropping it.
- Remove a Visual by clicking the X in the top right-hand corner of the icon.

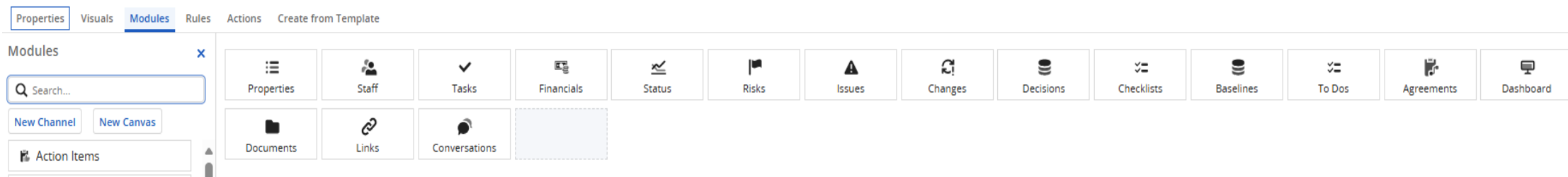
The screenshot displays the 'Training Project Blueprint - EDIT MODE' interface. The top navigation bar shows 'CA PPM', 'Training Project Blueprint - EDIT MODE', a notification bell, and a user profile 'MP Hi, Mike'. Below the navigation bar are tabs for 'DETAILS', 'VISUALS', and 'MODULES', along with 'EXIT', 'DISCARD EDITS', and 'PUBLISH' buttons. The 'VISUALS' section is active, showing a search bar and a list of 10 visual options:

- % COMPLETE
- BUDGET REMAINING
- BUDGET SPENT
- DAYS REMAINING
- DAYS TO START
- EFFORT REMAINING
- EFFORT SPENT
- FINISH DATE
- NEXT MILESTONE
- START DATE

The main canvas shows a project tile with sections for 'Project Name', 'TIMELINE', and 'MODULES'. Three visual tiles are placed on the canvas: '% COMPLETE' (highlighted with a blue box and an arrow), 'BUDGET SPENT' (a donut chart), and 'NEXT MILESTONE' (a diamond shape with an 'X' in the top right corner, highlighted with a red box).

Modules

- Once inside of a Project, Modules are displayed across the top.
- The first 4 Modules will also be displayed on the Project Tile for direct navigation to that Module.
- There are several core Modules which are not configurable and provide project functionality like financials, staff, task, etc.
- The sub-object modules can be configured for the following:
 - **Enable Properties Navigation** – "Name" attribute of sub-object instance is hyperlinked to Properties page
 - **Enable Quick Create** – Create new records in the grid using the '+' button
 - **Enable Create Dialog** – Create new records with button that loads a modal popup window
 - **Show in Details Flyout** – Visible as a tab in the details flyout for of the parent instance
- In addition to the core Modules, there are configurable Channels and the ability to add custom sub-objects.



Modules – Channels

- Channels are configurable Modules that can be directed to other internal PPM locations, external applications, or external URLs.
- Users can stay directly in their Project and get the additional pertinent information.
- Configuration:
 - Channel Name – The name displayed to the user
 - Channel URL – The URL where the channel will navigate
 - Referrer URLs – These are additional URLs need for navigation like authentication
- The URL fields accept various parameters to send as query strings to other apps and/or options to interact with the Classic UI.

The image displays two screenshots from the CA PPM interface. The top screenshot shows a project plan for 'ACME Project' with a 'smartsheet' view. The bottom screenshot shows the 'Training Project Blueprint - EDIT MODE' interface, specifically the 'MODULES' section. A 'Configure - Smartsheet' dialog box is open, showing the configuration for a channel named 'Smartsheet'.

Task Name	Duration	Start	Finish	% Complete	Staffing Size	Status	As
- Initiation	14d	12/28/15	01/14/16	100%			
Detailed Requirer	6d	12/28/15	01/04/16	100%	4	Green	Mike
Hardware Requirer	5d	01/05/16	01/11/16	100%	4	Blue	Ed
Final Resource P	2d	01/12/16	01/13/16	100%	4	Yellow	Ba
Staffing	1d	01/14/16	01/14/16	100%	4	Yellow	Ed

Configure - Smartsheet

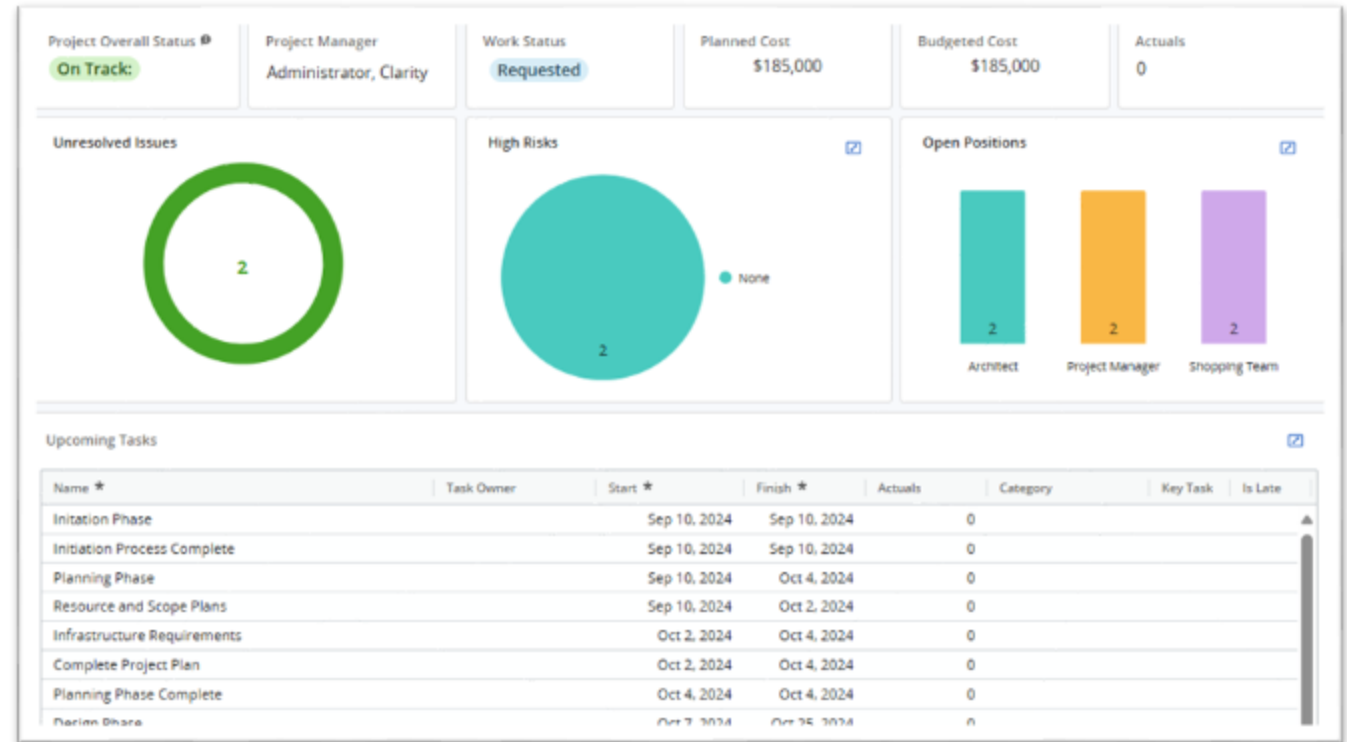
CHANNEL NAME *
Smartsheet PREVIEW

CHANNEL URL *
https://app.smartsheet.com/b/publish?EQBCT=xxxxxxxxxxxx

REFERRER URLS
Comma Separated

Modules – Canvas

- Canvas provides a consolidated view.
- Canvas layout is configurable with various widgets.
- Users can export to PDF.
- Different views can be configured per persona.



Modules – Custom Sub-Objects

- Custom sub-objects of Projects, Ideas or Custom Investments as a Module in the New User Experience.
- Check the “API Enabled” checkbox on the existing custom object or a new custom object.
 - Once this is checked and saved, it can’t be undone
- After saving, an API Attribute ID will be automatically created for the object and it will be available as a Module.

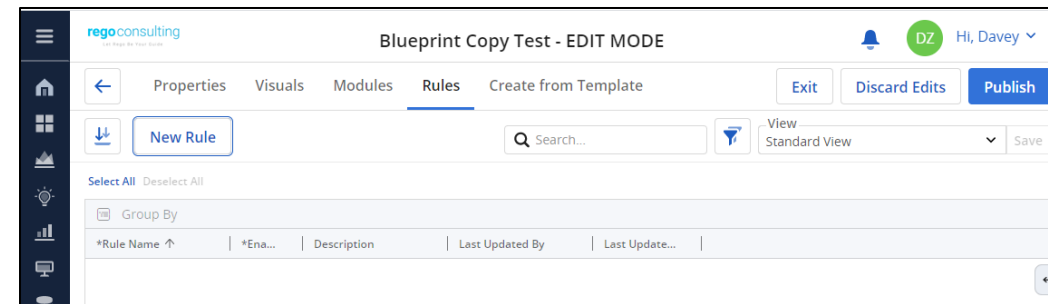
The screenshot displays the configuration interface for a custom sub-object. The top section shows the 'Subobject' configuration with the 'Master Object' set to 'Project'. The 'API Enabled' checkbox is checked and highlighted with a red box, with a note below it stating '(Once the value is enabled, it cannot be disabled.)'. Below this, the 'Properties' tab is active, showing the 'Object Name' as 'Test Sub Object', 'Object ID' as 'test_sub_prj_obj', and 'API Attribute ID' as 'custTestSubPrjObjs', which is also highlighted with a red box. The 'Content Source' is set to 'Description'. At the bottom, a grid of modules is shown, with the 'TEST SUB OBJECT' module highlighted in a red box.

Rules Engine

- In the Modern UX there is now the option to create logic for hiding certain modules or sections based off attribute values.
- Easy way to create a flexible and dynamic user experience using views.

Example: Demand Management process

- Navigate to Modern UX → Administration
- Select on the Blueprints tile
- Search for the blueprint you want to create logic on
- Select Edit
- Navigate to the Rules tab
- Select New Rule



View Rule

Rule Name *

Hide Financials and Tasks - Stage Closing

Description

Enter the rule description

Target Object

Project

Rule Type

Determine when the below actions are executed. Only one rule type may be selected per business rule.

View Page - Runs every time a user views page, supports only one complex condition (UI action Only)

Attribute Update - Run only when a selected attribute is updated, supports multiple conditions blocks (Data and UI actions)

Conditions Always True

Match Filters All Any

Stage = Closing x + Add filter Remove all

Actions

Hide Modules Financials x Tasks x + Select

+ Add actions

Attributes - Field Level Security

- In the Modern UX there is now the option to secure attributes.
- Easy way to avoid users from seeing certain attributes in their configurable list view.
 - Navigate to Modern UX → Administration
 - Select on the Attributes tile
 - Filter for object
 - Mark the attribute secure
 - Access Edits and Access View is required.



Administration ▾

Attributes

Attributes Investment Parents

↓

Match Filters **All** Any Add filter groups

▼ Object = Gates x + Add filter Remove all

Group By

Attribute ↑	Attribute ID	Object	Secure	Access Edit	Access View	Database Table
Active	r_active	Gates	✓	Checklist Sharing	All Users	odf_ca_r_gates

Attributes - Investment Parents

- We can define the allowed Parent investment types.

Investment ▾
Strategy

Actions ▾ [New from Template](#)

Select all Deselect all

Group By

A.. ▾	Investment ID *	Name *	Assignment Categ...	Created By	Parent
<input type="checkbox"/>	ST0019	Better Serve the Department by Strengthening		McCoubrie, Wes	
<input type="checkbox"/>	ST0017	Zero Harm Continually Improve Care		Administrator, Clarity	
<input type="checkbox"/>	ST0016				
<input type="checkbox"/>	ST0015				
<input type="checkbox"/>	ST0014				
<input type="checkbox"/>	ST0013				
<input type="checkbox"/>	ST0012				
<input type="checkbox"/>	ST0011				
<input type="checkbox"/>	ST0010				
<input type="checkbox"/>	ST0009				
<input type="checkbox"/>	ST0008				
<input type="checkbox"/>	ST0007				
<input type="checkbox"/>	ST0006				
<input type="checkbox"/>	ap1212121e				

Search...

Investment Name	Investment Code	Type
-- None --		
1.1.1 Undertake compensation reviews at the start of each fiscal year	STRAT0017	Strategic Plan
1.1 Establish Effective Leadership Practices	STRAT0016	Strategic Plan
1. Recruit, develop and retain key staff.	STRAT0015	Strategic Plan
Strategic Plan	STRAT0014	Strategic Plan
1.1.2 Review achievements at staff meetings >4 times per year (ongoing)	STRAT0018	Strategic Plan
Grow digital advertising investment from 9% to 40% of budget	a4	Value Stream

regoconsulting

Administration ▾

Attributes

Attributes [Investment Parents](#)

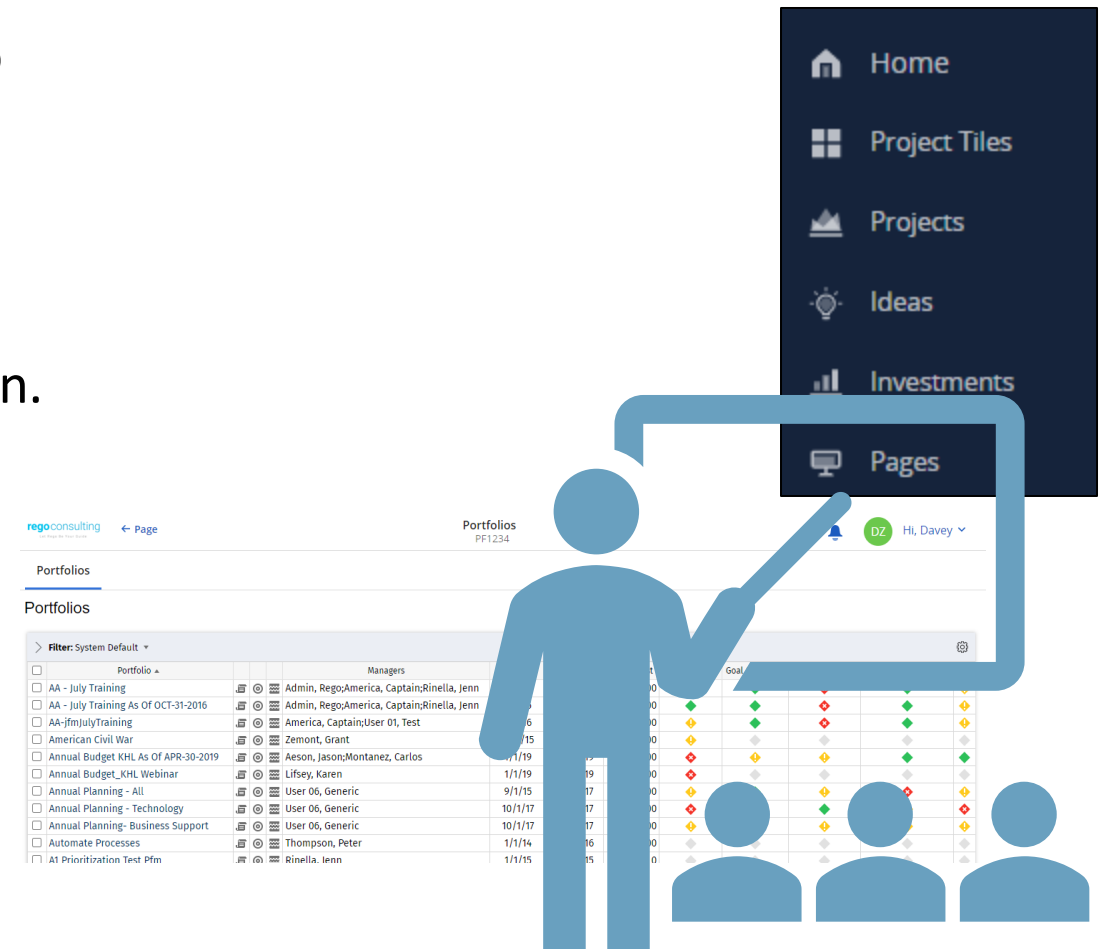
Set the allowed parent types of investments

[Add investment type](#)

Investment Type	Allowed Parent Types
Project	Product; Programs; Project
Strategy	Strategic Plan; Value Stream
Story	Enhancements
Enhancements	Release

Pages and Dashboards

- In the Modern UX there is a new option to display portlet dashboards called Pages.
 - Navigate to Modern UX → Pages
 - Select on the + icon to create a new row
 - Populate a Name and Id with relevant information.
 - Example: Portfolio List
 - Navigate to Administration → Blueprints
 - Filter Type → Page
 - Copy Standard Page
 - Edit Copied version to relevant name
 - Navigate back to Pages
 - Select new blueprint and add Channel
 - Input new Name and Address for Channel.
 - Example:
 claritytest.com/niku/nu#action:pfm.portfolioList&puiFull
 screen=on



Saving Views and Filters (1)

- In the Classic UX, Administrators may save configurations to list views and filters, to ensure users see the right fields in view.
- Similar functionality is open to users in the Modern UX.
- Mini visualizations called widgets are also possible in Modern UX.

Projects

Actions New from Template Search... Per-Period Metrics - 12 Periods View: Approval Status Insights Save Filter Widgets

Match Filters All Any Add filter groups

Active = Yes Template = No + Add filter Remove all

Actuals by Strategic AL... 2.75K 1.87K 1.01K

Planned Cost ... **90.46M**

Actual Cost **945.41K**

Estimates **100%**

Work Status Requested Active On Hold Other

Benefits **139.14M**

Project Cost by Type Major Proje... None Other Proposal

Select all Deselect all

Group By

Project Name *	Status *	pr... *	Actuals	Parent #	Planned Benefit	Department	Planned Cost	Planned Capital Cost	Approved By	Work Status	Template	ETC	EAC
<input type="checkbox"/> CRM Contact Center Development	Approved	PR1028	1,822	Order Manage...		Business Operations	26,000,000	13,000,000		Active		12,166	5,4
<input type="checkbox"/> CRM Enhancements	Approved	PR1029	960	Service Line Exp...	750,000	Business Operations	8,977,920	140,000		Active			1,440
<input type="checkbox"/> eCommerce Portal	Approved	PR1002	308	iSeries Program		Shared Services	612,000	420,000		Complete		11,559	1,3
<input type="checkbox"/> Minimal Online Shopping Site	Approved	PRS1012	254	Real Estate Mal...		Business Operations	8,356,600	4,584,008	Thompson, Peter	Closing		0	
<input type="checkbox"/> Internal Workflow Automation with RPA and ...	Unapproved	PR1466	0	New Insurance ...	1,300,000	Business Operations	109,600	95,200		Active		760	
<input type="checkbox"/> Creative Solutions	Unapproved	PR1489	39		240,000	Business Transfor...				Active		8,097	

Filter Widgets

Search...

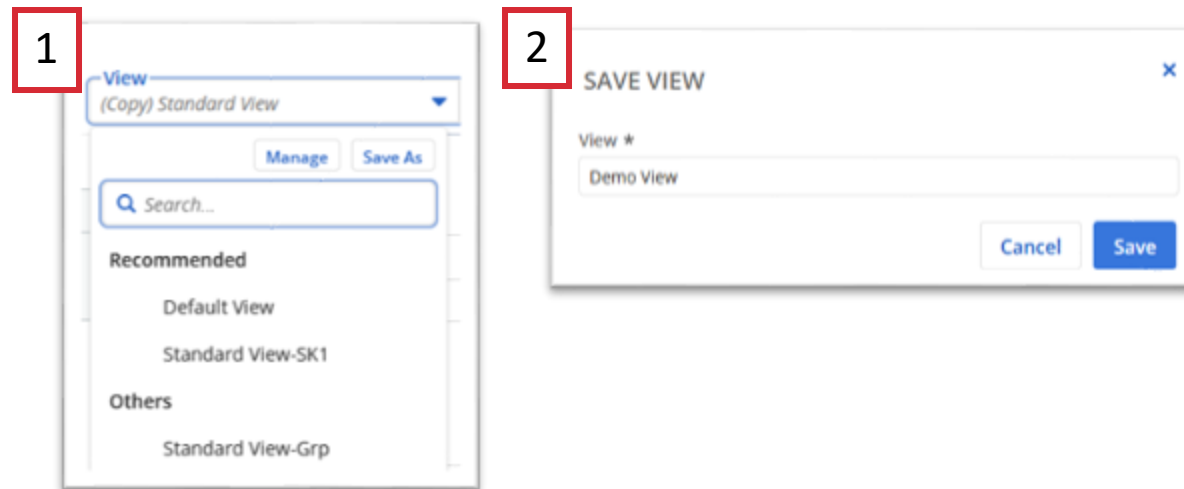
Favorites

- Approval Status Insights
- AW FM Demo
- Board view for status w actuals
- By strategic alignment by approval state
- By strategic alignment by approval state for wed
- Cost Out Benefits View
- Cost Out Benefits View - Starred

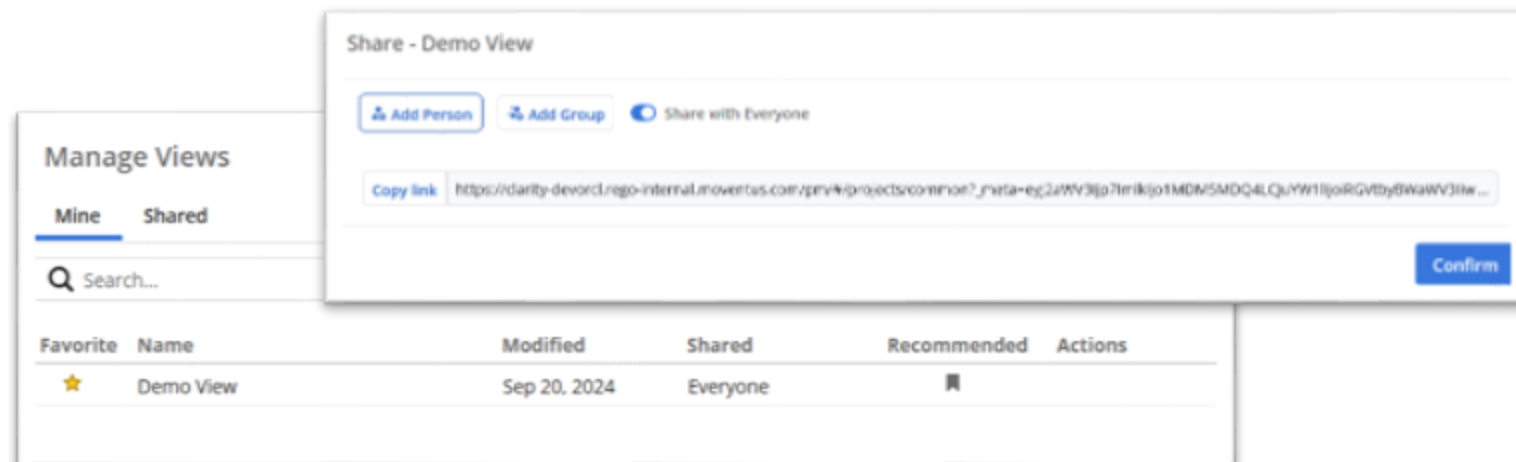
Saving Views and Filters (2)

To save a view (continued):

- Once the filter and fields are added, use the *View* menu in the upper right corner to select **Save As** to Save the View.
- Enter a name for the View, and click **Save**

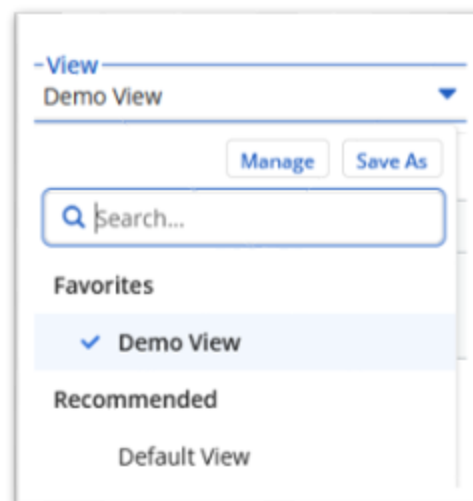
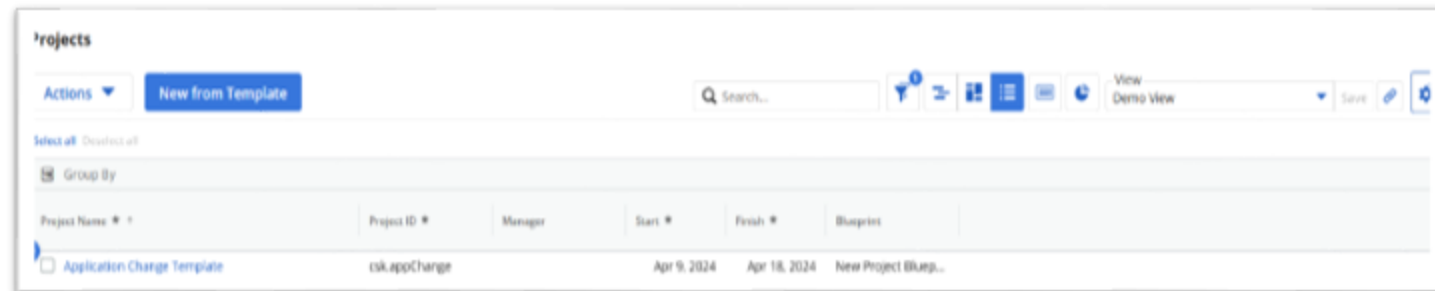


- Views can be shared and **recommended** by administrators as defaults for users



Saving Views and Filters (3)

- Once the View is Saved, End Users may then search for and apply the view applicable to them.
- Instruct users to search for and apply the view using the View menu.



Saving Views and Filters (4)

- The new view will now be applied. Saved Filters with desired fields, filters and flyout configuration.

The screenshot displays a project management interface. At the top, there are filter controls: 'Match Filters' with 'All' and 'Any' options, and 'Add filter groups'. Below this, a filter is applied: 'Template = Yes'. There are buttons for '+ Add filter' and 'Remove all'. Below the filters, there are 'Select all' and 'Deselect all' options.

The main part of the interface is a table with the following columns: Project Name, Project ID, Manager, Start, Finish, and Blueprint. The table contains several rows of project templates. The 'Application COTS Template' row is highlighted in blue, and its 'Finish' date, 'May 18, 2024', is also highlighted.

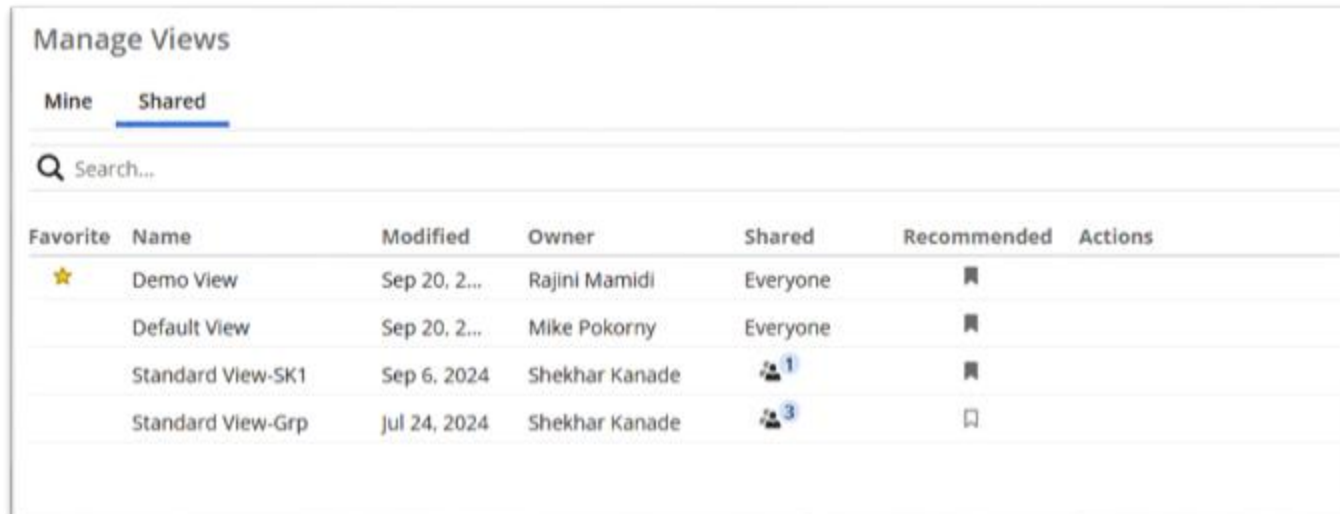
To the right of the table is a 'Columns' panel for the 'Application COTS Template (csk.appCOTS)'. It has a 'Details' tab and a 'Configure' button. The details section shows the following fields:

- Project Name *: Application COTS Template
- Project ID *: csk.appCOTS
- Manager: [Dropdown menu]
- Start *: Apr 9, 2024
- Finish *: May 18, 2024
- Blueprint: Standard Project

Project Name	Project ID	Manager	Start	Finish	Blueprint
<input type="checkbox"/> Application Change Template	csk.appChange		Apr 9, 2024	Apr 18, 2024	New Project Bluep...
<input checked="" type="checkbox"/> Application COTS Template	csk.appCOTS		Apr 9, 2024	May 18, 2024	Standard Project
<input type="checkbox"/> Associations Project	PR00000005	Nopcharoenwong, ...	Aug 5, 2024	Aug 5, 2024	New Project Bluep...
<input type="checkbox"/> Deleted Project Template	PR00000012	Nopcharoenwong, ...	Aug 26, 2024	Aug 26, 2024	Standard Project
<input type="checkbox"/> Infrastructure Deployment Template	csk.infrastructure		Apr 9, 2024	May 13, 2024	Standard Project
<input type="checkbox"/> Major Project Template	csk.majorIT		Apr 9, 2024	Jul 29, 2024	Standard Project
<input type="checkbox"/> TTM Template	ttm-1	Nopcharoenwong, ...	Jun 27, 2024	Jun 27, 2024	Standard Project - ...

Notes on Saved Views

- There is no security right associated with saving views.
- This means that any end user can save a view that others will see. If everyone starts doing this, the list of views can become quite long.
- To limit scrolling through a long list of views, end users may use the Search functionality to search for and more easily locate the view you're asking them to apply.



The screenshot shows the 'Manage Views' interface. It has two tabs: 'Mine' and 'Shared', with 'Shared' selected. Below the tabs is a search bar with a magnifying glass icon and the text 'Search...'. Below the search bar is a table with the following columns: Favorite, Name, Modified, Owner, Shared, Recommended, and Actions. The table contains four rows of data:

Favorite	Name	Modified	Owner	Shared	Recommended	Actions
★	Demo View	Sep 20, 2...	Rajini Mamidi	Everyone	👤	
	Default View	Sep 20, 2...	Mike Pokorny	Everyone	👤	
	Standard View-SK1	Sep 6, 2024	Shekhar Kanade	👤 ¹	👤	
	Standard View-Grp	Jul 24, 2024	Shekhar Kanade	👤 ³	👤	

User Personalizations - Picklists

What are Picklists?

- Picklists are similar to Static Lookups, but can be added by users on the Modern UX without depending on Admins/Clarity Studio.
- They are only visible in Modern UX and under certain cases like Roadmaps or Tasks, the values are specific to that Investment or Roadmap allowing for additional levels of personalization.

Creating a Picklist

- Navigate to an Object's List View that supports Picklists.
- Click on the gear icon and select "Manage Picklists."

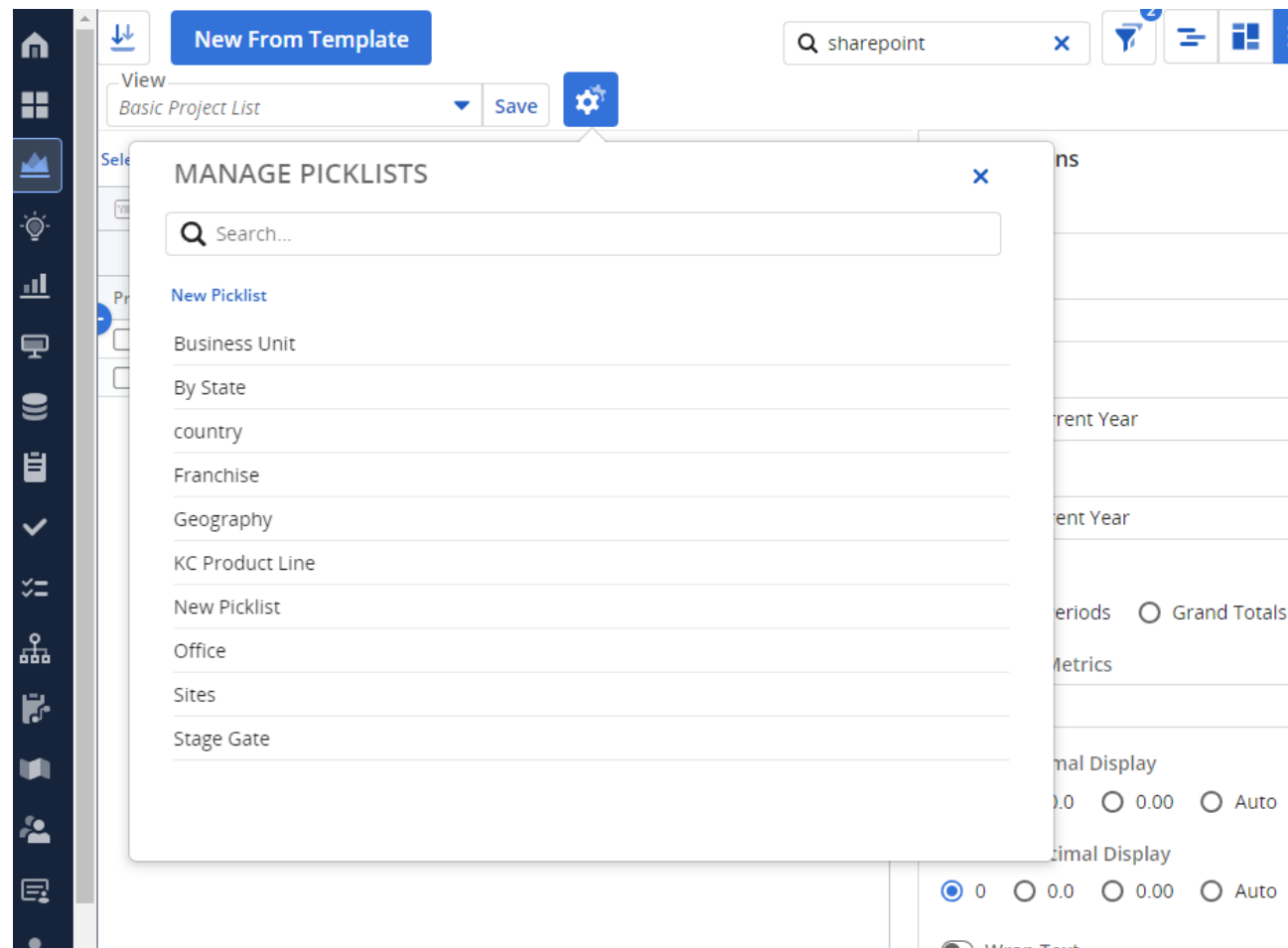
The screenshot shows the Clarity interface for the 'Projects' list view. The main table displays two project entries:

Project Type	Project ID *	Project Name * ↑
<input type="checkbox"/>	PR1332	Regolink SharePoint Sites Imple
<input type="checkbox"/>	PR1315	SharePoint sites enablement

The 'View Options' panel on the right is open, showing settings for the 'Grid' view. The 'Manage Picklists' link is visible at the bottom of the panel.

Creating a Picklist

- Once the “Manage Picklists” popup comes up, select “New Picklist.”



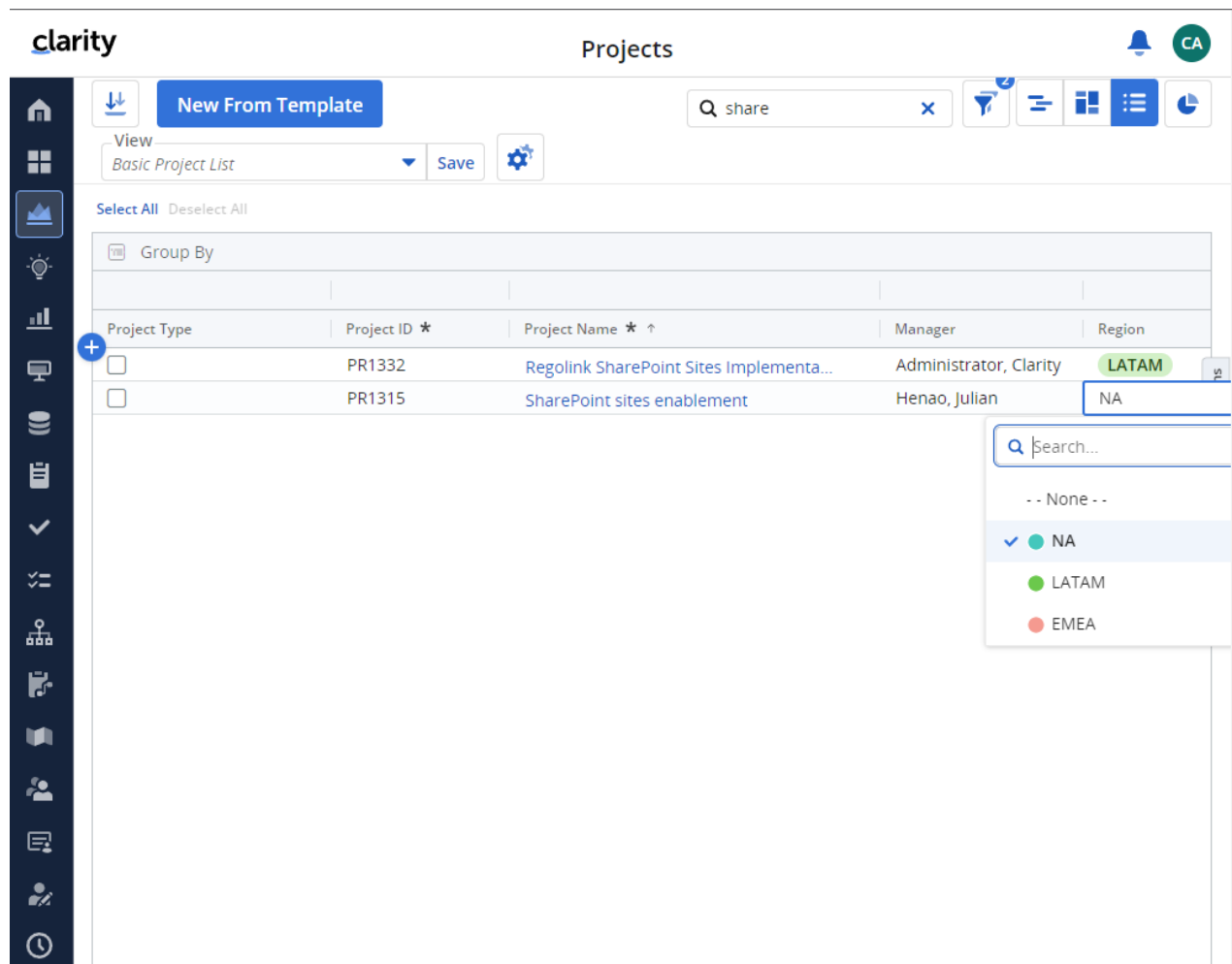
Creating a Picklist

- Provide a “Picklist Name.”
- Provide values by selecting “Add Choice.”
- Additionally, you can choose a color similar to Classic UI buckets.
- Once you are finished click on “Done.”

The screenshot shows the SharePoint 'EDIT REGION' picklist configuration interface. The main window is titled 'EDIT REGION' and contains a 'Picklist Name *' field with the value 'Region'. Below this, there are three choices listed with color-coded buttons: 'Green' for 'LATAM', 'Teal' for 'NA', and 'Red' for 'EMEA'. At the bottom of the main window, there are 'Delete' and 'Done' buttons. To the right of the main window is a 'View Options' panel with various settings like 'Grid', 'Periods', 'Start Period', 'End Period', 'Totals', and 'Money Decimal Display'. The background shows a SharePoint list view with a 'New From Template' button and a search bar.

Creating a Picklist

- The Picklist and values are now available.



The screenshot displays the Clarity Projects interface. The top navigation bar includes the Clarity logo, the title "Projects", and a user profile icon labeled "CA". Below the navigation bar, there is a "New From Template" button and a search bar containing "share". The main content area shows a table with columns for "Project Type", "Project ID", "Project Name", "Manager", and "Region". Two rows are visible: one for "PR1332" managed by "Administrator, Clarity" in the "LATAM" region, and another for "PR1315" managed by "Henao, Julian" in the "NA" region. A picklist menu is open over the "Region" column of the second row, showing options: "-- None --", "NA" (selected), "LATAM", and "EMEA".

Project Type	Project ID *	Project Name * ↑	Manager	Region
<input type="checkbox"/>	PR1332	Regolink SharePoint Sites Implementa...	Administrator, Clarity	LATAM
<input type="checkbox"/>	PR1315	SharePoint sites enablement	Henao, Julian	NA

Introduction to SQL

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 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 like 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 the tables related to that schema will be loaded under it.
 - e.g. Clarity uses NIKU as the default schema in which the related tables reside

Using SQL in Clarity

- Some common uses of SQL are:
 - To extract ad-hoc data
 - As a basis for NSQL in portlet writing
 - To get data within a process for data manipulation
- Clarity Data Model
 - Knowing the Clarity 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 must 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.

Basic SQL Syntax (cont.)

- 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;`

Basic SQL Syntax (cont.)

- 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. Note: 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

Query Exercise #1

- Write a select query to pull all the resources from the system and their associated username.

- a. Start with a select from the resource table

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

- b. Add a join to the user's table

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

- c. Add the username column from the user's table

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

SQL Aliases

- Table Aliases

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

- e.g.,

```
SELECT column_name1, column_name2  
FROM srm_resources res
```

- In the above example “res” is the alias and will be used to reference the table “srm_resources” throughout the rest of the query.

SQL Column Names

- Database columns can be renamed in a SELECT statement as such:
 - *SELECT user_name as user FROM cmn_sec_users*
 - In the above example the column will be displayed in the results as “user”
 - Note that the keyword “as” is implied and therefore not required
- You can also display a column in a more descriptive way with upper/lowercase and spaces. e.g.,
 - *SELECT user_name as “User Name”, email as “Email Address” FROM cmn_sec_users*
 - The double quotes ARE REQUIRED when you want to include spaces and lowercase text
 - The “as” keyword is optional here as well

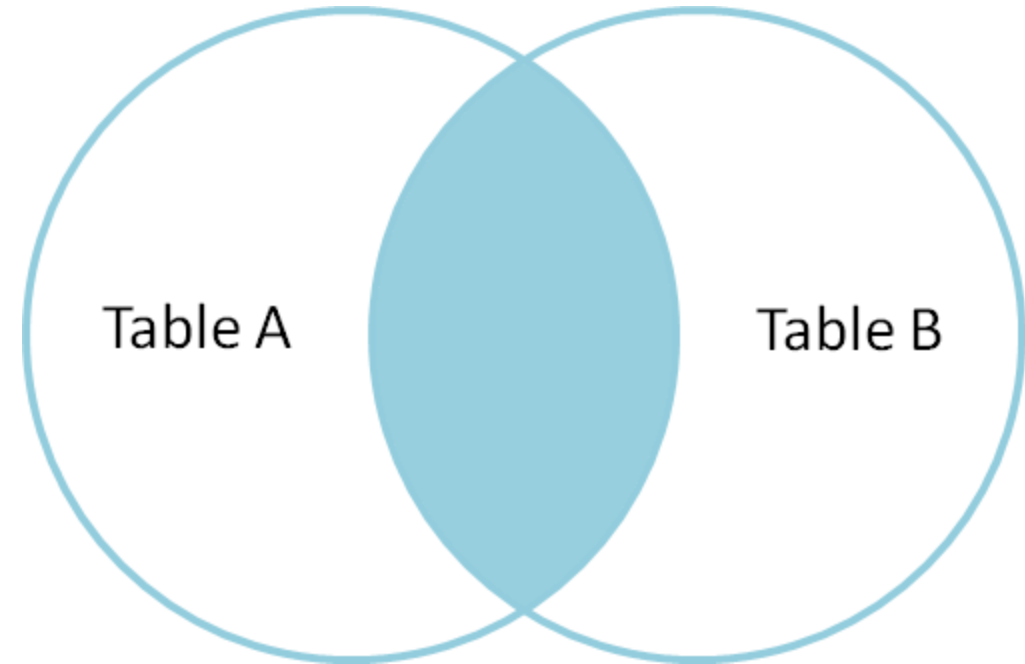
SQL Joins

- 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 Joins

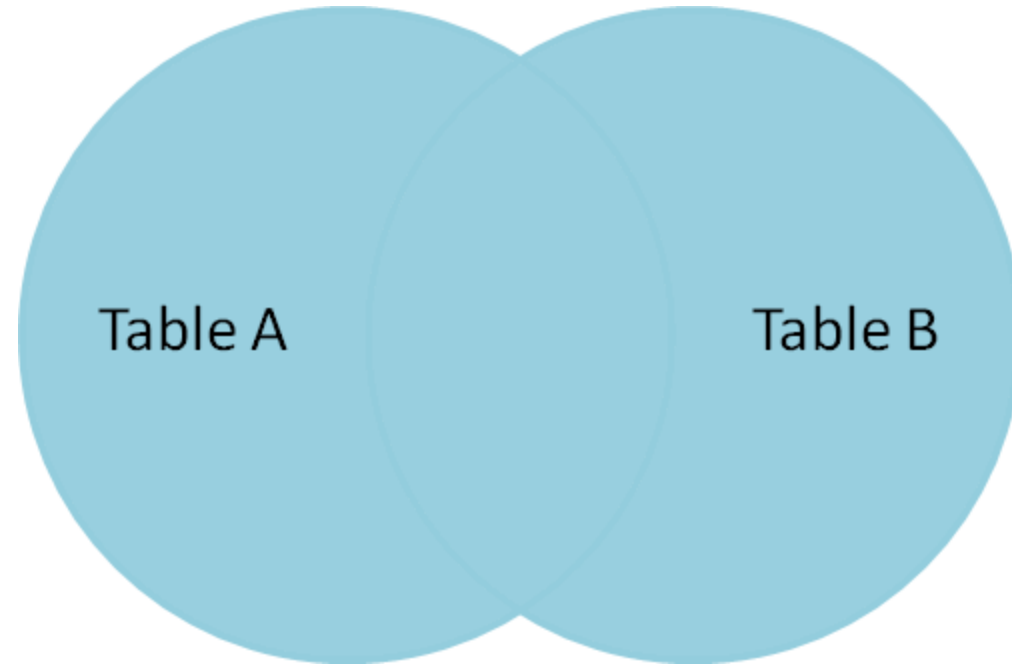
- Inner Join



```
SELECT *  
FROM TableA A  
INNER JOIN TableB B ON A.Key = B.Key
```

SQL Joins

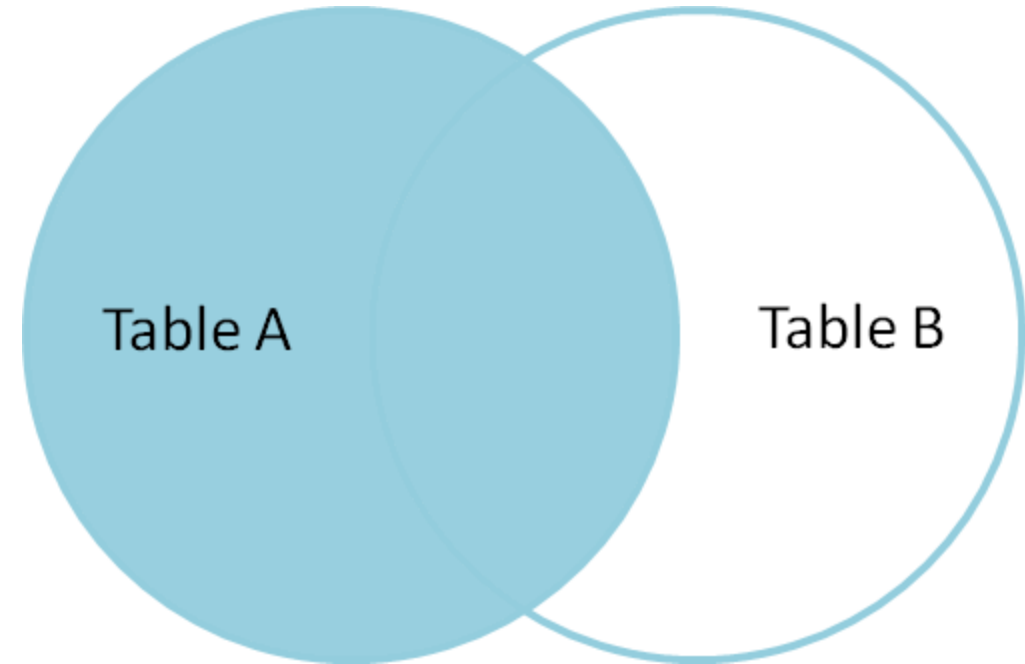
- Outer Join



```
SELECT *  
FROM TableA A  
FULL OUTER JOIN TableB B ON A.Key = B.Key
```

SQL Joins

- LEFT OUTER JOIN



```
SELECT *  
FROM TableA A  
LEFT OUTER JOIN TableB B ON A.Key = B.Key
```

More SQL Concepts

- **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;*

More SQL Concepts

- UNION Statement

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order
- Syntax:
 - *SELECT column_name(s) FROM table1*
 - *UNION*
 - *SELECT column_name(s) FROM table2;*

SQL Functions

- Functions are used to perform processing on string and numeric data
- Types
 - Aggregate
 - Return a single value, calculated from values in a column
 - Examples: 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
 - Examples: ROUND, UCASE (Uppercase), LCASE (Lowercase), FORMAT

SQL Subqueries

- Subqueries are nested queries (a query within a query)
- They must be enclosed in parentheses
- Subqueries can be included within SELECT, INSERT, UPDATE or DELETE statements
- Example:
 - *SELECT full_name FROM srm_resources
WHERE user_id IN (SELECT manager_id FROM srm_resources)*

Query Exercise #2

- Write a query to pull all active projects starting in 2018 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 2018

```
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('2018-01-01','yyyy-mm-dd')
```

Query Exercise #2 (cont.)

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
```

Query Exercise #2 (cont.)

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

Lookups, Queries and Portlets

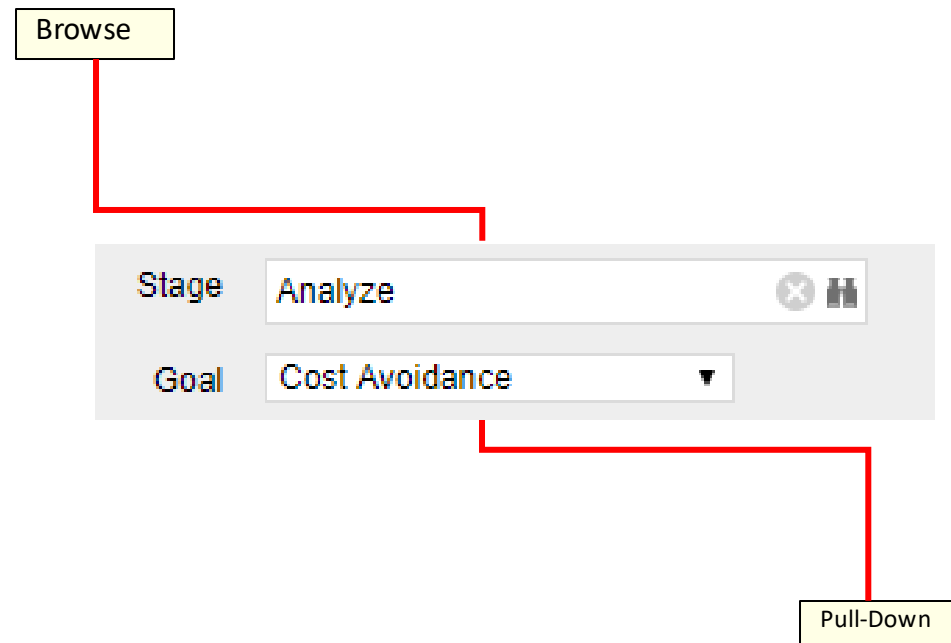
Lookups

- 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 – Static List

- 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 Lookup

- 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 (cont.)

- 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 (cont.)

- Dynamic Query Lookup.
 - Use this type of lookup to capture data from the Clarity database in real time to populate the drop-down or browse lists.
 - These lookups provide the most up-to-date values possible and are often used inside browse windows.
 - Eliminates the need to maintain a list since the values are dynamically pulled from tables within the database (e.g., list of all resources).
 - Dynamic queries are written in NSQL – more on this in a couple of slides.

Exercise #2: Dynamic Query Lookup

- 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

- A Query is created and managed in Studio (Administration->Studio->Queries).
- Queries extract data from the system for use in portlets.
- Similar to Dynamic Lookups, Queries are also written in NSQL.
- NSQL is a version of SQL specific Clarity; 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 (cont.)

- 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 (cont.)

- 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 Query

- 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.

Queries (cont.)

- NSQL Constructs / Cross Platform
 - @BROWSE-ONLY@
 - SECURITY –
 - @WHERE:SECURITY:ISSUE:i.odf_pk@
 - @WHERE:SECURITY:PROJECT:inv.id@
 - User and Locale –
 - @WHERE:PARAM:USER_ID@
 - @WHERE:PARAM:LANGUAGE@
 - @WHERE:PARAM:LOCALE@
 - Cross Platform / Functions
 - @UPPER@, @SYSDATE@, @NVL@, @SUBSTR@, @DBUSER@, @+@
 - @COP_DATE_TRUNC_FCT@
 - @WHERE:PARAM:USER_ID@
 - @COALESCE@

Portlets

- What is a Portlet?

- Portlets are snapshots of Clarity data and can consist of grids, graphs, or snippets of HTML.
- Portlets do not replace Clarity reports but are often preferred due to ease of use and access.
- Portlets obtain information and business intelligence from Clarity, 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.

Portlet Types

- **Chart Portlet** – A graphical view of Clarity 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 Clarity 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 Clarity data using imported Xcelsius visualizations.

Portlet Data Sources

Query-based (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

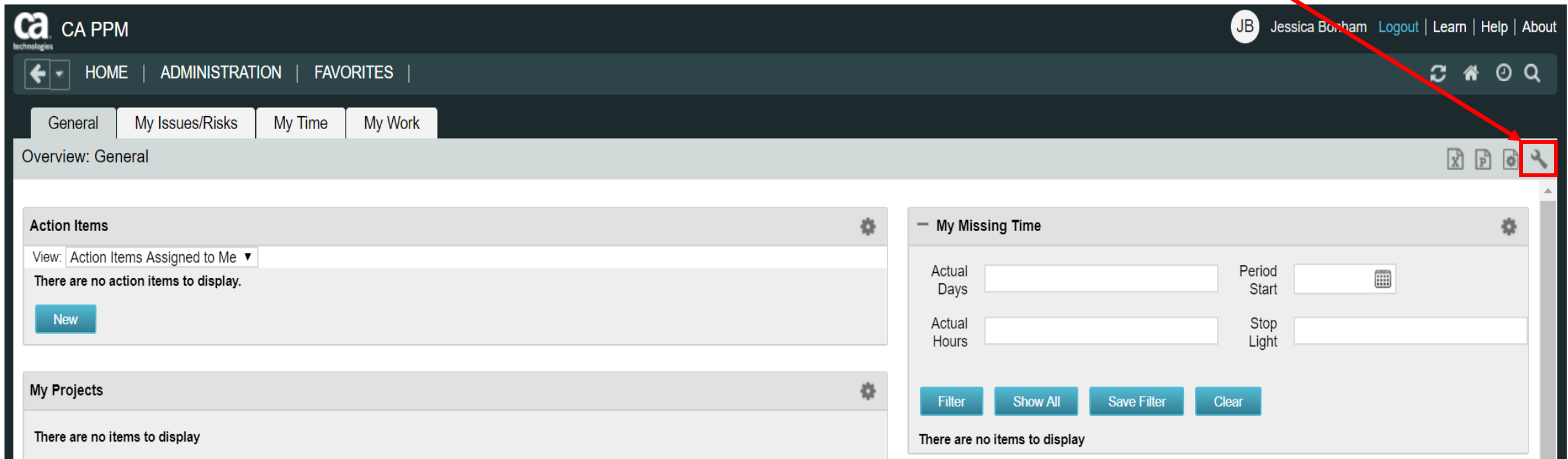
- Portlets created using an Object instead of a query to define the data set gathered.
- Pros:
 - Customizable
 - Security
 - In-Line Editing
 - Dynamic
 - Time Scaled Values
- Cons:
 - Multiple Objects Difficulty
 - No Custom Logic

Exercise #4: Basic Grid Portlet

- 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?

Exercise #4: Basic Grid Portlet (cont.)

- Use the Manage My Tabs link on the toolbar at the far right to add the portlet to your home page:



The screenshot displays the CA PPM user interface. At the top left, the logo 'ca technologies' and 'CA PPM' are visible. The top right shows the user 'JB Jessica Bonham' with links for 'Logout | Learn | Help | About'. Below the navigation bar, there are tabs for 'General', 'My Issues/Risks', 'My Time', and 'My Work'. The main content area is titled 'Overview: General' and contains several portlets: 'Action Items' (with a 'View' dropdown set to 'Action Items Assigned to Me' and a 'New' button), 'My Projects' (with 'There are no items to display'), and 'My Missing Time' (with input fields for 'Actual Days', 'Actual Hours', 'Period Start', and 'Stop Light', and buttons for 'Filter', 'Show All', 'Save Filter', and 'Clear'). A toolbar at the top right of the portlet area contains icons for 'Print', 'Refresh', and 'Manage My Tabs' (a wrench icon), which is highlighted with a red box and a red arrow pointing to it.

Introduction to Workflow (Processes)

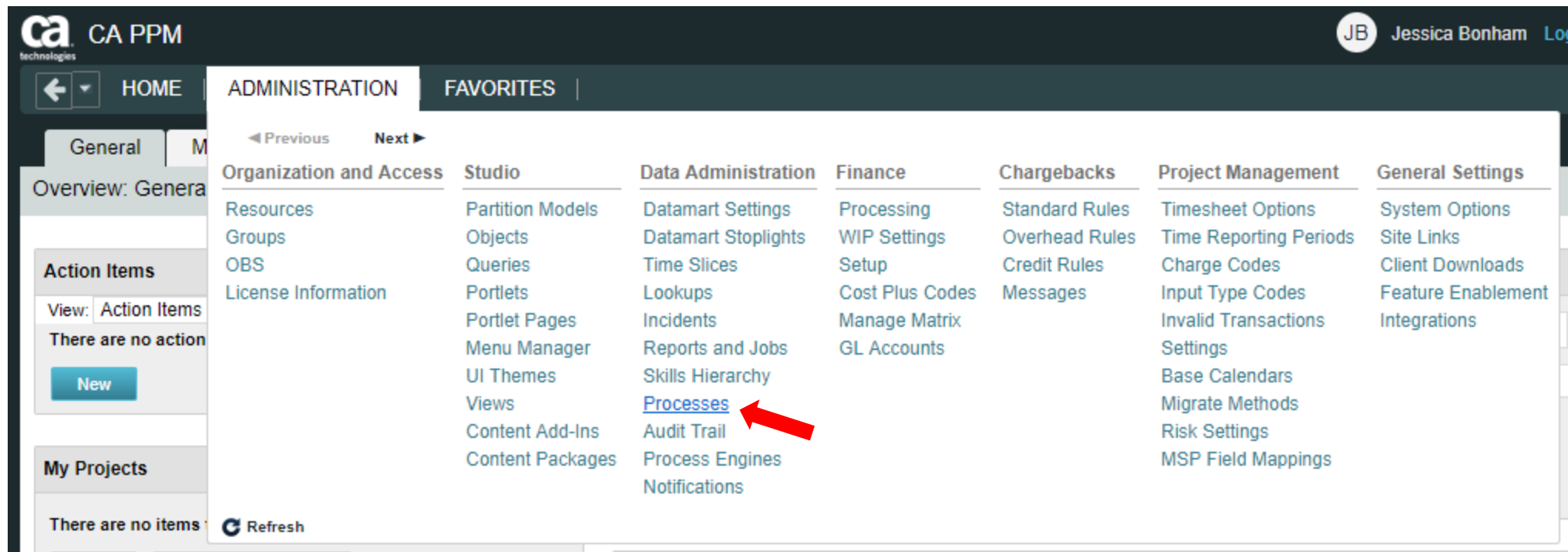


Processes: Overview

- 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 Start step, a Finish step, and - if desired - any number of steps in between
 - Each step performs one or more actions that move the process toward completion
 - Processes use pre- and post-conditions to connect the steps
- Clarity provides stock processes that you can use to:
 - Approve documents
 - Approve timesheets
 - Approve ideas

Processes: Overview (cont.)

- Process definitions are accessed via the Administration menu:



The screenshot shows the CA PPM Administration menu. The 'ADMINISTRATION' tab is selected, and the 'Processes' link is highlighted with a red arrow. The menu is organized into several columns:

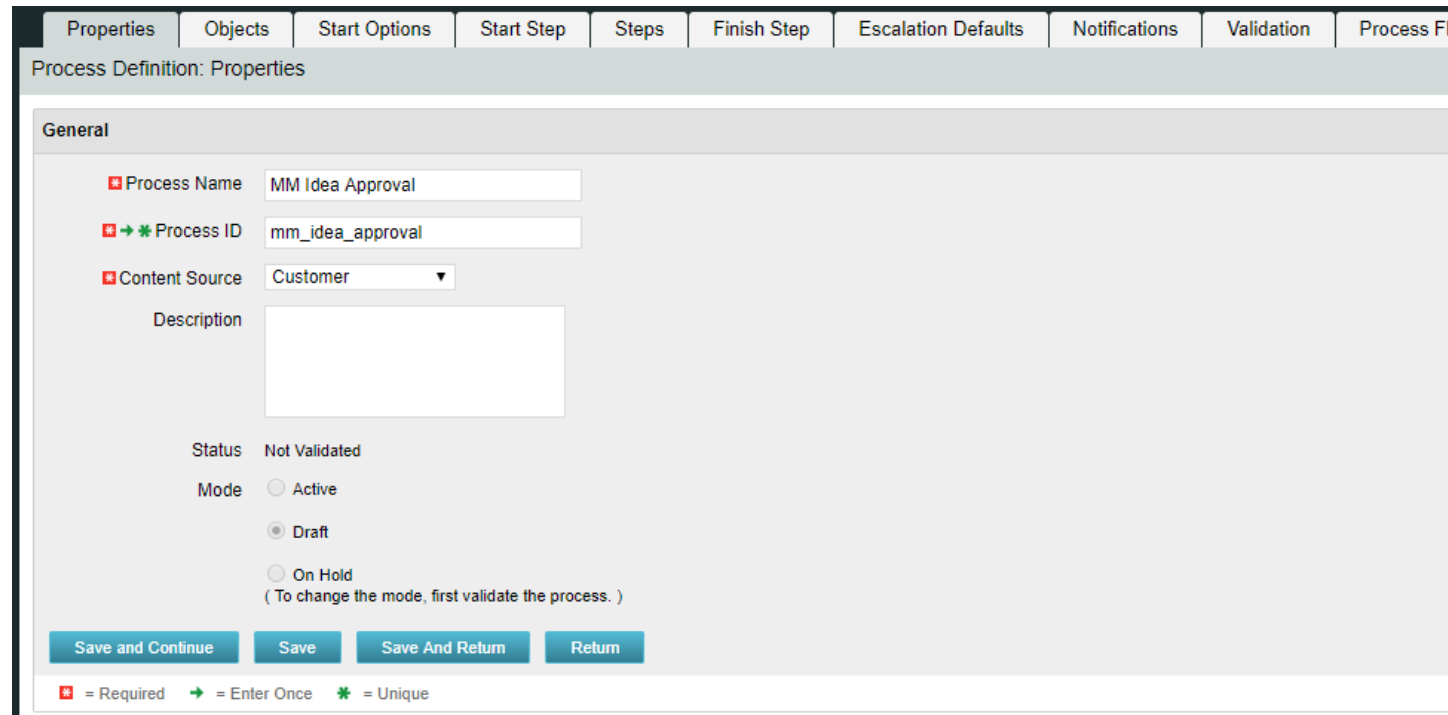
Organization and Access	Studio	Data Administration	Finance	Chargebacks	Project Management	General Settings
Resources	Partition Models	Datamart Settings	Processing	Standard Rules	Timesheet Options	System Options
Groups	Objects	Datamart Stoplights	WIP Settings	Overhead Rules	Time Reporting Periods	Site Links
OBS	Queries	Time Slices	Setup	Credit Rules	Charge Codes	Client Downloads
License Information	Portlets	Lookups	Cost Plus Codes	Messages	Input Type Codes	Feature Enablement
	Portlet Pages	Incidents	Manage Matrix		Invalid Transactions	Integrations
	Menu Manager	Reports and Jobs	GL Accounts		Settings	
	UI Themes	Skills Hierarchy			Base Calendars	
	Views	Processes			Migrate Methods	
	Content Add-Ins	Audit Trail			Risk Settings	
	Content Packages	Process Engines			MSP Field Mappings	
		Notifications				

Processes: Design Basics

- Creating a process involves the following:
 - Define the Process properties
 - Determine if it will be associated with a Clarity object
 - Set Start option
 - Determine/Define the number of steps required
 - Create Actions
 - Create pre- and post-conditions if needed
 - Validate the process
 - Activate the process

Processes: Design Basics (cont.)

- Define Process Properties.
 - Set the name and ID of the process
 - Processes are always created in “Draft” mode (default)



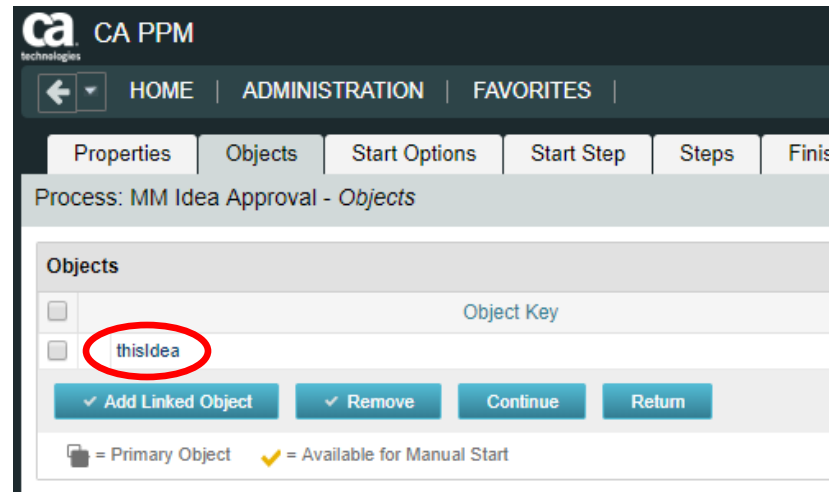
The screenshot shows the 'Process Definition: Properties' dialog box, specifically the 'General' tab. The dialog has a tabbed interface at the top with the following tabs: Properties, Objects, Start Options, Start Step, Steps, Finish Step, Escalation Defaults, Notifications, Validation, and Process Flow. The 'General' tab is active and contains the following fields and options:

- Process Name:** MM Idea Approval (Required field)
- Process ID:** mm_idea_approval (Required and Unique field)
- Content Source:** Customer (Dropdown menu)
- Description:** (Empty text area)
- Status:** Not Validated
- Mode:** Draft (Selected), Active, On Hold (To change the mode, first validate the process.)

At the bottom of the dialog, there are four buttons: Save and Continue, Save, Save And Return, and Return. A legend at the bottom left explains the field markers: a red square with a white 'x' for Required, a green arrow for Enter Once, and a green asterisk for Unique.

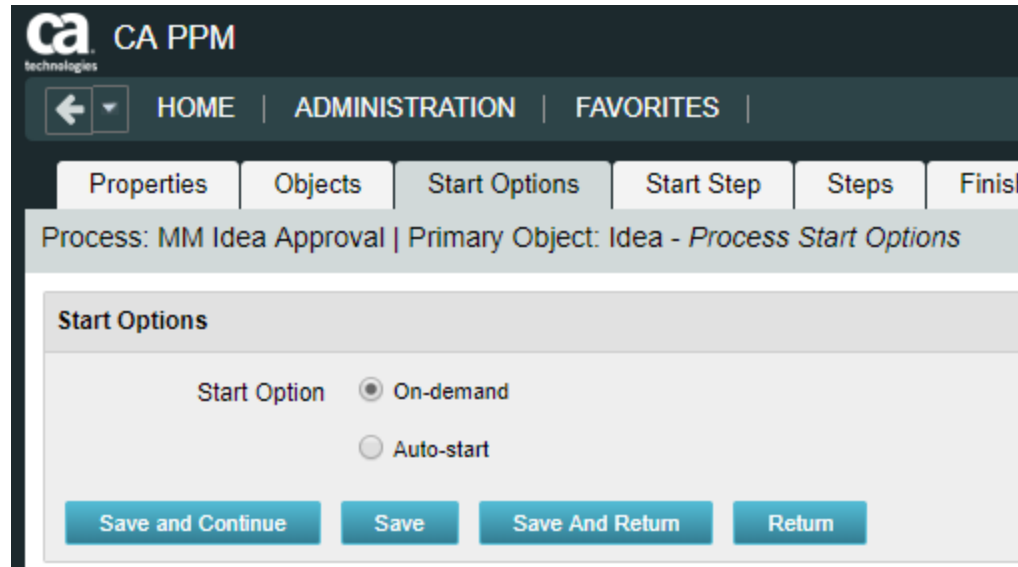
Processes: Design Basics (cont.)

- Associate the process with a Clarity object if:
 - The process should be triggered by a user update (or create)
 - The process will include system actions, manual actions or sub-processes
- In this example the process is associated with Ideas:



Processes: Design Basics (cont.)

- Set Start option.
 - On Demand: process will 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: process will be kicked off when a user updates or creates an object instance



The screenshot displays the CA PPM web interface. At the top left is the CA Technologies logo. A navigation bar includes a back arrow, 'HOME', 'ADMINISTRATION', and 'FAVORITES'. Below this is a tabbed interface with tabs for 'Properties', 'Objects', 'Start Options', 'Start Step', 'Steps', and 'Finish'. The current view is 'Start Options' for the process 'MM Idea Approval' and primary object 'Idea - Process'. The 'Start Options' section contains two radio buttons: 'On-demand' (selected) and 'Auto-start'. At the bottom are four buttons: 'Save and Continue', 'Save', 'Save And Return', and 'Return'.

Processes: Design Basics (cont.)

- 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 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 Finish 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 (cont.)

- Splits / Joins determine the direction of the process flow
 - A Split branches into multiple directions
 - A Join brings multiple branches back together

Split Types

- Serial
- Parallel
- Decision Point
- Multi choice

Join Types

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

Processes: Design Basics (cont.)

Example of Post-conditions and Split Type:

Process: Assign Incidents | Step: Acquire Incident - Step Details

General

Step Name: Acquire Incident

Step ID: AcquireIncident

Status: Validated

Description:

Group: [--Select--]

Raise a Warning After: Days

Milestone:

Notifications

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

Send Notification To:

Notify Owner:

Escalation

There is no escalation rule setup to display.

Actions

Name	Description
Send acquire incident action item	Please review and acquire incident.

Referring Steps

Assign IT Worker.Incident Not Resolved

Pre-conditions

Join Type: None

There is no precondition to display.

Post-conditions

Split Type: Decision Point (XOR)

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

If ...	Then Go To	Description
(Action Item.AcquireIncident.AcquireIncident Number of assignees with Status Done >= 1)	Work On Incident	Incident Acquired
(Action Item.AcquireIncident.AcquireIncident Number of assignees with Status Rejected >= 1)	Incident Escalated	Incident Not Acquired



Processes: Design Basics (cont.)

Example of Pre-condition Join Type:

Process: Assign Incidents | Step: Incident Escalated - Step Details

General

Step Name: Incident Escalated

Step ID: IncidentEscalated

Status: Validated

Description:

Group: [--Select--]

Raise a Warning After: Days

Milestone:

Referring Steps

Resolution Verification, Acquire Incident

Pre-conditions

Join Type: Merge (XOR)

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
	Incident Not Resolved	

Notifications

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

Send Notification To:

Notify Owner:

Escalation

There is no escalation rule setup to display.

Actions

Name	Description
Escalate Incident	Set incident status to Escalated.

Processes: Design Basics (cont.)

- A step **Action** can be one of the following:
 - Manual Action
 - Example(s): Actions Items
 - With manual actions you can include some object data using variables
 - System Action
 - Example(s): Lock/Unlock Attributes, Set Attribute Values, and certain System Operations (copy a financial plan from a template)
 - Run a Job
 - Example(s): Post Timesheets
 - Custom Script (run a GEL script)
 - Example(s): Notification Scripts
 - Subprocess
 - Sub processes are invoked as embedded processes within the context of the current process

Processes: Design Basics (cont.)

Example of step Action:

Process: Assign Incidents | Step: Assign IT Worker - Step Details

General

Step Name: Assign IT Worker

Step ID: AssignITWorker

Status: Validated

Description:

Group: [--Select--]

Raise a Warning After: Days

Milestone:

Referring Steps

Start

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
	Acquire Incident	Incident Assigned

Notifications

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

Send Notification To:

Notify Owner:

Escalation

There is no escalation rule setup to display.

Actions

Name	Description
Run assign incident job	Run assign incident job.

Processes: Design Basics (cont.)

- 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: Assign Incidents - Process Validation

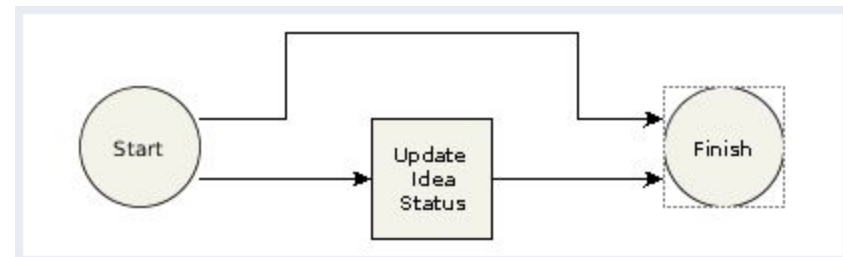
Validation Object	Status
Record Effort	Validated
Resolution Verification	Validated
Incident Not Resolved	Validated
Incident Escalated	Validated
Process	Validated
BPM-0664: The process contains cycle. Cycle steps: Acquire Incident Work On Incident Record Effort Resolution Verification Incident Escalated Incident Not Resolved Entry steps: Acquire Incident Exit steps: Resolution Verification	Re-validation Required
BPM-0664: The process contains cycle. Cycle steps: Acquire Incident Incident Escalated Incident Not Resolved Entry steps: Acquire Incident Exit steps: Acquire Incident	Re-validation Required

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

Validate Activate Process Validate All and Activate Continue Return

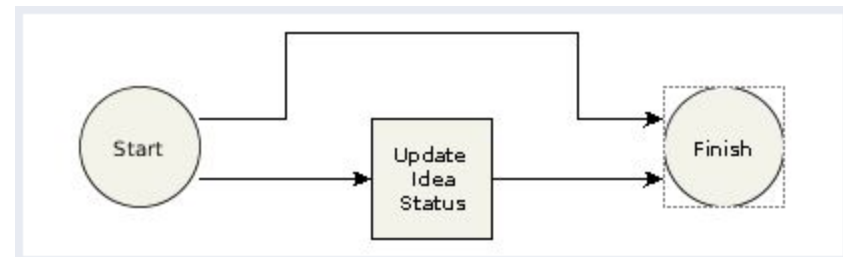
Processes: Exercise

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 the steps
8. Ensure the conditions point to the correct step
9. Validate and Activate the process
10. Now TRY IT OUT!! 😊



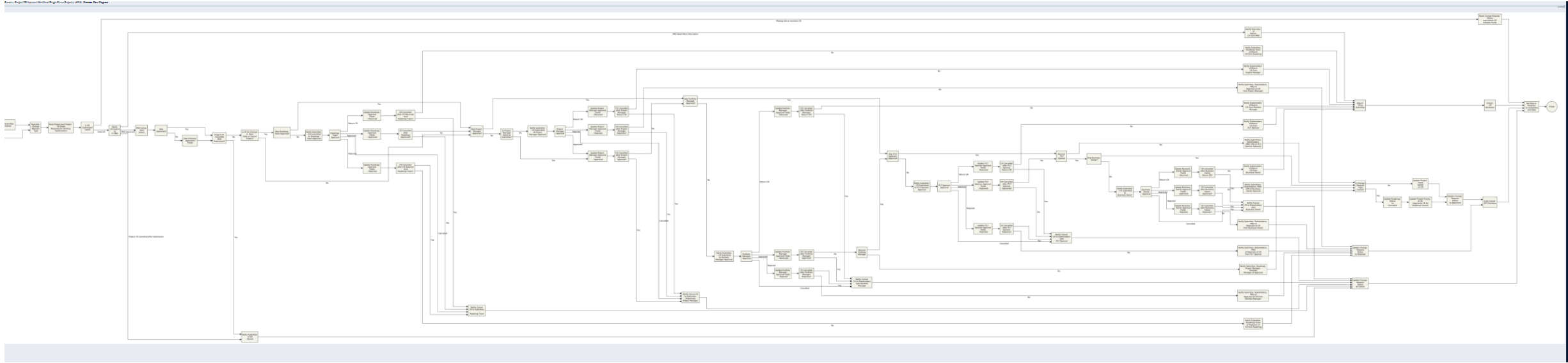
Processes: Exercise

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 the steps
8. Ensure the conditions point to the correct step
9. Validate and Activate the process
10. Now TRY IT OUT!! 😊



Process Don'ts

1. Keep processes as simple as possible
2. Monitor high volume or long running processes
3. Callings jobs and processes should be used carefully
4. Ensure error handling and logs are incorporated when using scripts



XOG



XOG: Introduction

- Clarity has a Web Services interface called XML Open Gateway (XOG)
- Although REST API is being actively invested on, XOG has functionality that hasn't been ported to the REST API
- What XOG Does
 - Export/Import data (e.g., Projects, Resources)
 - Export/Import configurations (e.g., Lookups, Portlets, Processes)
- Execution Options
 - Run XOG Client remotely (from another computer or desktop)
 - Run XOG Client from Clarity using HTML portlet (available in RegoXchange)
 - Issue XOG request from a GEL script
- Common uses
 - Updating large numbers of records
 - Moving configurations (or data) from one environment to another
 - Making updates that can't be done via the UI (e.g., locked fields)

XOG (Remote): 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 7 Runtime Environment version 1.7.0_25
 - Oracle Java 8 Runtime Environment version 1.8.0
- Clarity
- XOG client version that matching the Clarity
- Clarity user with XOG Global Rights
 - Administration – XOG
 - Administration – Access
 - XOG rights for individual objects (for example, Resource, Project, OBS)

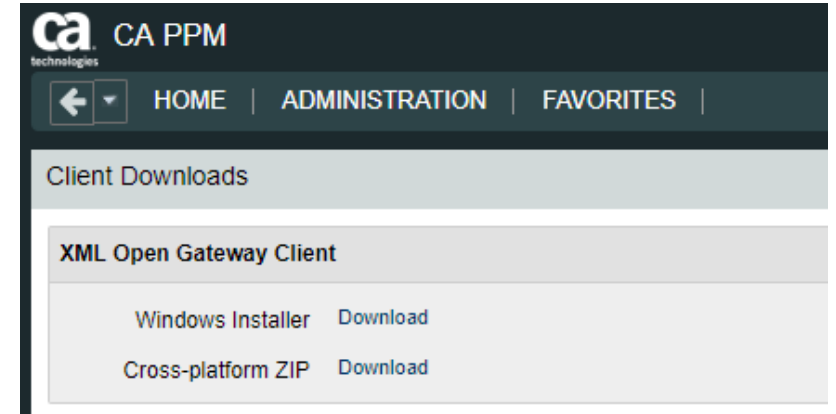
XOG (Remote): Installation

- Download compatible JRE from <http://www.java.com/en/>
 - Installation folder: C:\jre8\
- Set the environment variables:
 - **JAVA_HOME**=C:\jre8
 - **PATH**=;%JAVA_HOME%\bin
- Test for Java using a command prompt
 - java -version

```
C:\> Command Prompt
C:\> java -version
java version "1.8.0_172"
Java(TM) SE Runtime Environment (build 1.8.0_172-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.172-b11, mixed mode)
C:\>
```

XOG (Remote): Installation - continued

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



```
C:\>cd xog/bin
C:\XOG\bin>xog
-----
PPM XML Open Gateway ( version: 15.4.0.270 )
-----
Usage: xog

login [user/passwd@host:port] Create new active session.
logout                          Close active session.
call <file>                      Invoke given XOG request.
verbose <on|off>                 Turn on verbose error logging.
output <console|path>           Output to console or file.
exit                             Logout and exit the shell.
?                                Display this help information.

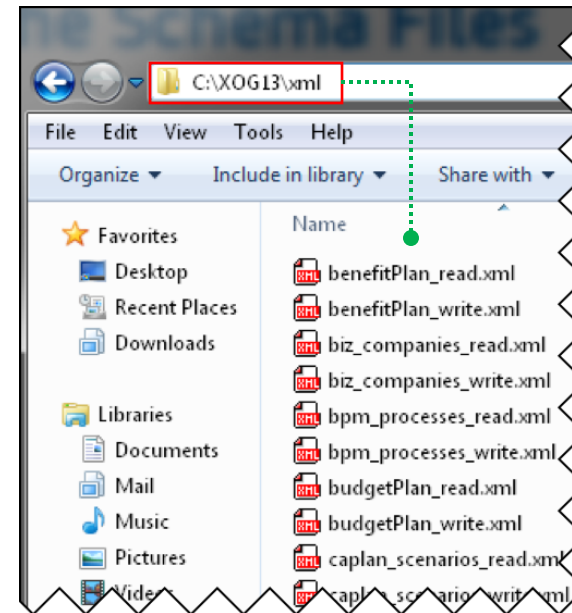
Examples:
  call xml/cmn_users_read.xml
>
```

XOG (Remote): Verify Connectivity

- Use XOG client shell commands to test the connection between the XOG client and Clarity server as follows:
 1. Open a command prompt
 2. Type **cd C:\xog\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 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 Clarity 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)



XOG XML Read Files

- Use the XML Read files to export a specific item from Clarity.
- Each Read XML file contains the following structure:
 - **Header:** Supported Clarity 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)

XOG XML Read Files

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>
```

XOG XML Read Files

- 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)

XOG (Remote): 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

XOG (Remote): Properties File

- 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

```
dev.txt
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
```

XOG (Remote): Exercise #1 – Export Data

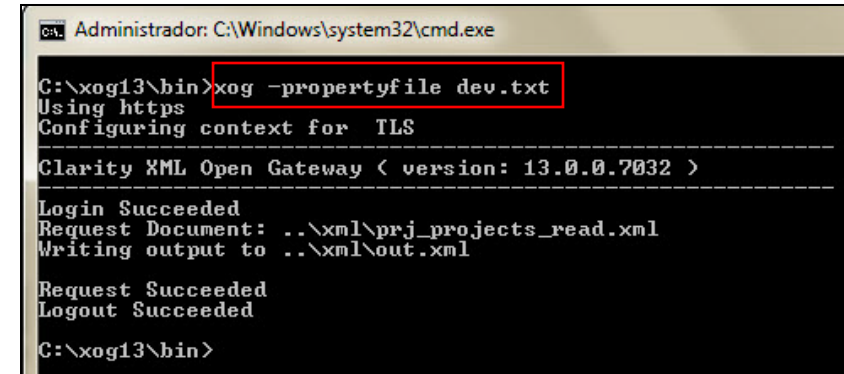
- Submit a XOG Read request using the XOG client as follows
 - Create a .properties file with the default values for the XOG parameters
 - Create an input XML file with the necessary header information, arguments, and query filters
 - Navigate to the “bin” folder under the XOG client installation folder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

XOG (Remote): Exercise #1 – Export Data

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>
```

XOG: Exercise #1 – Export Data

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>
```


XOG: Exercise #1 – Export Data

Output File

```
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.w3.org/2001/XMLSchema-instance" >
  <Header action="write" externalSource="NIKU" objectType="project" version="13.0.0.7032"/>
  <Projects>
    <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="USD"
equipmentExchangeRateType="AVERAGE" evCalcMethod="0" expenseExchangeRateType="AVERAGE" finish="2012-04-26T03:00:01"
materialExchangeRateType="AVERAGE" name="Application Change Template" openForTimeEntry="true" pageLayoutCode="data
csk.appChange" requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-04-09T08:00:00"
useSystemDefinedTotalCostOfCapital="true">
    <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="USD"
expenseExchangeRateType="AVERAGE" financialStatus="0" finish="2007-04-26T10:40:00" format="11" materialExchangeRateType="AVERAGE"
name="Application COTS Template" openForTimeEntry="true" pageLayoutCode="data" requiredForScenarios="false"
setBudgetValuesEqualToPlannedValues="true" start="2007-04-09T08:00:00" useSystemDefinedTotalCostOfCapital="true">
    <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="USD"
expenseExchangeRateType="AVERAGE" financialStatus="0" finish="2007-06-25T17:00:00" format="11" materialExchangeRateType="AVERAGE"
name="IT Infrastructure Deployment Template" openForTimeEntry="true" pageLayoutCode="data" requiredForScenarios="false"
setBudgetValuesEqualToPlannedValues="true" start="2007-05-01T08:00:00" useSystemDefinedTotalCostOfCapital="true">
    <Project active="true" approved="false" approvedForBilling="1" asOf="2007-03-15T00:00:00" assgnPool="0"
equipmentExchangeRateType="AVERAGE" evCalcMethod="0" expenseExchangeRateType="AVERAGE" finish="2007-03-12T06:00:00"
materialExchangeRateType="AVERAGE" name="New IT Project" openForTimeEntry="true" pageLayoutCode="data"
requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-03-12T06:00:00" useSystemDefinedTotalCostOfCapital="true">
  </Projects>
  <XOGOutput>
    <Object type="project"/>
    <Status state="SUCCESS"/>
    <Statistics failureRecords="0" insertedRecords="0" totalNumberOfRecords="4" updatedRecords="0"/>
    <Records/>
  </XOGOutput>
</NikuDataBus>
```

XOG: XML Write Files

- Use the XML Write files to import a specific object into Clarity
- Each Write XML file contains the following structure:
 - **Header:** Supported Clarity 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

XOG: Creating Your Own XML

- XML can be built manually or using a variety of tools including:
 - GEL scripts (both remotely and in a process)
 - Other programming languages (Perl, Python, JavaScript, etc.)
 - MS Mail Merge
 - Excel formulas
 - 3rd party tools (E.g., Kettle)

XOG: Required Fields

- XOG will return an error if any required fields are missing
- Remove any other fields that you don't need
- Required for Resource updates:
 - resourceId, externalId, firstName, lastName, emailAddress
 - resourceType is not required but generates a warning and “defaults” to LABOR
 - employmentType is not required but generates a warning and “defaults” to EMPLOYEE
- Required for Project updates:
 - projectID, name
- Field names are case-sensitive!!

XOG: Special Characters

- 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	&	&
Left angle bracket	<	<
Right angle bracket	>	>
Straight quotation mark	"	"
Apostrophe	'	''

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

XOG: Special Characters

Escaping special characters example:

```
<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>
```

XOG (UI Client): Exercise #2

Set Financial Status to “Closed” on a project:

```
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_project.xsd">  
  <Header action="write" externalSource="NIKU" objectType="project" version="8.0"/>  
  <Projects>  
    <Project financialStatus="C" name="Test Project" projectID="PR123456">  
      </Project>  
    </Projects>  
</NikuDataBus>
```

Open a Resource for Time Entry:

```
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_resource.xsd">  
  <Header action="write" externalSource="NIKU" objectType="resource" version="15.9.0.294"/>  
  <Resources>  
    <Resource employmentType="CONTRACTOR" externalId=" " resourceId="ABC2436" resourceType="LABOR">  
      <PersonalInformation emailAddress="dave.lord@regoconsulting.com" firstName="Dave" lastName="Lord"/>  
      <ManagementInformation availability="8" openForTimeEntry="false"/>  
    </Resource>  
  </Resources>  
</NikuDataBus>
```

XOG (UI Client): Common Issues

- The UI Client can be difficult to troubleshoot as some issues do not generate an error, but instead simply cause the HTML portlet to hang
- Some common scenarios that hang the XOG Client portlet:
 - Missing tags
 - Malformed XML, syntax errors
 - Un-escaped special characters (common ones are & and “)
- A large XOG request (hundreds of records) can take a long time to complete, and the UI Client does not indicate progress and can appear to be hung
- Do not run large XOG requests during business hours as it can impact performance

XOG (Remote): Exercise #3 – 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

XOG (Remote): Exercise #3 – Import Data

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>`

XOG (Remote): Exercise #3 – Import Data

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>`

XOG: Common Errors

Error	Description
Login Failed	Verify username and password are correct by accessing Clarity 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 (Clarity 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.

XOG: 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)
 - FOXE (Freeware)
- 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 Clarity server and XOG client versions match
- Other tools like Postman can also be used to interact with XOG

REST API



REST: Introduction

- Representational State Transfer
- Architectural principals to define a web service organized around Resources (URIs)
- Primary message format is JSON
- Future of Clarity web services
- MUX is built entirely on REST
- Simpler to use than SOAP
- More granular

REST: When to use

- Use
 - Create, update, or delete targeted data within Clarity
 - To update data within GEL scripts
 - Update data in another application
 - To update CITs (cannot use XOG)
- Do Not / Cannot Use
 - Studio Configuration – Still requires XOG
 - Not all objects are REST enabled
 - Pull large volumes of data from Clarity
 - Data Extracts
 - Pull data to a Data Warehouse

REST: Describe

- View REST resources: Administration -> System Options

System Options	
Data Warehouse Database Schema	PPM_DWH
API	
REST API Status	Enabled
API URL	http://localhost/ppm/rest/v1
API Documentation URL	http://localhost/niku/rest/describe/index.html

- Describe and test endpoints

Clarity PPM REST API

Clarity PPM only supports REST APIs that are listed below.

Auth : Login, Logout, Tokens

Show/Hide | List Operations | Expand Operations

GET	/apiClients	Retrieve apiClients.
POST	/apiClients	Creation of apiClients.
GET	/apiClients/{apiClientsInternalId}	Retrieve apiClients.
PATCH	/apiClients/{apiClientsInternalId}	Partial object update of apiClients.
PUT	/apiClients/{apiClientsInternalId}	Full object update of apiClients.

REST: Challenges

- No GEL tags to support REST (ie: Xog -> soapInvoke)
- Functionality still catching up with XOG i.e. Financials, Users, Resources, etc.
- Have to code using java
 - Java URLConnection
 - Apache HTTP Libraries

REST: More Details

- Check out the presentation from Ben and James "Using REST APIs"
- Broadcom documentation - <https://techdocs.broadcom.com/us/en/ca-enterprise-software/business-management/clarity-project-and-portfolio-management-ppm-on-premise/16-0-1/reference/Clarity-REST-APIs/rest-api-documentation-for-authorized-developers.html>
- Postman - <https://www.postman.com>
- GSON - <https://github.com/google/gson/blob/master/UserGuide.md>

Surveys

Please take a few moments to fill out the class survey.
Your feedback is extremely important for future events.



Questions?



Thank You For Attending Rego University

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 Provider = **Rego Consulting**
- Class Name = **regoUniversity**
- Course **Description**
- 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@regoconsulting.com



Website

www.regouniversity.com