K2® | LEARNING

K2 smartforms

Course Handbook

Version: 3.1

Date: September 2016

"Terms Of Use" on page 244

Table of Contents

K2 smartforms builder

Course Overview

Estimated Duration: about 10 hours
Version: 3.1 (What's New)
Publish Date: September 2016
Based on product version: 4.7 (Later versions of K2 may have additional features or functionality not covered in this learning module)

Course Description

The *K2 smartforms Builder* training course will enable participants to create K2 business applications using K2 smartforms as the user interface. During this course, participants will learn how to build user interfaces with K2 smartforms and then integrate these forms with K2 workflows and K2 SmartObjects.

Who should take this course

This training course is intended for users that will be building SmartForm-based business applications on the K2 platform without writing code. Typical roles that fit this description include analyst developers, business analysts, power users and application designers. This course is also suitable for .NET developers who may be building applications with SmartForms and extending SmartForms with custom components. K2 Solution Architects may also find value in this course.

Note
Participants should have completed the “K2 blackpearl Core” training course before starting this training course, since this course builds on the foundational knowledge of K2 gained in the Core course.

This course is a 200-level course. See additional requirements in the Course Prerequisites section.

What you’ll learn

During this course, participants will learn:

- How to build simple and advanced SmartObjects in K2 Designer
- How to build simple and more complex Views with different kinds of Controls
- How to build simple and more complex Forms, including Header-Details Forms
- Configuring simple and more complex Rules with Events, Conditions and multiple Actions
- How to integrate Forms with Workflows
- The architecture of SmartForms, including authentication
- How and where to troubleshoot and debug
- How to move applications between environments
- Best practices and tips

Requirements

To get most out of this training course, it is highly recommended that the participants have the following skills and proficiency levels.

Technology/Skill	Proficiency	Notes and examples
Understanding of user interface concepts	Basic	A basic understanding of the elements of user interfaces, such as Controls and Events. What Web Pages are
Understanding of common data providers	Basic	A basic understanding of what SQL databases and web services are

Course Learning Materials

This online site contains the course materials for this training course, including the hands-on exercise step-by-step guides. If required, the course materials are also available as downloadable files.

Item	Format	Notes
Course Handbook	PDF	The student guide (slide handouts) for this course, excluding the hands-on exercises
Course Exercises Guide	PDF	The step-by-step exercises for this course, excluding the slide handouts

Environment for practical exercises

This course includes several exercises where participants will build K2 applications and work with K2 in a live K2 environment. For training events led by a K2 instructor or purchased with K2 credits, a K2 Virtual Machine will be provided in either local (virtual machine) or hosted (cloud) modes. Depending on the course logistics and delivery mechanism, K2 may provision the hosted environments ahead of time or participants will use on-demand provisioning to set up their virtual environments. Please refer to the K2 Knowledge Base article KB001397 for more information about the different Virtual Machine options and the prerequisites for each approach.

For hosted (cloud) virtual machines, participants should verify that they are able to access the virtual machine as described in the Knowledge Base article KB001279 prior to the training event starting, to allow sufficient time to troubleshoot connectivity issues.

For self-directed training where K2 does not provide a cloud-hosted virtual machine, participants may download a Virtual Machine image and use virtualization software to run the machine. This procedure is described further in the Knowledge Base article KB001613. Alternatively, the hands-on exercises in this training course can be repeated in any environment with K2 4.6.9 or later installed, provided that the participant has sufficient rights to deploy K2 applications to their K2 environment. When using your own environment to complete the exercises, some screenshots may be different (for example when searching for users or environment-specific settings shown in screenshots), but fundamentally the exercises can be rebuilt in any K2 environment.

Course Content

This course will take approximately 10 hours to complete (excluding breaks and depending on the additional exercises selected). The course is divided into two modules:

100.YYZ - K2 smartforms Fundamentals

The *100.YYZ - K2 smartforms Fundamentals* training module introduces K2 smartforms and covers the fundamental components, design and use of K2 smartforms

This module covers the following concepts:

- Introduces K2 smartforms and covers the fundamental components, design and use of K2 smartforms
- Building basic SmartForms applications with SmartObjects, Views, Forms
- Integrating with workflows built in K2 Designer

200.YUL - K2 smartforms Intermediate

The *200.YUL - K2 smartforms Intermediate* training module describes more advanced features, functionality and use cases of K2 smartforms, and will give participants a solid overall understanding of K2 smartforms

This module covers the following concepts:

- Building more complex Rules, Views and Forms
- Integrating with workflows built in K2 Studio
- Architecture and Troubleshooting
- Package and Deployment
- Best practices

Additional content and resources

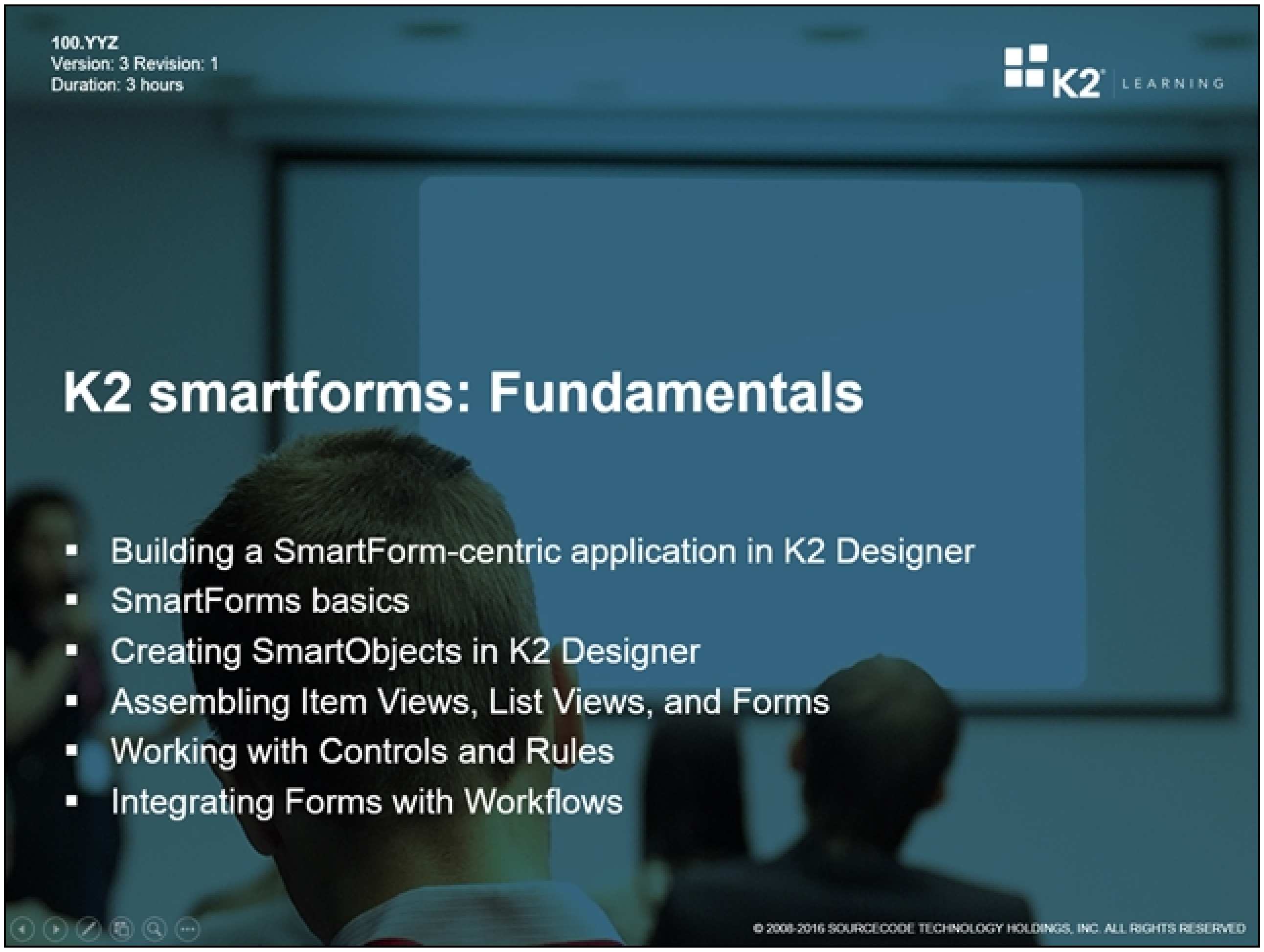
Beyond this course, the following resources and content are also available to users of K2 smartforms.

Content/Resource	Overview and notes
K2 smartforms Product Documentation	The official product documentation for K2 smartforms
K2 Community Site	A community of K2 professionals featuring forums, blogs, samples and other resources
K2 Knowledge Center	K2's Knowledge Base
K2 Product Support	The K2 portal where you may log product support requests

Questions, Comments or Feedback about this training course?

Use of this content is subject to the Terms of Use. Please e-mail learning@k2.com with your comments or feedback. We appreciate any feedback that helps us to improve the quality of our learning material.

100.YYZ: K2 smartforms Fundamentals



100.YYZ
Version: 3 Revision: 1
Duration: 3 hours

K2 LEARNING

K2 smartforms: Fundamentals

- Building a SmartForm-centric application in K2 Designer
- SmartForms basics
- Creating SmartObjects in K2 Designer
- Assembling Item Views, List Views, and Forms
- Working with Controls and Rules
- Integrating Forms with Workflows

© 2008-2016 SOURCECODE TECHNOLOGY HOLDINGS, INC. ALL RIGHTS RESERVED

Video

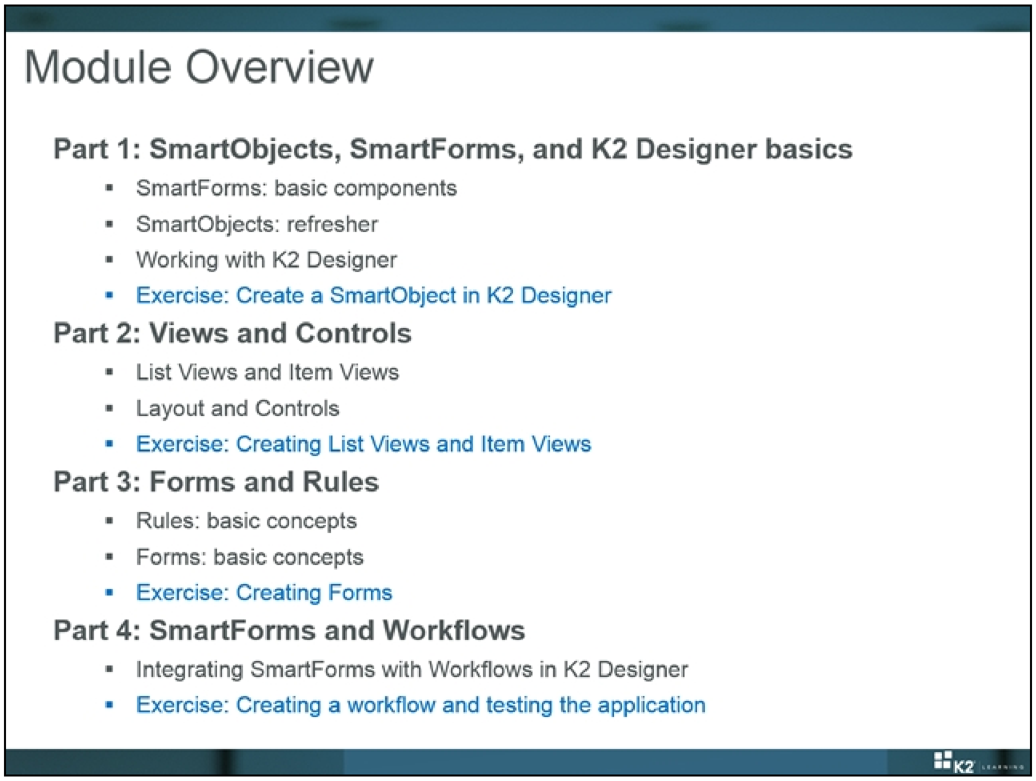
The *100.YYZ - K2 smartforms: Fundamentals* training module explains how to build basic SmartForms and SmartForm-centric applications with Data, Forms and Workflows. In this module, you will learn:

- How to build a basic SmartForm application in K2 Designer
- About the basic components of SmartForms including Views, Forms, Controls and Rules
- How to build a SmartObject in K2 Designer
- Building Item Views, List Views and Forms
- How to work with Controls and Rules
- How to build a workflow in K2 Designer and integrate the workflow with a SmartForm

After completing this module, you will continue on to the *200.YUL- K2 smartforms: Intermediate* training module to learn how to build more complex SmartForms. .

Note
Some of the topics covered in this module should be familiar to you if you recently completed the K2 Core training course. We cover these topics in both courses as a refresher and to ensure that everyone has the same base understanding before we dive deeper.

Module Overview



Module Overview

Part 1: SmartObjects, SmartForms, and K2 Designer basics

- SmartForms: basic components
- SmartObjects: refresher
- Working with K2 Designer
- Exercise: Create a SmartObject in K2 Designer

Part 2: Views and Controls

- List Views and Item Views
- Layout and Controls
- Exercise: Creating List Views and Item Views

Part 3: Forms and Rules

- Rules: basic concepts
- Forms: basic concepts
- Exercise: Creating Forms

Part 4: SmartForms and Workflows

- Integrating SmartForms with Workflows in K2 Designer
- Exercise: Creating a workflow and testing the application

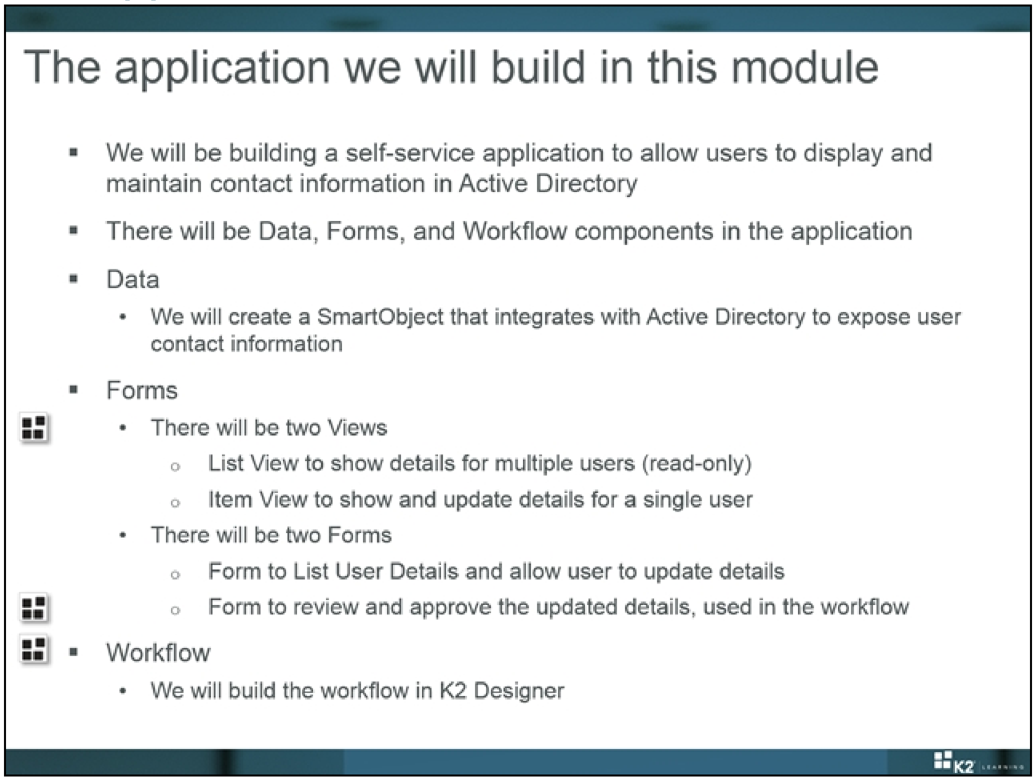
Video

This module is broken into four parts. The parts are structured such that at the end of each part, you will complete a hands-on exercise to apply the concepts discussed in the preceding topics.

- Part 1 covers the basic principles of SmartForms, SmartObjects and K2 Designer
  - SmartForms basics and how SmartForms are used
  - Using the Category (System) Browser
  - Creating SmartObjects in K2 Designer
  - Part 1 ends with a lab exercise where participants will build a SmartObject in K2 Designer.
- Part 2 covers Views and Controls
  - List Views and Item Views
  - View layouts and using tables to control View layouts
  - Sorting and filtering on List Views
  - Working with Controls (including basic validation and expressions)
  - Part 2 ends with a lab exercise where participants will build two Views (an Item View and a List View) that use the SmartObject created in Part 1.
- Part 3 covers Forms and Rules
  - Rule basics and working with Rule Actions to perform User Interface interaction
  - Events, Conditions and Actions
  - Part 3 ends with a lab exercise where participants will build a Form to host the two Views that were created in Part 2.
- Part 4 covers building Workflows in K2 Designer
  - How SmartForms integrate with workflows built in K2 Designer
  - Start Forms vs. Task Forms
  - Part 4 ends with a lab exercise where participants will build a Workflow that uses the Form created in Part 3.

At the end of this module, you will have built a simple SmartForms-based application in K2 Designer by creating SmartForms, Views, Forms and a Workflow.

The application we will build in this module



The application we will build in this module

- We will be building a self-service application to allow users to display and maintain contact information in Active Directory
- There will be Data, Forms, and Workflow components in the application
- Data
  - We will create a SmartObject that integrates with Active Directory to expose user contact information
- Forms
  - There will be two Views
    - List View to show details for multiple users (read-only)
    - Item View to show and update details for a single user
  - There will be two Forms
    - Form to List User Details and allow user to update details
    - Form to review and approve the updated details, used in the workflow
- Workflow
  - We will build the workflow in K2 Designer

Video

To gain some hands-on experience using K2 Designer, you will be building a complete application in this learning module. The application we will build is a self-service application where we will allow users to display and maintain contact information that is stored in Active Directory (AD). Users can maintain their own contact information in AD, or update someone else's information. If the user is updating someone else's information, that person must review and accept the changes before AD will be updated.

Tip
You will be able to rebuild this same application in your own organization’s K2 environment if you wish; you may follow these lab exercises to do so.

The Application will have Data, Forms and Workflow components

Data components

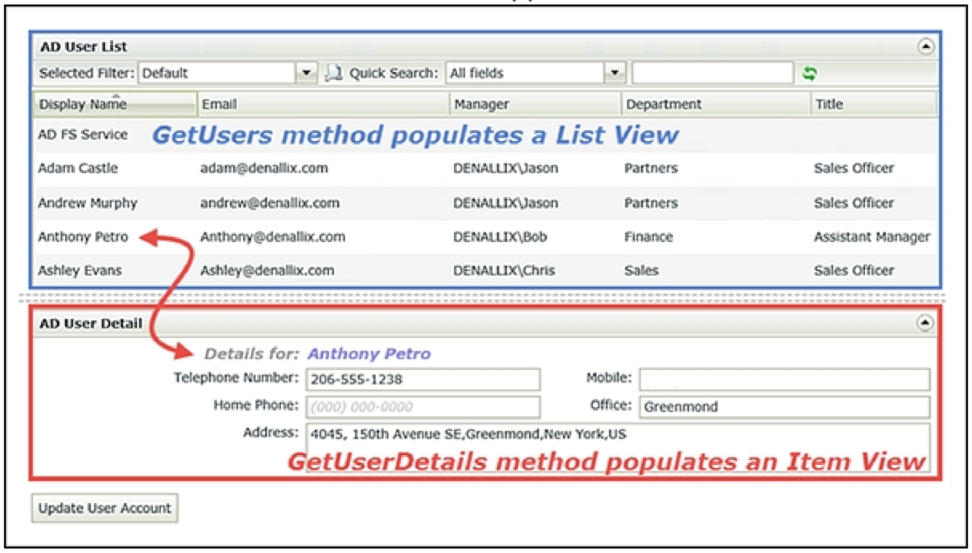
The Data for this application consists of a single SmartObject that integrates with Active Directory to read single user details (*GetUserDetails*) or multiple user's information (*GetUsers*). Other temporary data storage for the application such as storing updated telephone numbers will be handled within the K2 Workflow.

Form components

This application will contain two Views and two Forms.

For the Views, there will be a List View to show a list of multiple users in read-only mode. An Item View is used to show the details for a specific user. This View will also allow the viewer to update the contact information.

The Views used in the AD User Details application


AD User List
Selected Filter:
Quick Search:
GetUsers method populates a List View
AD User Detail
GetUserDetails method populates an Item View
Update User Account

Display Name	Email	Manager	Department	Title
AD FS Service				
Adam Castle	adam@denallix.com	DENALLIX\Jason	Partners	Sales Officer
Andrew Murphy	andrew@denallix.com	DENALLIX\Jason	Partners	Sales Officer
Anthony Petro	Anthony@denallix.com	DENALLIX\Bob	Finance	Assistant Manager
Ashley Evans	Ashley@denallix.com	DENALLIX\Chris	Sales	Sales Officer

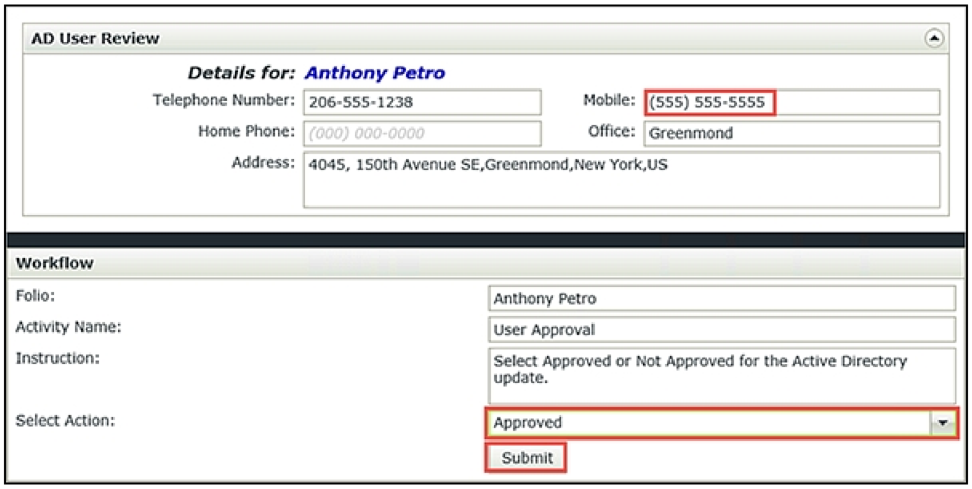
Selected Filter: Default
Quick Search: All fields

Details for: Anthony Petro
Telephone Number: 206-555-1238
Mobile:
Home Phone: (000) 000-0000
Office: Greenmond
Address: 4045, 150th Avenue SE,Greenmond,New York,US

For the Forms components, one Form will be used to host the two Views created: one View to list users and another View to show the details of a selected user. The Form will also contain a button that will initiate the workflow if user details are updated. The screenshot above shows the layout of this Form.

The second Form will be used for the Review Step of the workflow, and will only host the Item View. This Form will feature a panel where the user may decide whether they accept or reject the updated contact information.

The Review Form



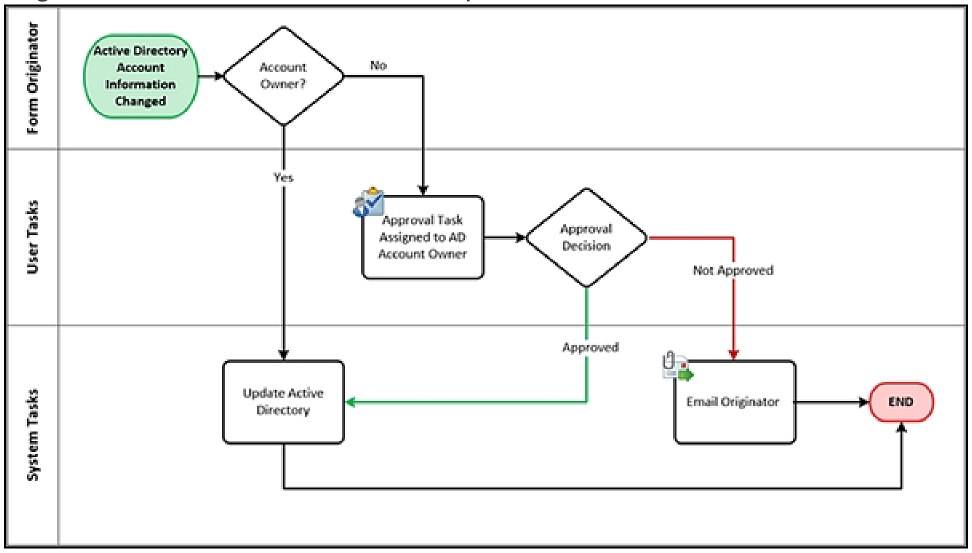
Details for: Anthony Petro
Telephone Number: 206-555-1238
Mobile: (555) 555-5555
Home Phone: (000) 000-0000
Office: Greenmond
Address: 4045, 150th Avenue SE,Greenmond,New York,US

Folio: Anthony Petro
Activity Name: User Approval
Instruction: Select Approved or Not Approved for the Active Directory update.
Select Action: Approved

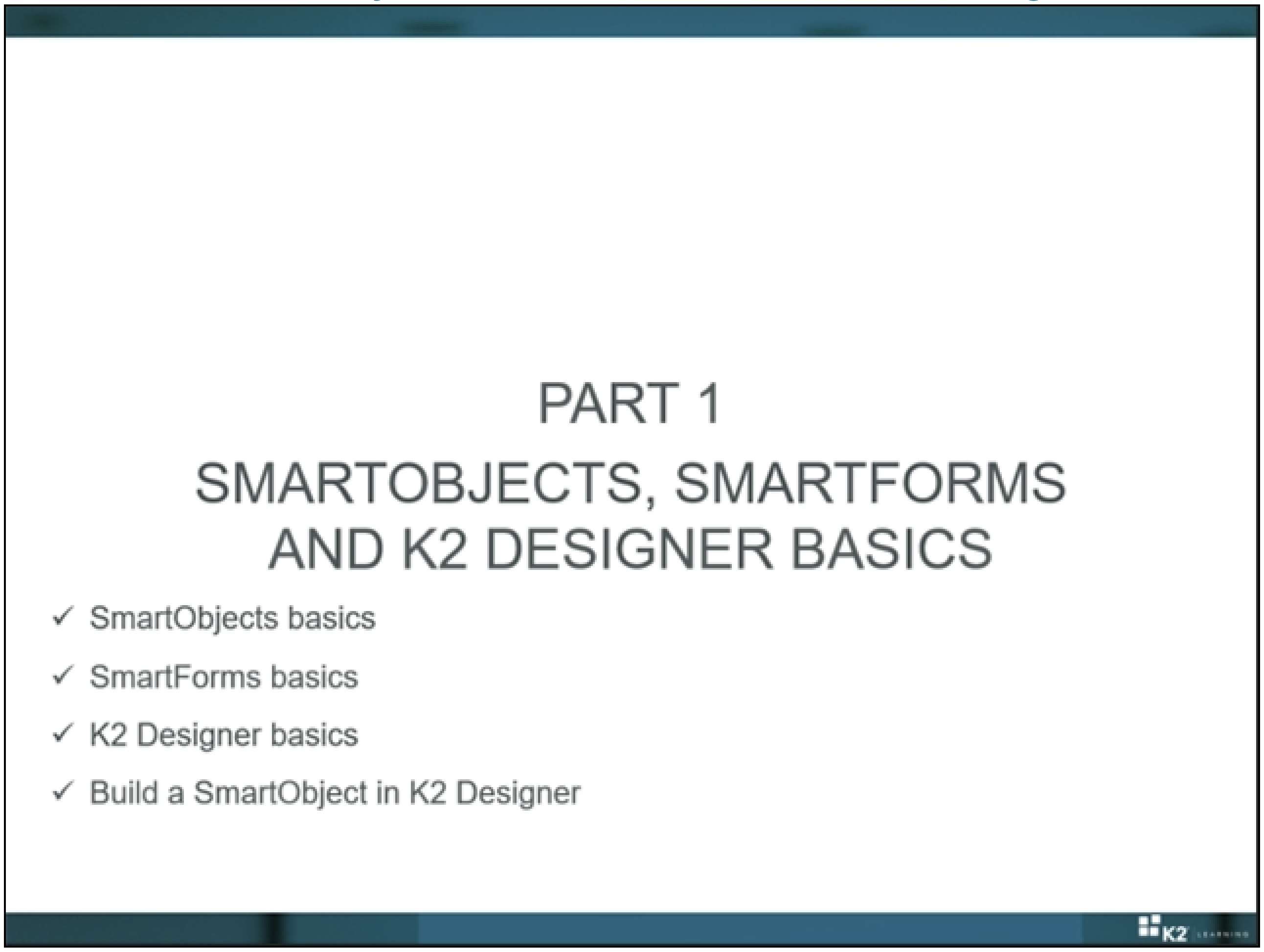
Workflow component

The workflow will be responsible for updating Active Directory with the updated user details. If the workflow determines that a user updated someone else's details, the workflow will first route to the other user (the account 'owner') for their acceptance, before it will update Active Directory.

Logical view of the AD User Details update workflow



PART 1: SmartObjects, SmartForms, and K2 Designer Basics



PART 1

SMARTOBJECTS, SMARTFORMS AND K2 DESIGNER BASICS

- ✓ SmartObjects basics
- ✓ SmartForms basics
- ✓ K2 Designer basics
- ✓ Build a SmartObject in K2 Designer

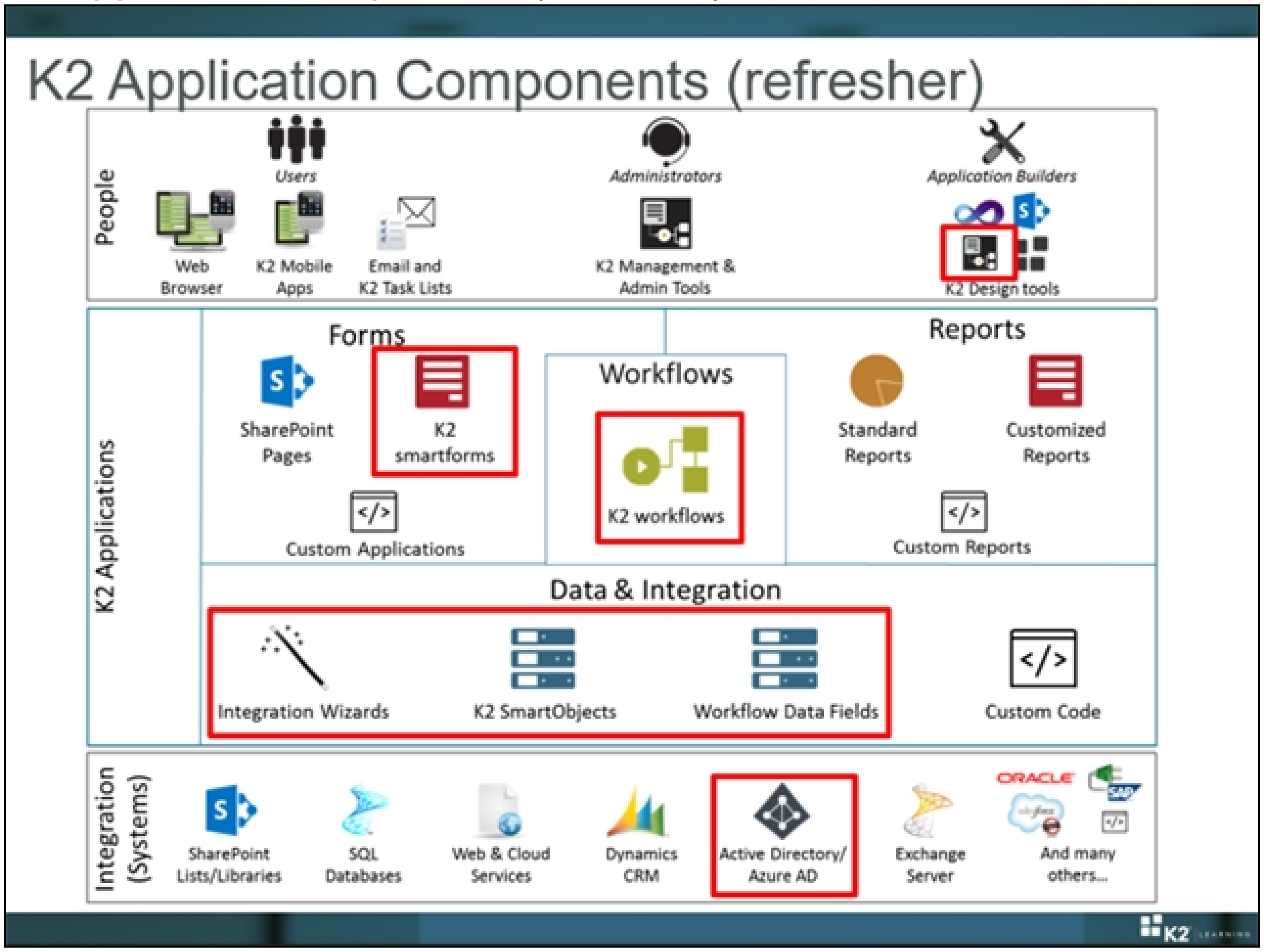
Video

In Part 1 we will look at the basics of SmartObjects and SmartForms, as well as working with the browser-based K2 Designer tool.

There are a number of refresher topics in this section. If you have recently completed the K2 Core training course, some of these topics may still be familiar to you. Since these topics are important when working with SmartForms, we are refreshing your knowledge (particularly if it has been some time since you completed the Core training course), just to ensure that everyone who covers this training is on the same level of understanding. Feel free to skip lightly over the refresher topics if the concepts are still fresh in your mind.

At the end of Part 1, we will build a SmartObject in K2 Designer that will integrate with Active Directory to display user contact information. We will use this SmartObject in the Views of our AD User Details application.

K2 Application Components (refresher)



Video

The diagram above illustrates the main components of the K2 platform. We have separated these components into **People** (users who work with K2 or use K2 applications), **K2 Applications** elements (the things that make up K2 applications) and **Integration** (the systems that K2 integrates with). We have highlighted the specific components that we are covering in this module with red boxes.

K2 applications consist of four main elements: **Data**, **Forms**, **Workflows** and **Reports**. The application elements fully integrate with each other to provide a robust framework for building applications. However, they also work independently from each other: for instance, you might have a SharePoint List that utilizes SmartObjects and SmartForms, but not Reports or Workflows. K2 also provides tools for designing and customizing the application elements as well as tools and interfaces to administer the K2 environment, such as managing permissions and managing the active workflows in the environment.

People, Roles and Administering K2 Applications

K2 defines a number of logical roles that make up the human element of your application. Users are the consumers of your application's functionality and they use the Forms that you build to interact with your application. Usually, users would use your applications with web browsers or mobile applications. Administrators manage workflow instances and permissions, and use K2's Management Pages or other administration tools to perform these functions. Application Builders are the designers and builders of the Forms, Data, Workflows and Reports using the web-based tools such as the K2 Designer.

Forms

Forms are the User Interfaces that users will use to interact with your application. K2 supports a range of technologies for these forms: K2 smartforms, InfoPath forms, SharePoint interfaces or custom applications written in .NET or other programming languages. The exact technology used for an application depends on the requirements and the organization's infrastructure, but the main point is that K2 does not bind you to a particular technology for creating forms.

Workflows

Workflows are a logical sequence of steps and events performed by users and/or systems. Workflows provide the "business process" functionality in your application. K2 features a mature and powerful workflow engine which has proven itself over many years, in many different organizations.

Consider something like a Leave Request application: in this case, there would be a workflow component which runs the Leave Request-Approval-Processing steps of the application. The workflow assigns tasks to users (the employee's manager, for example, when the Leave Request must be approved) and performs system tasks as well (for example, sending e-mail notifications to the employee to notify them that their Leave Request was approved).

Reports

K2 provides standard reports which expose the metrics of your application workflows, such as the time taken to complete Leave Requests and audit trails. You can also use SmartForms and SmartObjects to create customized dashboard-style reports, leveraging the available Reporting Controls in the K2 Designer toolbox. Or, you can build custom reports using third-party reporting tools that are able to consume the available K2 APIs and services (such as Excel, for example).

Data and Integration

The Data component allows you to integrate your application with providers of data and consumers of data. K2 primarily uses a technology called SmartObjects to interact with back-end systems. You can think of SmartObjects as a "connector" or a "middle layer" that provides the necessary integration to expose the data (that resides in some system) to the K2 application elements that need to use that data. The important point is this: SmartObjects are the consistent interface that K2 uses to interact with some provider of Data, regardless of what technology that provider might be. The artifacts in that provider (for example the "Employee" or "Customer" or "Account" entities) are represented as logical business entities that you can easily use within your applications, without having to know where that data resides.

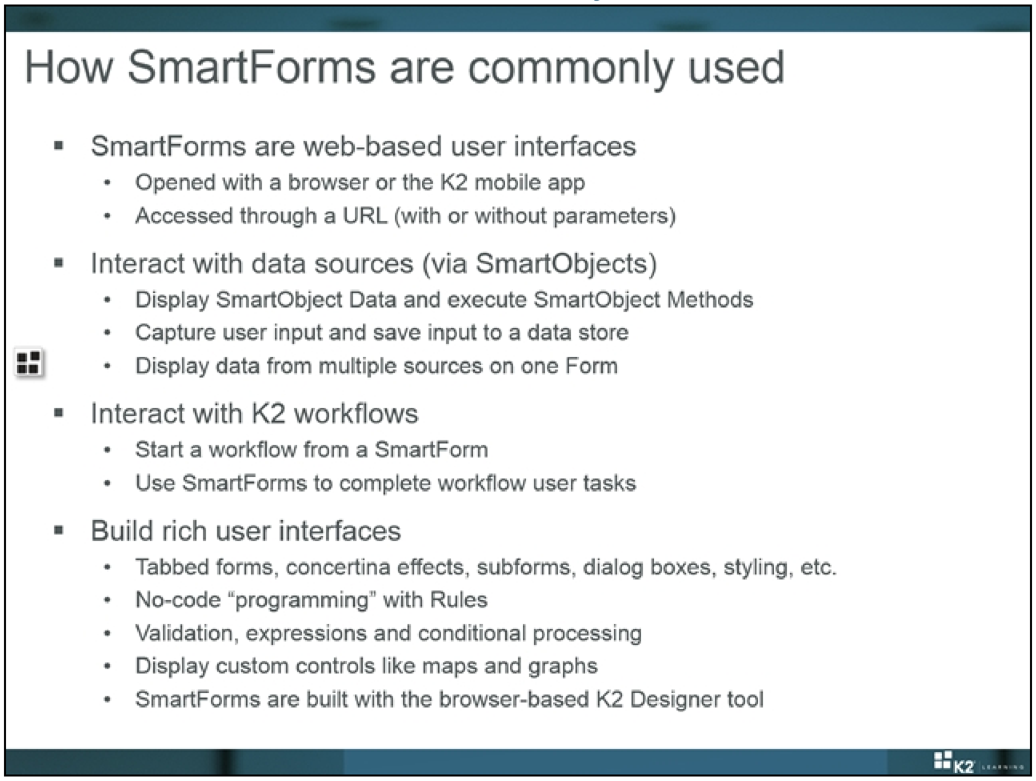
Data could also refer to workflow-level data fields that are defined in workflows, but this is normally limited to storing data that is only applicable to the workflow itself.

K2 provides a set of standard workflow wizards that make it easy to integrate with common enterprise systems like Active Directory, Exchange, SharePoint and many others.

Summary

- K2 applications consists of Forms, Reports, Workflows and Data (Data includes integration with Line-Of-Business systems).
- The diagram in the slide lists the most common options for each components. For example, SmartForms for the Forms part, SmartObjects for the Data part, standard reports for the Report part, etc.
- In this particular module, we will look at using the following:
  - The browser-based K2 Designer environment to build SmartObjects, Forms and Workflows.
  - Forms: using K2 SmartForms for the User Interface component.
  - Workflow: building a K2 Workflow with basic workflow functionality using the browser-based workflow design tool.
  - Data: We will use a SmartObject to read data from Active Directory, and workflow-level data fields for temporary data storage.
  - For integration with Active Directory, we will use workflow wizards that interact with Active Directory.

How SmartForms are commonly used



How SmartForms are commonly used

- SmartForms are web-based user interfaces
  - Opened with a browser or the K2 mobile app
  - Accessed through a URL (with or without parameters)
- Interact with data sources (via SmartObjects)
  - Display SmartObject Data and execute SmartObject Methods
  - Capture user input and save input to a data store
  - Display data from multiple sources on one Form
- Interact with K2 workflows
  - Start a workflow from a SmartForm
  - Use SmartForms to complete workflow user tasks
- Build rich user interfaces
  - Tabbed forms, concertina effects, subforms, dialog boxes, styling, etc.
  - No-code “programming” with Rules
  - Validation, expressions and conditional processing
  - Display custom controls like maps and graphs
  - SmartForms are built with the browser-based K2 Designer tool

Video

Let’s describe some of the common ways SmartForms are used. These are by no means all of the use cases. You may have already identified other use cases in your organization where SmartForms may be a good fit.

Web-based User Interfaces

First and foremost: K2 smartforms are a web-based User Interface. This means that users can view and interact with SmartForms through most modern web browsers. They may use devices with browser support such as computers, tablets and smartphones to work with SmartForms. SmartForms are exposed though URLS (which may or may not include query string parameters for additional processing). SmartForms can even be exposed in a SharePoint site using the K2 smartforms Form Viewer WebPart.

Forms can also be displayed in the K2 mobile application such as the K2 iOS and K2 Android mobile application.

Interact with data sources through SmartObjects

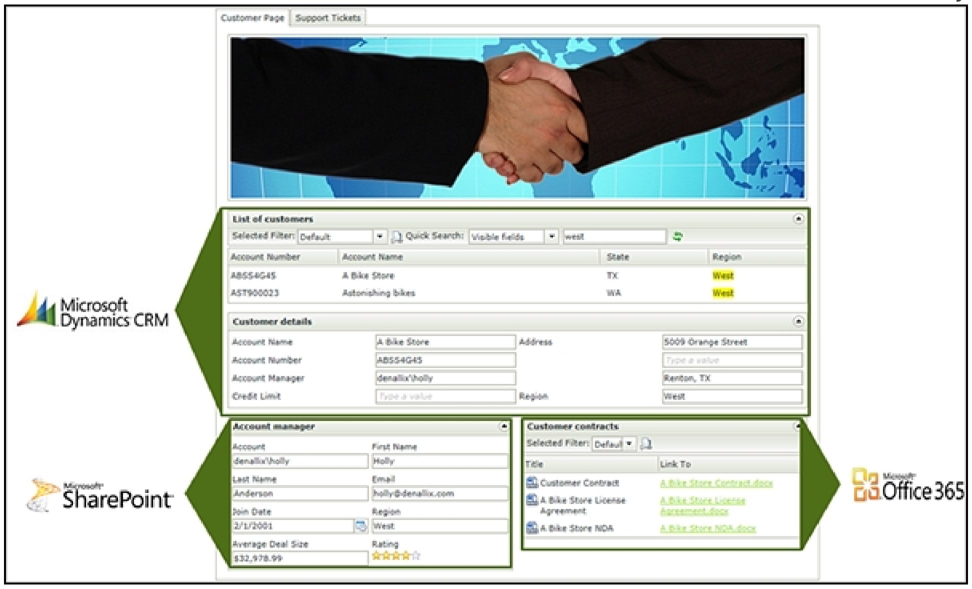
SmartObjects are used to allow SmartForms to interact with data sources. Any system exposed as a SmartObject can become a data provider for a SmartForm. SmartForms can be as basic as just displaying read-only data on a Form, or can be "wired up" to execute SmartObject methods to interact with a data source, including scalar (single-record operations) or List methods (multiple records). It is a very common use case to use SmartForms to capture user input and then save that input to a data store.

Note
The only standard data source for SmartForms is SmartObjects. If you have some bespoke system that provides data that you would like to integrate with in SmartForms, you would either need to create a custom service broker to expose the system data in SmartObjects or create a custom control in SmartForms to query the data source directly and display data.

Once you grasp the power of SmartObjects, Views and Forms, you will start to appreciate the possibilities open to you. Consider the sample Form below. Here, we are using standard K2 service brokers to expose data from Microsoft Dynamics CRM, SharePoint and Office 365 as SmartObjects. These SmartObjects have been exposed on re-usable Views, and these have in turn been added to a Form that represents customer information. Ultimately, the user gets a global view of a customer without having to go to different systems. The individual Views can be re-used in other

Forms, so if another developer has a requirement to show account manager information on a Form for example, they can just re-use the existing View.

A Form with three Views, where the data in the Views comes from different systems



List of customers

Selected Filter: Default | Quick Search: Visible fields | west

Account Number	Account Name	State	Region
ABSS4G45	A Bike Store	TX	West
AST900023	Astonishing bikes	WA	West

Customer details

Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

Account manager

Account	First Name
denallix\holly	Holly
Last Name	Email
Anderson	holly@denallix.com
Join Date	Region
2/1/2001	West
Average Deal Size	Rating
$32,978.99	★★★★☆

Customer contracts

Selected Filter: Default

Title	Link To
Customer Contract	A Bike Store Contract.docx
A Bike Store License Agreement	A Bike Store License Agreement.docx
A Bike Store NDA	A Bike Store NDA.docx

Interact with K2 Workflows

Since SmartForms are so tightly integrated with K2, it is only logical that they should be able to interact with K2 Work-flows. You could use SmartForms to start K2 Workflows as well as use them for the User Interfaces of Client Event-s/User Tasks when users need to provide input to the workflow. You can integrate Forms with workflows created in any workflow design tool provided by K2: K2 Designer, K2 Studio or K2 for Visual Studio.

K2 smartforms are provided with standard rules that make interacting with workflows easy, and the workflow design tools have new Event Wizards that can add the required Rules to workflow-enable a Form. Of course, you are not lim-ited to only these approaches: you may define your own Rules that determine if, how and when SmartForms should interact with workflows.

Here is an example: suppose you create an Order Entry Form. Perhaps there is a business rule that says that Order Entries do not need to be approved unless a certain set of conditions are true. Using Rules and Conditions, you could use the Advanced Conditions editor in a Rule to design a Rule that will always save the new Order Entry data to a data-base, and only start the approval workflow if the required conditions are met. You will learn how to do this, and more, in the available SmartForms Learning Modules.

Build powerful web-based User Interfaces

Designers can create rich and advanced user input Forms that can utilize Tabbed Forms, conditionally hidden controls, expandable and collapsible Views and Sub-Forms, dialog boxes and custom pop-ups, browser navigation, styling, and more, to build very dynamic and complex User Interfaces. All of this is possible without any code. SmartForm Rules allow you to build the processing logic or "program" the Forms without having to write any code.

SmartForms also allow you to define validation Rules so that user input is accurate, and you can even use expressions to perform calculations or manipulation on data before saving data to the underlying system.

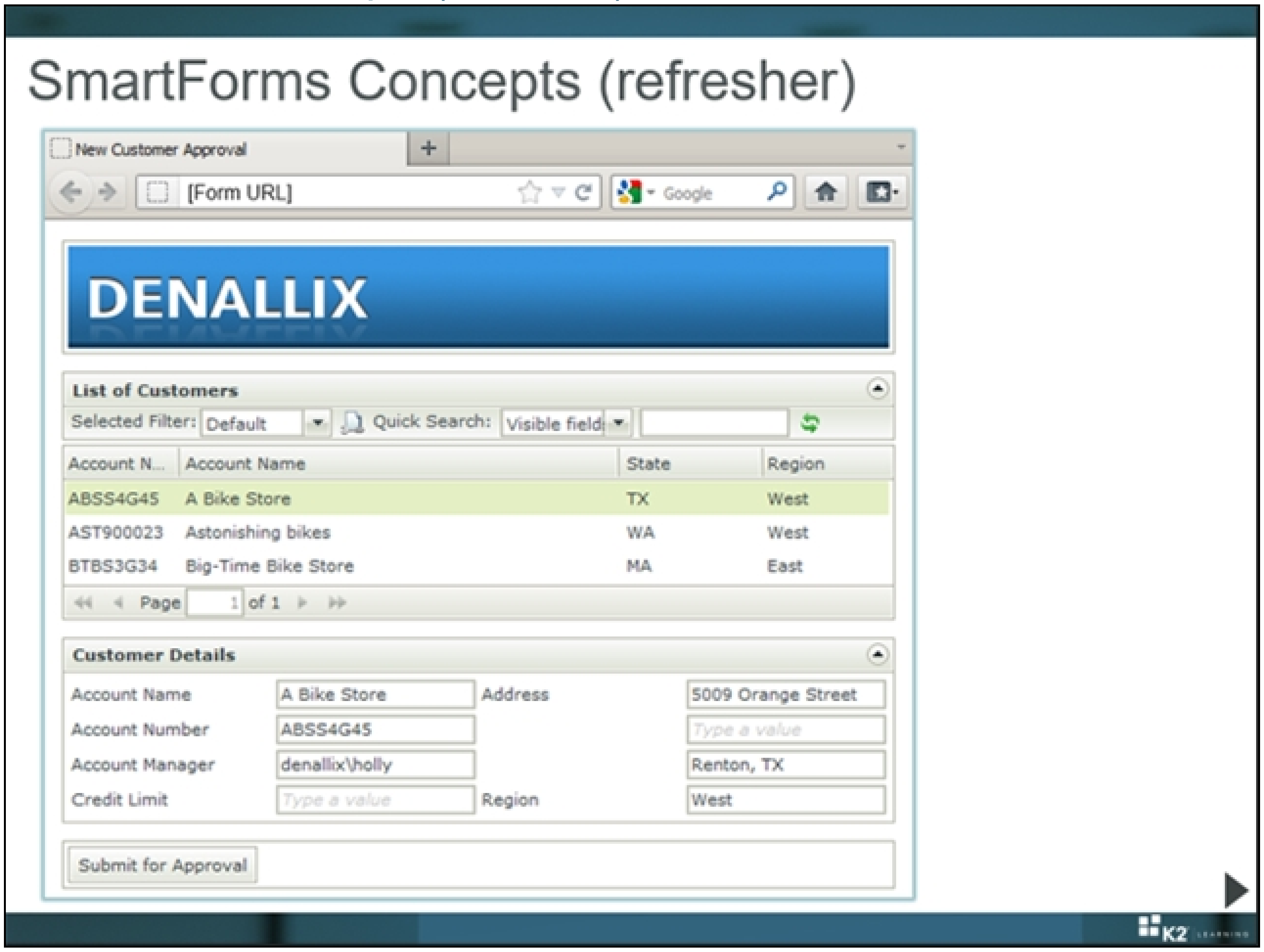
Developers can even extend the set of default controls available for SmartForm designers and can create custom con-trols to display other data like maps, dynamic content, graphs and so on.

Summary

- SmartForms are basically web-based User Interfaces and can be opened with web browsers like Internet Explorer, or the K2 mobile app.
- SmartForms use SmartObjects to interact with data providers (e.g. SQL databases, web services, etc.).
  - They can display SmartObject data and execute SmartObject Methods.
  - You can capture user input and save input to a data store.
  - You can display data from multiple sources on one Form.

- You can use SmartForms to interact with K2 Workflows, including starting a workflow and completing user tasks in a workflow.
- You can build rich User Interfaces with features like:
  - Tabbed Forms, concertina effects, Sub-Forms, dialog boxes, styling, etc.
  - No-code “programming” with Rules.
  - Validation, expressions and conditional processing.
  - Display custom controls like maps and graphs (creating custom controls is a developer task).
  - SmartForms are built with the browser-based K2 Designer tool.

SmartForms Concepts (refresher)


SmartForms Concepts (refresher)

New Customer Approval

[Form URL]

DENALLIX

List of Customers

Selected Filter: Default   Quick Search: Visible field

Account N...	Account Name	State	Region
ABSS4G45	A Bike Store	TX	West
AST900023	Astonishing bikes	WA	West
BTBS3G34	Big-Time Bike Store	MA	East

Page 1 of 1

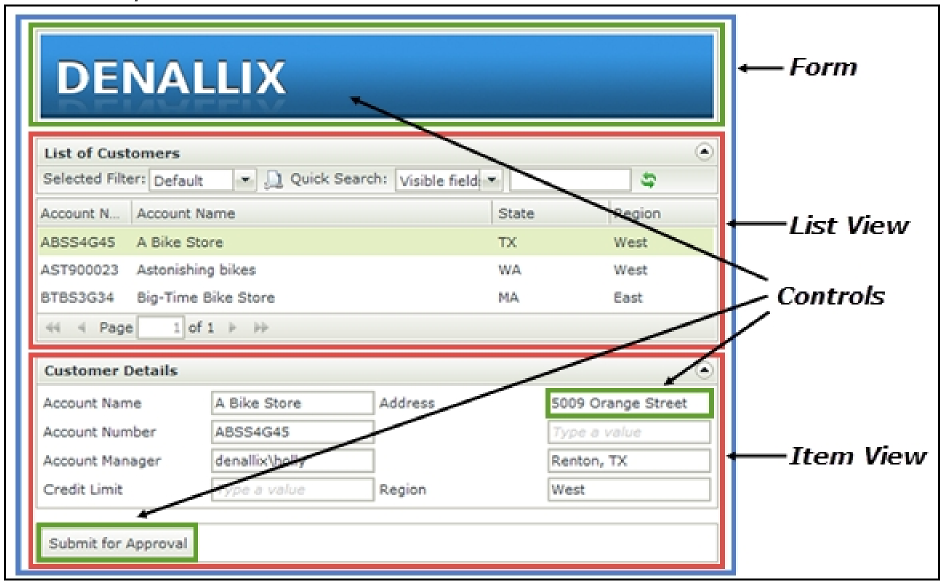
Customer Details

Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

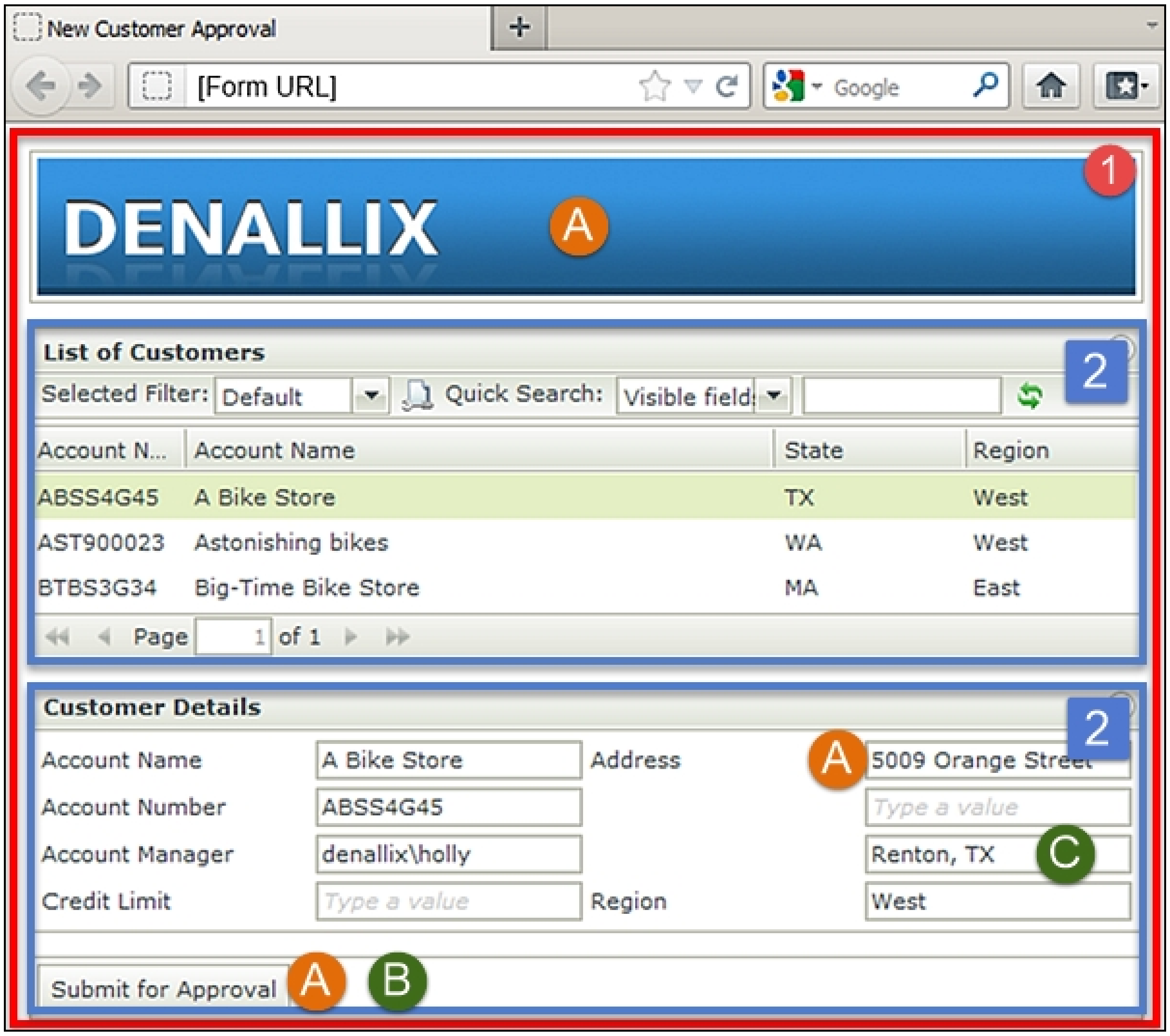
Submit for Approval

A SmartForm is made up of several building blocks and it is important to understand how these building blocks are used together to build a SmartForm. In this topic, we will briefly cover the basic components of a typical Form: Form, Views (List and Item Views), Controls, Rules and Expressions. Other topics in this module will dive deeper into these components, but for now we just want you to have an overall understanding how these components work together.

Basic components of SmartForms



In the image below, we break down the components of a SmartForm for a sample Form. Review the component details that are shown just below the image so that you are familiar with the concepts and terminology of a SmartForm.



List of Customers

Selected Filter: Default   Quick Search: Visible fields

Account N...	Account Name	State	Region
ABSS4G45	A Bike Store	TX	West
AST900023	Astonishing bikes	WA	West
BTBS3G34	Big-Time Bike Store	MA	East

Page 1 of 1

Customer Details

Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

1. (Red) **Form**

All of the Form contents are contained within the Form.

2. (Blue) **View**
Forms contains Views. In this sample, there are two Views on the Form: a List View listing several customer records, and an Item View that shows the details for a specific customer. (List Views contains rows of records, while the Item Views contain the content from one record only.)

A. (Orange) **Controls**
There is an image control at the top of the form, an Address text box control and a Submit Button control. All of these are different types of Controls that you can add to Views and Forms.

B. (Green) **Rules**
The Submit Button contains Rules that start the associated Workflow. Rules might also validate that certain Form fields are not empty.

C. (Green) **Expressions**
Expressions can be used to format fields. In this sample, Renton,TX is an expression that concatenates the City field and the State field.

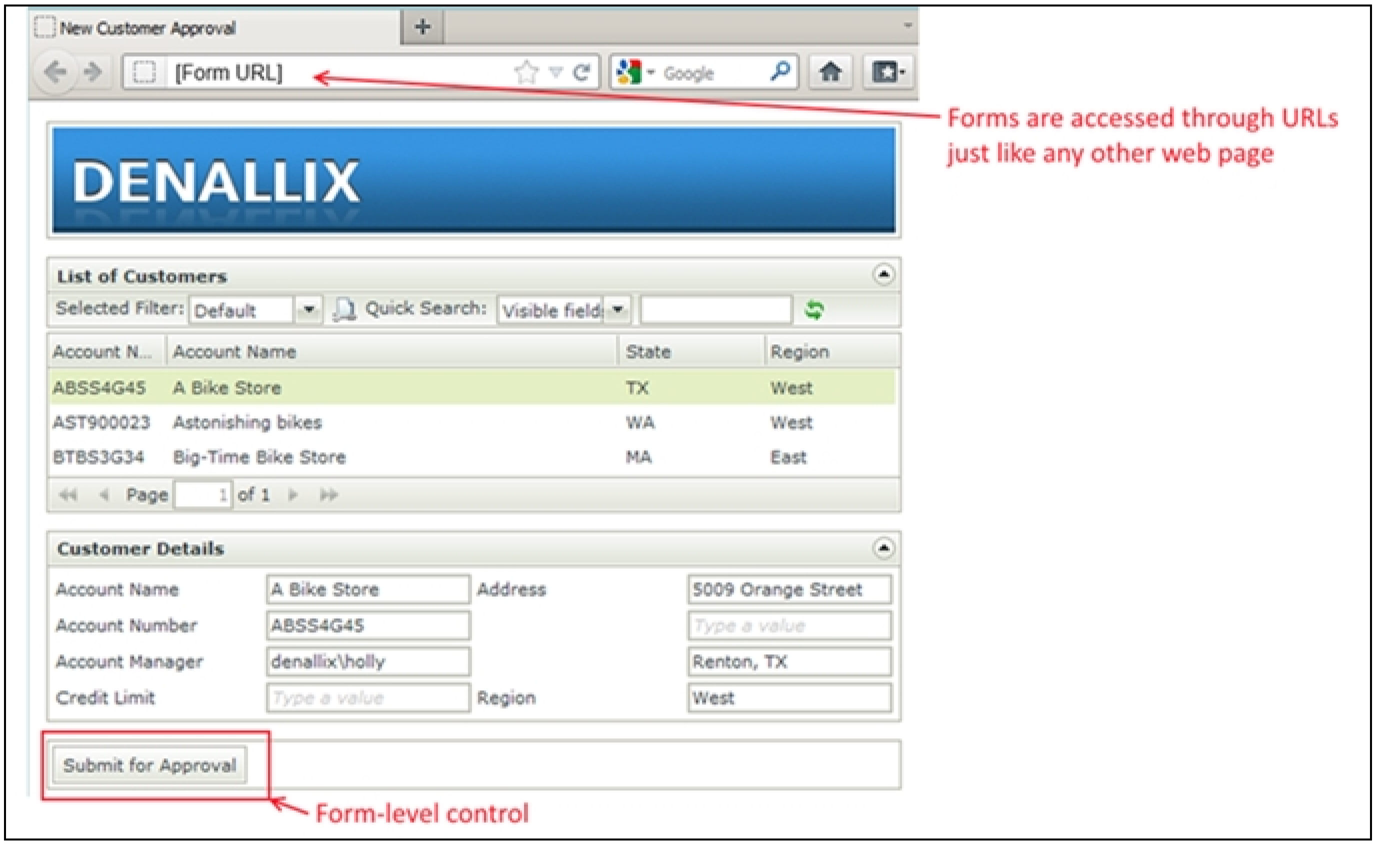
Form

A Form is the web page that is presented to the end user. A Form is a container for one or more Views, and may contain Form-level controls and Form-level Rules as well. Note that Forms do not display data from SmartObjects directly: SmartObject data is always displayed in Views. Forms can be “Tabbed” so that multiple Views can be presented in the same Form.

Forms can include Form-level Controls and Rules. These Controls or Rules can act on Controls and Rules in the Views that are displayed on a Form. Because a Form is aware of the Views that it contains, Form-level Rules can enable interaction between Views on the Form. For example, a Form-level Rule can read values in one View and then pass those values to another View on the same Form. A Form can also call Rules on its child Views.

End users cannot use Views directly; they must access Views through Forms. This effectively means that, until a View is added to a Form, end users will not be able to use the View.

Sample of a Form



List of Customers

Selected Filter: Default Quick Search: Visible fields

Account N...	Account Name	State	Region
ABSS4G45	A Bike Store	TX	West
AST900023	Astonishing bikes	WA	West
BTBS3G34	Big-Time Bike Store	MA	East

Page 1 of 1

Customer Details

Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

View

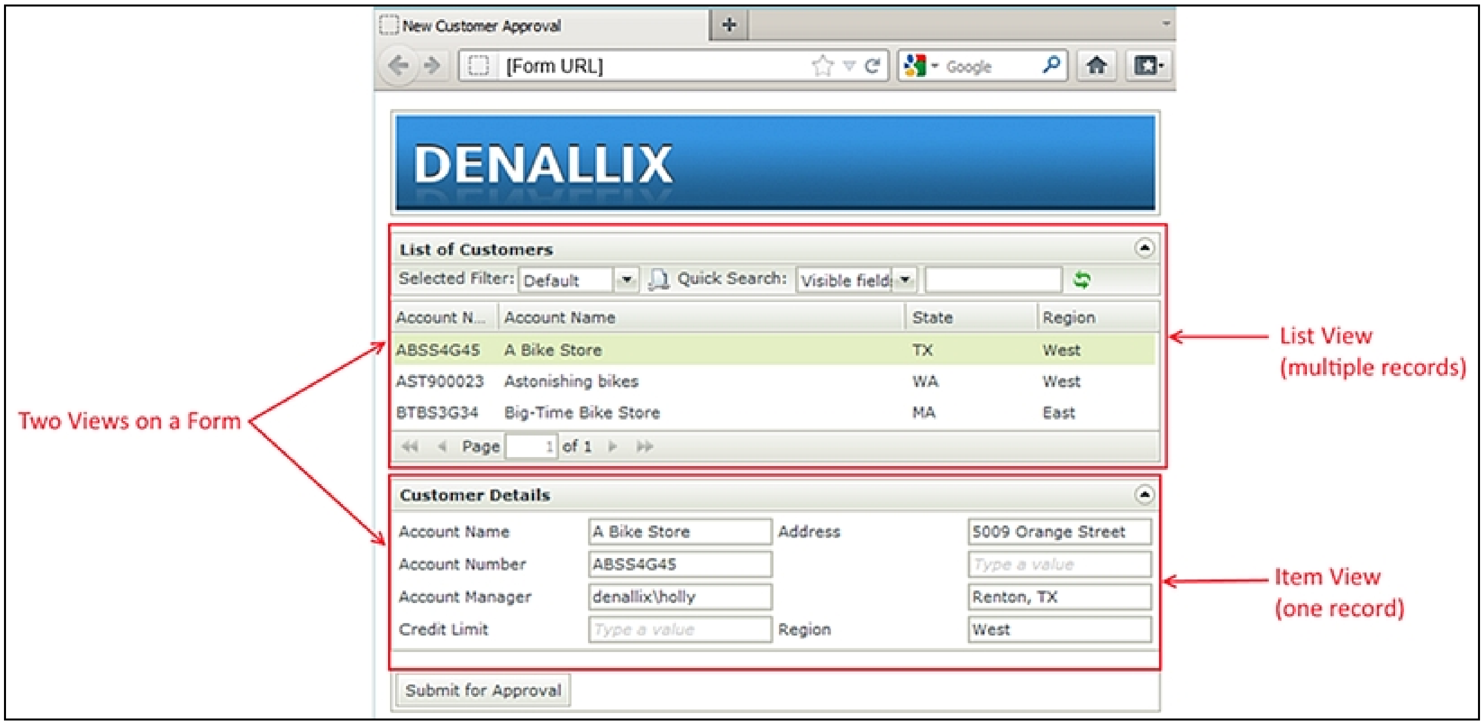
A View visually represents data from a SmartObject and contains Controls such as buttons, images, labels or lists of information. These Controls may or may not be bound to SmartObject properties. Views are also used to input data, which is then saved into a SmartObject using Rules.

When designing a View, you can select various kinds of Controls to represent the properties of the SmartObject, for example a date/time picker Control for a Date property, or a file attachment Control for a File property. Views are normally laid out with a table-like structure, and it is possible to modify the table layout by adding columns and rows and merging cells. You can also apply styling to Controls so that they are displayed in a specific format in the View. Views can be expanded and collapsed to control the display of multiple Views on a single Form.

An important point is that a View can be re-used on multiple Forms. This helps to speed up User Interface development significantly since generic Views can be re-used in many other solutions.

Views can be classified into two main types: List Views (which display data from multiple items) and Item Views (which display data from only one item). A classic example of a List View is a grid-like display where data from multiple records of the same SmartObject is presented in a tabular format. Item Views normally display the details of a specific record, or to allow users to capture/update data for a specific record.

Views on a Form



List of Customers

Selected Filter: Default Quick Search: Visible fields

Account N...	Account Name	State	Region
ABSS4G45	A Bike Store	TX	West
AST900023	Astonishing bikes	WA	West
BTBS3G34	Big-Time Bike Store	MA	East

Page 1 of 1

Customer Details

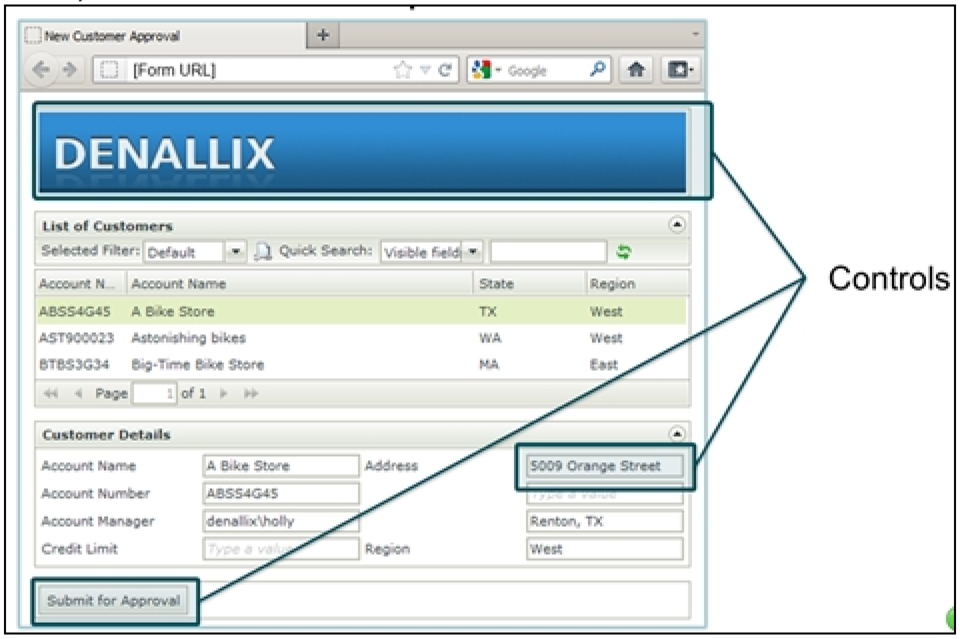
Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

Control

Controls are items like text boxes, labels, drop-down menus, images, buttons and more. Controls can be used on a View or a Form. Controls are typically used to display the individual properties of a SmartObject, display the result of an expression/calculation, show images or to expose buttons that an end-user can click to execute a Rule.

Controls can be styled (statically or conditionally) using the Style Builder. Controls can have validation to ensure that user input is accurate. Controls can be populated by expressions and have control-level Rules that perform some action when an event happens on the Control. (For example, as soon as the user changes the content of the Credit Limit text box, show a Sub-form to warn the user that changing the credit limit will require approval.)

Examples of Controls



List of Customers

Selected Filter: Default Quick Search: Visible fields

Account N...	Account Name	State	Region
ABSS4G45	A Bike Store	TX	West
AST900023	Astonishing bikes	WA	West
BTBS3G34	Big-Time Bike Store	MA	East

Page 1 of 1

Customer Details

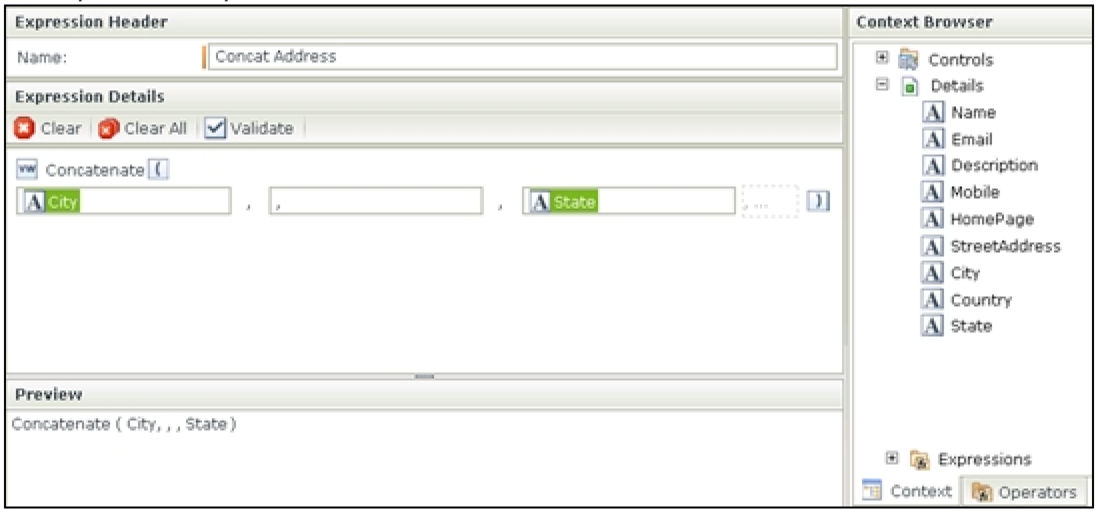
Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

Expression

Expressions are essentially operations or functions that can manipulate data or perform calculations. Normally, expressions are used to auto-populate controls on an Item View (such as using text manipulation functions to perform concatenations, search-and-replace-functions or perform mathematical calculations. Expressions are also commonly used to aggregate data across Lists (such as calculating a sub-total or average for a list of items).

There is a range of standard operators that can be used to build up expressions. The Expression Editor is used to combine these operators with properties to perform processing of some kind. Designers can also use the Expression Editor to define more advanced processing such as nesting expressions and re-using the result of other expressions that exist in the same View.

Example of an Expression



Concatenate (City, , , State)

Rule

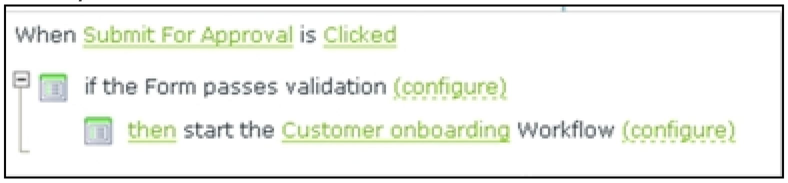
Rules are the business logic behind Forms, Controls and Views, and normally execute an action such as calling a SmartObject method, launching a sub-form or navigating the user to another page. (There are many more uses for Rules, which we will look at in more detail later in this module, and in other modules.)

Rules can be defined for Controls (e.g. execute a specific SmartObject method when a button is clicked); Views (e.g. populate the View with information as soon as it is opened) or Forms (e.g. when the Form is opened, use a parameter passed to the Form, set a value in a View and then execute a SmartObject method).

Rules are made up of three basic components

1. An optional Event (defines WHEN the Rule should fire)
2. An optional Condition (defines Conditions that determines IF the Rule should execute)
3. An Action (WHAT the Rule should do)

Example of a Rule associated with the click of a Button control



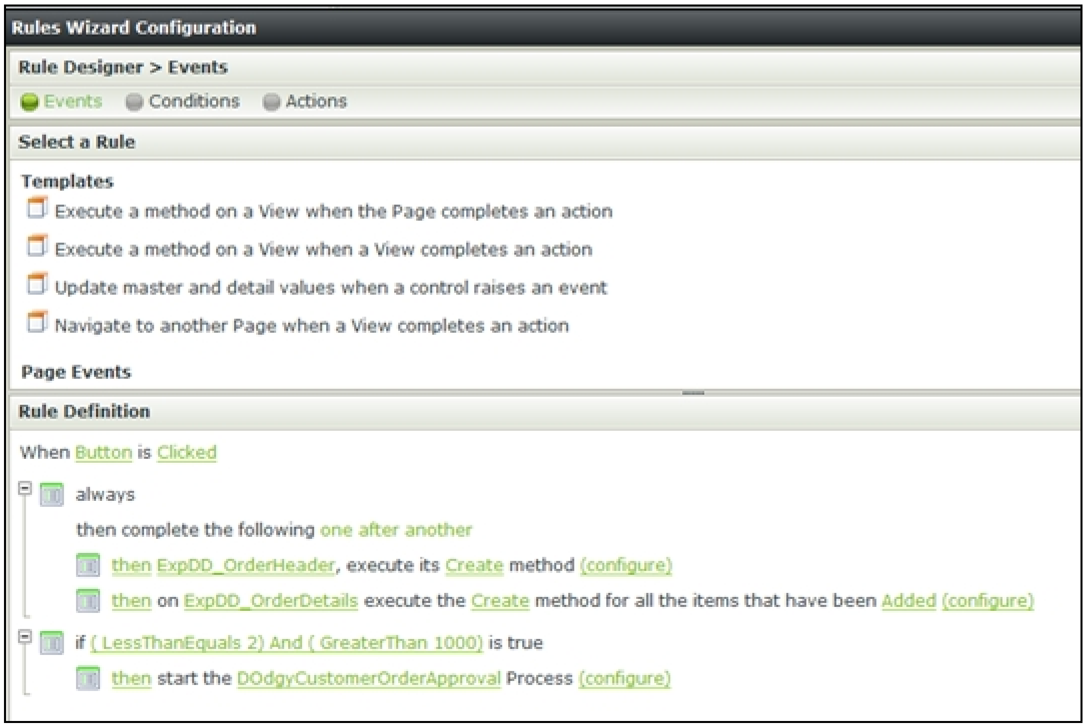
When Submit For Approval is Clicked

if the Form passes validation (configure)

then start the Customer onboarding Workflow (configure)

Of course, designers can create more complex and advanced Rules that include batch actions, sequential actions, multiple conditions, or re-use and override View Rules in Forms.

An example of a more advanced Rule with Conditions and multiple Actions


Rules Wizard Configuration

Rule Designer > Events

Events Conditions Actions

Select a Rule

Templates

Execute a method on a View when the Page completes an action

Execute a method on a View when a View completes an action

Update master and detail values when a control raises an event

Navigate to another Page when a View completes an action

Page Events

Rule Definition

When Button is Clicked

always

then complete the following one after another

then ExpDD_OrderHeader, execute its Create method (configure)

then on ExpDD_OrderDetails execute the Create method for all the items that have been Added (configure)

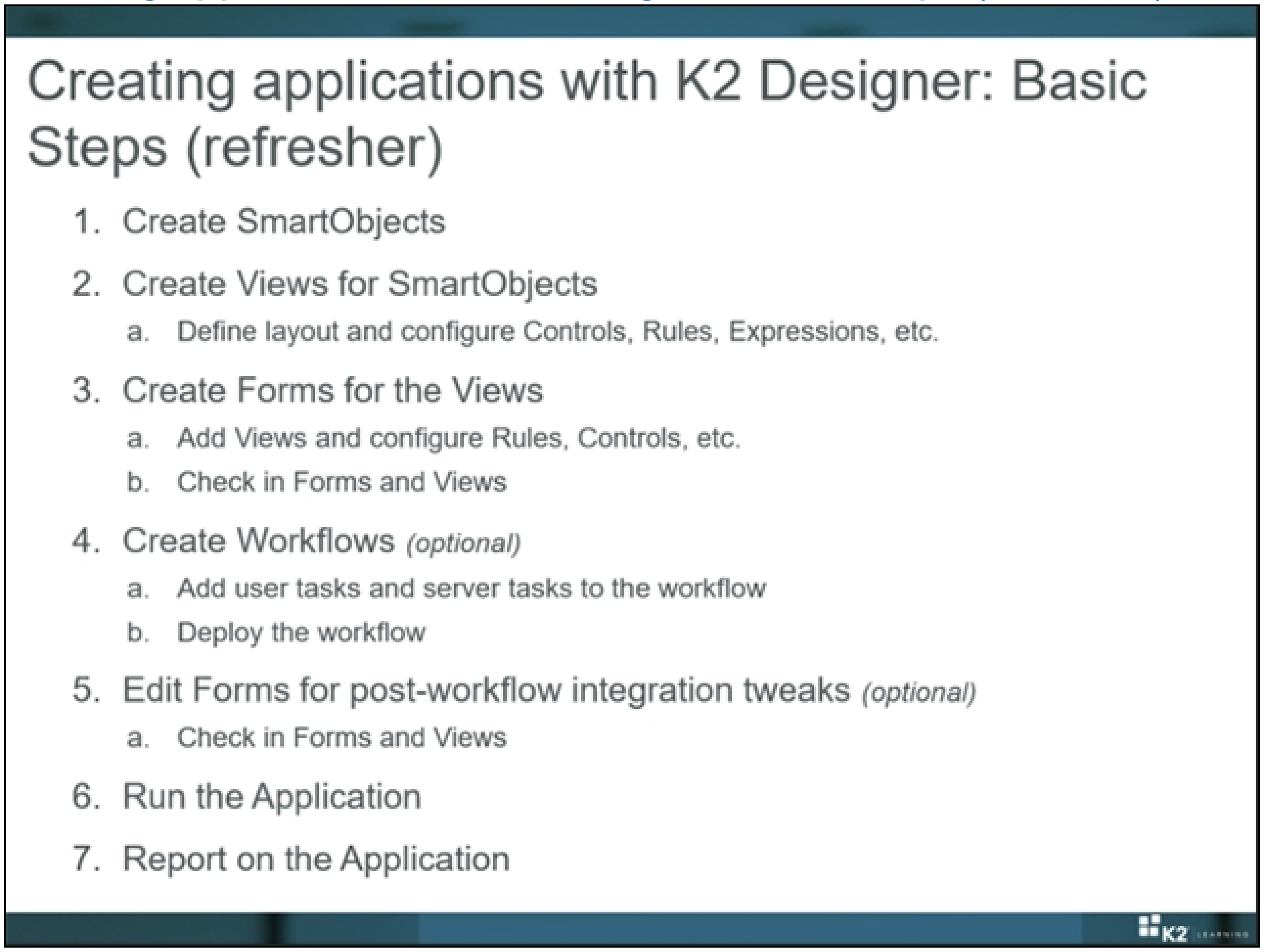
if (LessThanEquals 2) And (GreaterThan 1000) is true

then start the DOdgyCustomerOrderApproval Process (configure)

Summary

- The basic components and concepts of SmartForms are:
  - Forms which are pages that contain Views, Controls and Rules
  - Views, such as Item Views to show single records and List Views that show multiple records
    - Views can contain Rules and Controls
  - Controls, such as text boxes, labels, drop-down lists and buttons
  - Rules, which are the basic "programming " behind SmartForms
  - Expressions, which are used to perform calculations and data manipulation

Creating applications with K2 Designer: Basic Steps (refresher)



Creating applications with K2 Designer: Basic Steps (refresher)

1. Create SmartObjects
2. Create Views for SmartObjects
   a. Define layout and configure Controls, Rules, Expressions, etc.
3. Create Forms for the Views
   a. Add Views and configure Rules, Controls, etc.
   b. Check in Forms and Views
4. Create Workflows *(optional)*
   a. Add user tasks and server tasks to the workflow
   b. Deploy the workflow
5. Edit Forms for post-workflow integration tweaks *(optional)*
   a. Check in Forms and Views
6. Run the Application
7. Report on the Application

Video

This slide lists the basic steps you need to follow when creating applications with K2 Designer. You don't have to build applications in this order, but this is the usual approach.

We usually start with SmartObjects, then create Forms and Views for those SmartObjects, then create Workflows if any are needed. By this time the application is usually ready, although you would typically go and edit the Forms for the new Workflow integration to make them behave differently depending on where in the Workflow the Forms are being used.

You have to deploy Workflows before they will be available to your users. Deploying the Workflow "injects" the necessary rules into the Forms and Views. You have to check in Forms and Views before they are available to your users. (If you need to "promote" applications between environments, you would use the K2 Package and Deployment tool to do so.)

Once the application is deployed, you can run the application and report on the application.

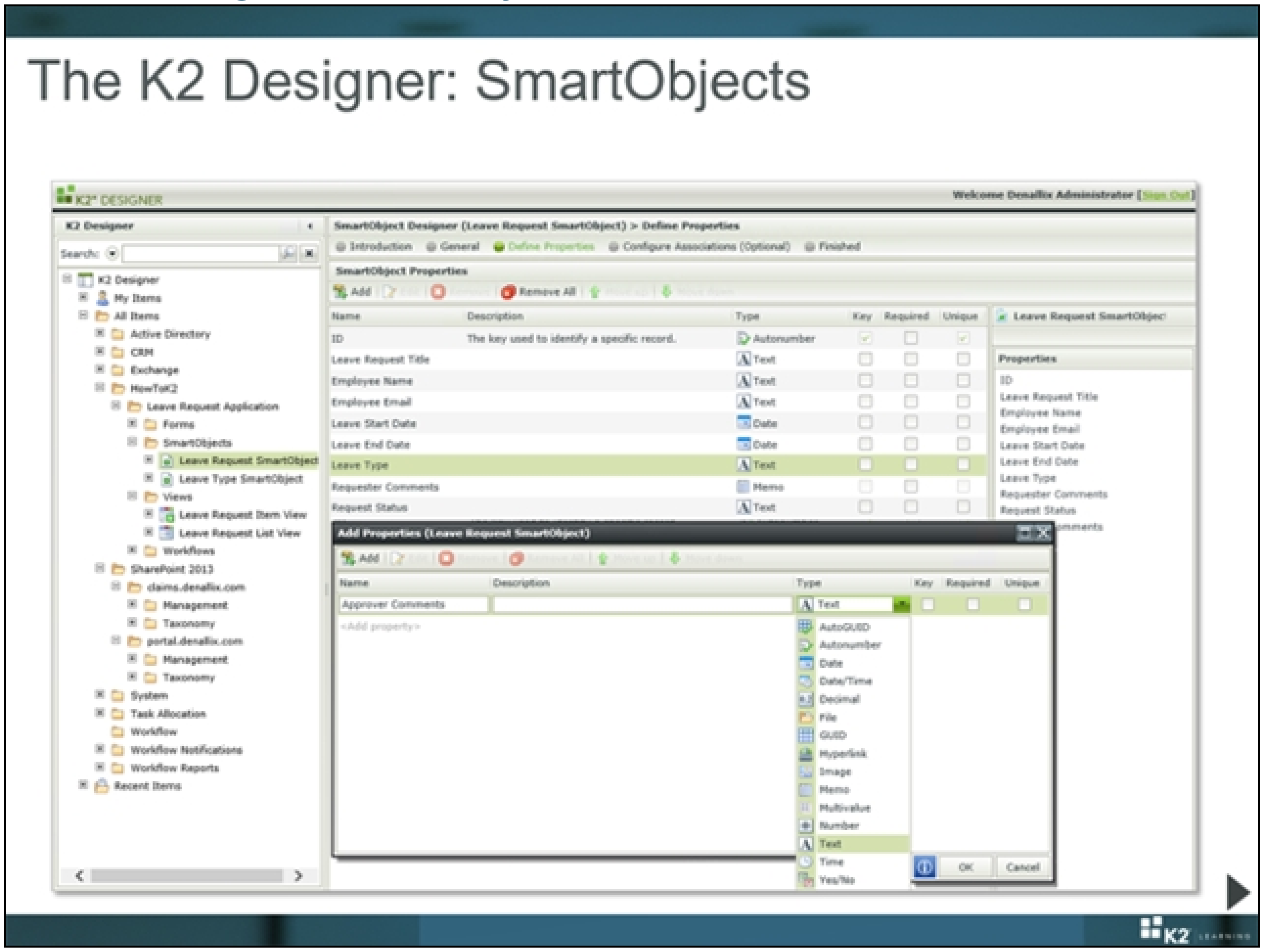
Note
If you need to promote/migrate applications between environments (such as moving an application from Development to Production), you would use the K2 Package and Deployment tool.

To summarize, here are the main steps to create applications with K2 Designer:

1. Create SmartObjects using tools like K2 Designer or K2 Studio. You could also auto-generate SmartObjects in some cases to integrate with other systems like SQL databases.
2. Create Views for SmartObjects using K2 Designer. In this step you would build a User Interface that exposes the SmartObjects you created in Step 1. As part of this step, you would also define the View layout, then add and configure Controls, Rules and Expressions. You may need to create multiple Views for the application.

3. Create Forms for the Views. You have to add Views to Forms since users interact with Forms. In this step, you would take the Views created in Step 2 and add them to a Form. You would also configure additional Rules and Controls on the Form as needed. Once this is done, check in the Forms and Views.
4. The next (optional) task is to create workflows. This is optional because not all applications have a workflow component but when they do, it is best to create the Forms used in the workflow first, before you start building the workflow. While building the workflow, you would add user tasks and server tasks and configure them using wizards. When the workflow is complete, deploy the workflow.
5. Next, you may need to check out and edit the Forms for post-workflow integration tweaks. This is also optional, but sometimes you may want to change the way in which workflows are integrated with your Forms or make some enhancements now that the Forms are workflow-aware. If you do change the Forms or Views, remember to check in the Forms and Views so that other users can see the changes you made.
6. Next, run the application to test it, or ask users to start testing the application. At this point you may want to set permissions on the workflow so that a large (or limited) group of users can start the workflow.
7. Once some instances of the application's workflow have started, you can use the available reports to report on the application.

The K2 Designer: SmartObjects



Name	Description	Type	Key	Required	Unique
ID	The key used to identify a specific record.	Autonumber	✓		✓
Leave Request Title		Text			
Employee Name		Text			
Employee Email		Text			
Leave Start Date		Date			
Leave End Date		Date			
Leave Type		Text			
Requester Comments		Memo			
Request Status		Text			

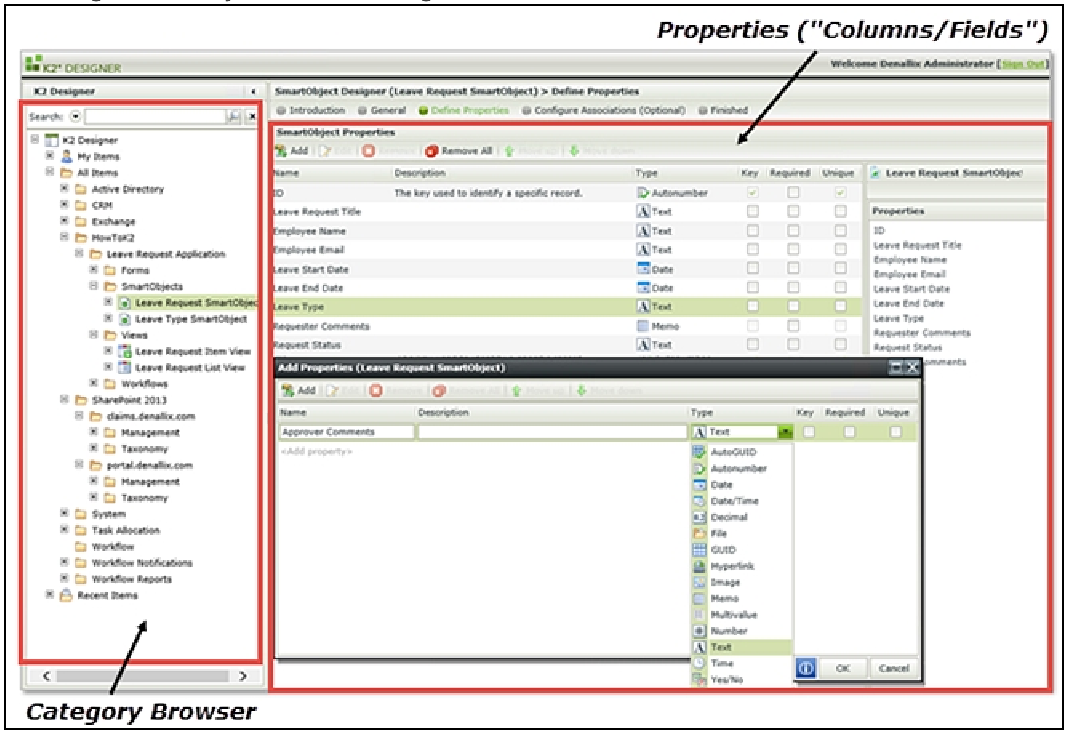
Video

When working with K2 Designer to build SmartObjects, there are two main areas in the design canvas: The Category Browser and the Properties Pane. We will look at the Category Browser in the next topic, for now let's just focus on the SmartForms Properties section.

The Properties pane is where you define the "columns" or fields, that make up your SmartObject. Here you can define the fields and the data types for the fields. For simple (i.e. K2 SmartBox-based SmartObjects) you will typically only define Properties (you may also know these as “columns” or “fields”) and perhaps associations (“relationships”) to other SmartObjects.

For advanced-mode SmartObjects (which we will look at later), the Properties Pane is also where you would configure the methods that point to particular back-end systems.

Building SmartObjects in K2 Designer



SmartObject Designer (Leave Request SmartObject) > Define Properties

SmartObject Properties

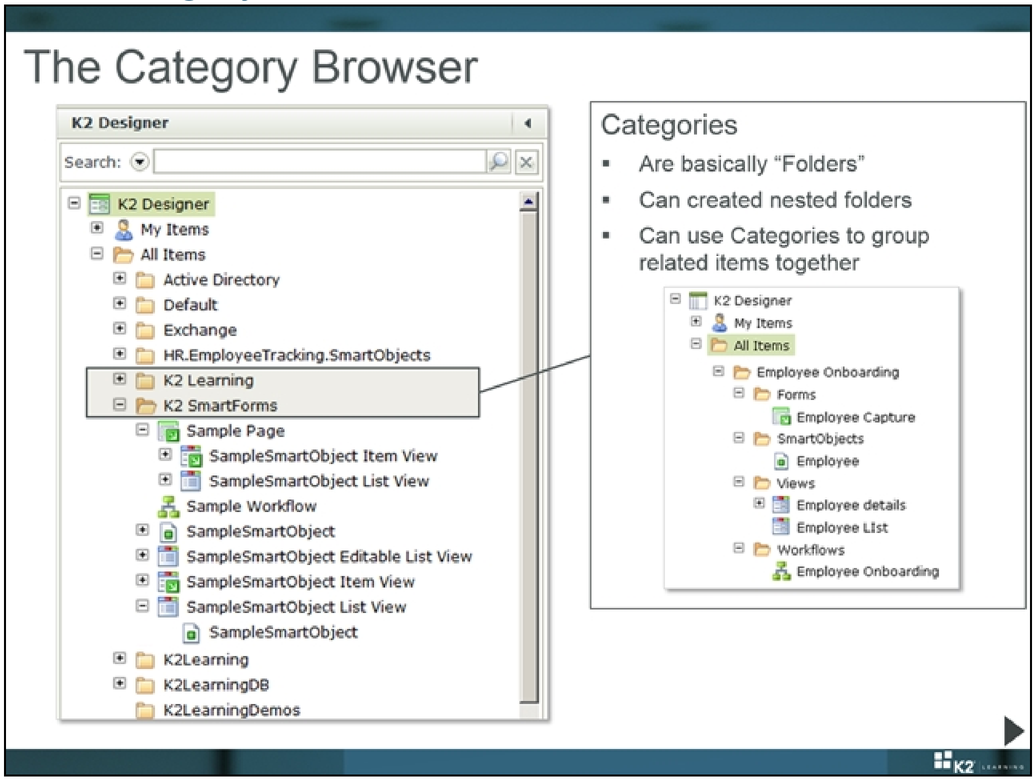
Name	Description	Type	Key	Required	Unique
ID	The key used to identify a specific record.	Autonumber	✓	☐	✓
Leave Request Title		Text	☐	☐	☐
Employee Name		Text	☐	☐	☐
Employee Email		Text	☐	☐	☐
Leave Start Date		Date	☐	☐	☐
Leave End Date		Date	☐	☐	☐
Leave Type		Text	☐	☐	☐
Requester Comments		Memo	☐	☐	☐
Request Status		Text	☐	☐	☐

Properties
ID
Leave Request Title
Employee Name
Employee Email
Leave Start Date
Leave End Date
Leave Type
Requester Comments
Request Status

Add Properties (Leave Request SmartObject)

Name	Description	Type	Key	Required	Unique
Approver Comments		Text	☐	☐	☐

The Category Browser


The Category Browser

Categories
- Are basically "Folders"
- Can created nested folders
- Can use Categories to group related items together

Video

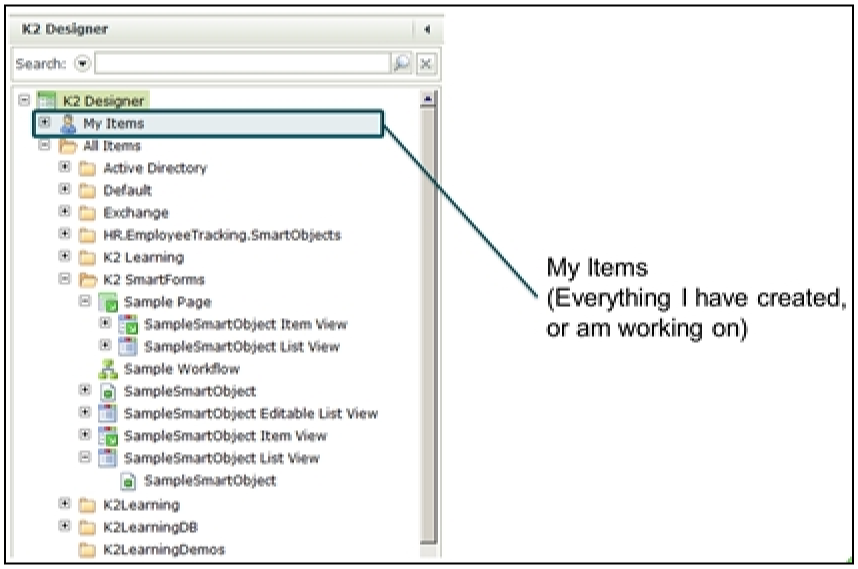
The Category Browser (also referred to as the Category System) is your entry point to the various SmartObjects, Views, Forms and Workflows that have been deployed to the K2 Server. The Category Browser allows you to explore the items that you have been working on. Think of this as a repository of items on the K2 Server.

In the Category Browser, different icons are used to represent different types of objects. There is also Expand/Collapse functionality to see the dependencies between items, and you can check Views and Forms in and out using the Category Browser. This is also where you can open items in Edit mode, Run Views and Forms to test them and review the Properties of the selected item.

Only the following artifacts will show up in the Category Browser:
- Forms
- Views
- SmartObjects
- Workflows designed with the K2 Designer

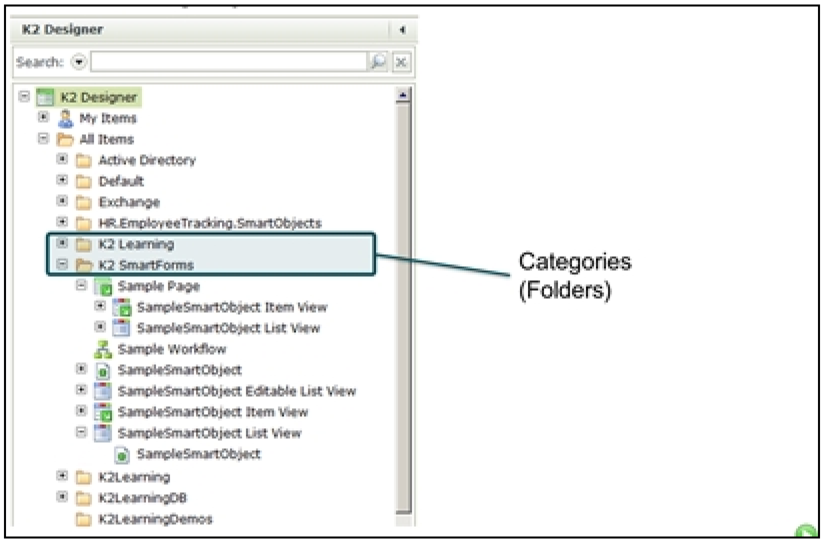
My Items

Shows all the items that you created or that you are working on (in other words, items that are checked out to you).


K2 Designer
Search:
K2 Designer
My Items
All Items
Active Directory
Default
Exchange
HR.EmployeeTracking.SmartObjects
K2 Learning
K2 SmartForms
Sample Page
SampleSmartObject Item View
SampleSmartObject List View
Sample Workflow
SampleSmartObject
SampleSmartObject Editable List View
SampleSmartObject Item View
SampleSmartObject List View
SampleSmartObject
K2Learning
K2LearningDB
K2LearningDemos
My Items
(Everything I have created, or am working on)

Categories/Folders

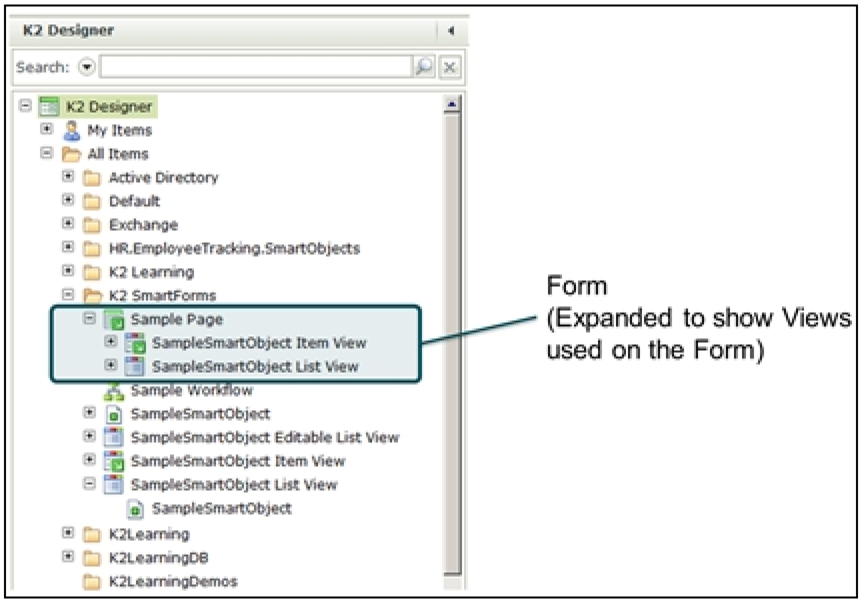
Containers for SmartObjects, Views, Forms and Workflows. Use Categories to organize your artifacts into easy-to-find structures.



Form

A Form contains Views. You can expand the Form to see a list of the Views used on the Form, and you can expand the Views to see the SmartObject used in each View.

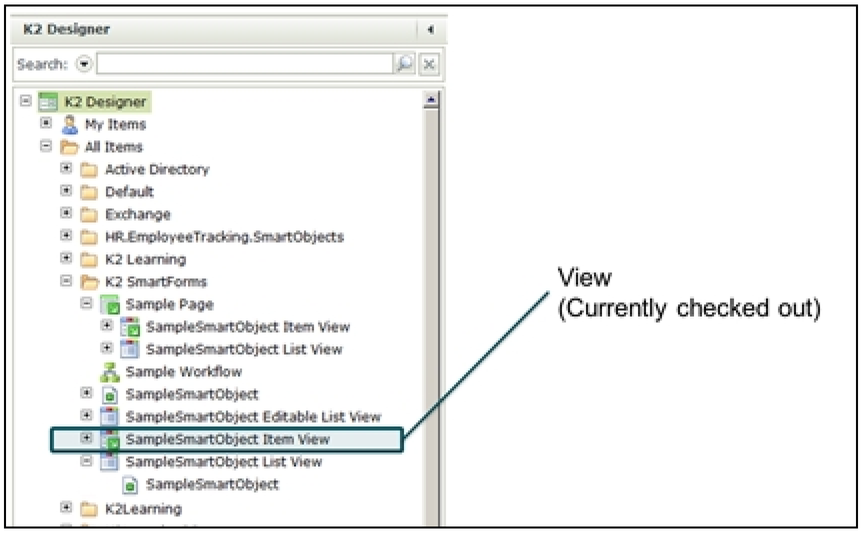
Forms can be checked out, and will not appear to other users until the Form is checked in. The Form icon will indicate whether the Form is currently checked out (look for the green arrow that indicates the Form is checked out).


K2 Designer
Search:
K2 Designer
My Items
All Items
Active Directory
Default
Exchange
HR.EmployeeTracking.SmartObjects
K2 Learning
K2 SmartForms
Sample Page
SampleSmartObject Item View
SampleSmartObject List View
Sample Workflow
SampleSmartObject
SampleSmartObject Editable List View
SampleSmartObject Item View
SampleSmartObject List View
SampleSmartObject
K2Learning
K2LearningDB
K2LearningDemos
Form
(Expanded to show Views used on the Form)

View

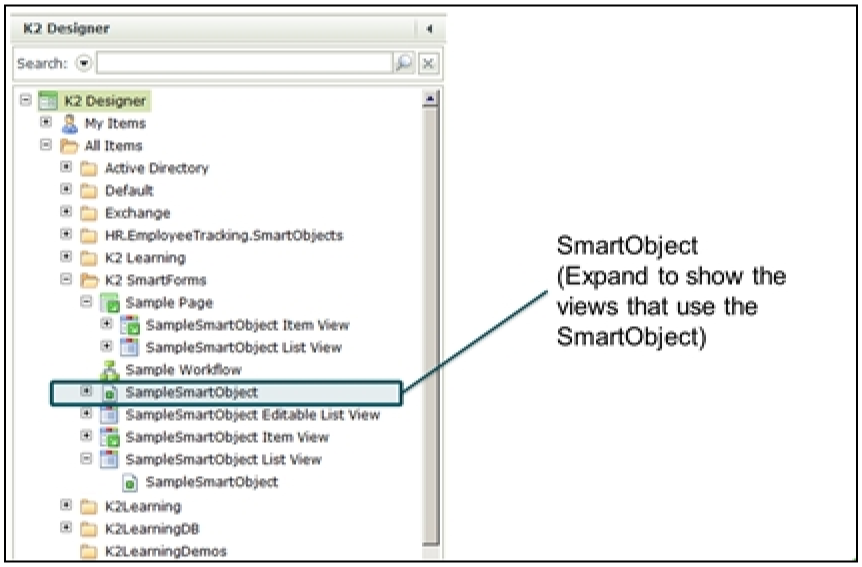
You can also expand the View to see the SmartObject that is used in the View. Views can be checked out, and will not appear to other designers or users until the View is checked in. The View icon will indicate whether the view is currently checked out (look for the green arrow that indicates the View is checked out).

If a View is used on a Form and the Form is checked in, the View on that Form will be checked in as well.



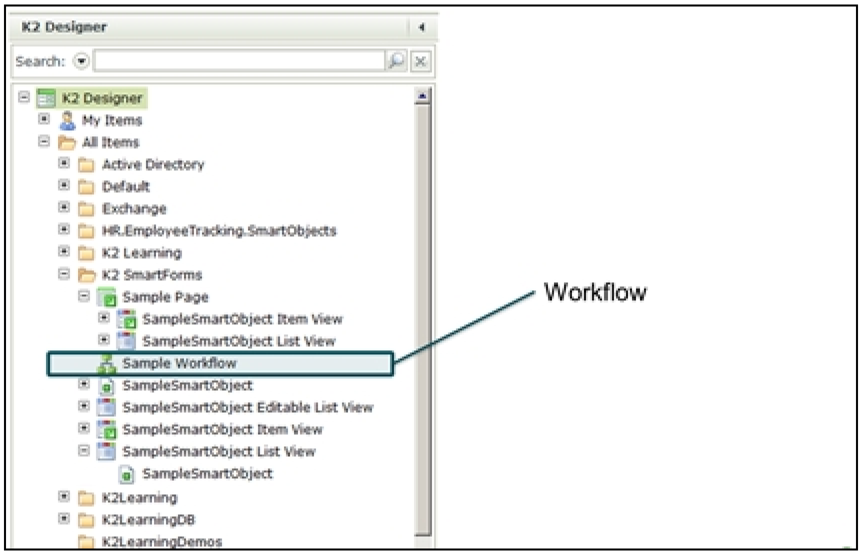
SmartObject

A SmartObject is a logical business entity which interacts with some data provider. SmartObjects are typically used on Views and in Workflows, and are never directly used by an end-user.


K2 Designer
Search:
K2 Designer
My Items
All Items
Active Directory
Default
Exchange
HR.EmployeeTracking.SmartObjects
K2 Learning
K2 SmartForms
Sample Page
SampleSmartObject Item View
SampleSmartObject List View
Sample Workflow
SampleSmartObject
SampleSmartObject Editable List View
SampleSmartObject Item View
SampleSmartObject List View
SampleSmartObject
K2Learning
K2LearningDB
K2LearningDemos
SmartObject (Expand to show the views that use the SmartObject)

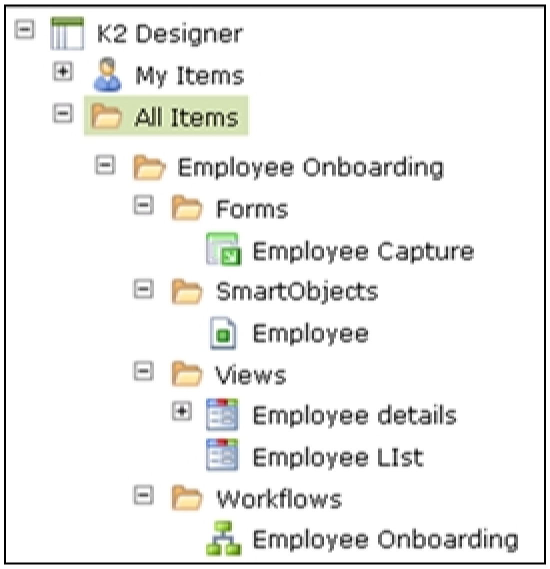
Workflow

A K2 Workflow designed with the web-based K2 Designer tool will appear in the Category Browser. Workflows designed with other design tools like K2 Studio or K2 for Visual Studio, will not appear in the Category Browser, since you cannot edit them using the browser-based (K2 Designer) workflow design tool.



Using sub-categories to group solution artifacts

You can also use sub-categories to group the artifacts in a solution. While you may choose your own categorization system, one recommended approach is to use categories to separate the Forms, Views, SmartObjects and Workflows in a solution, as shown in the example below.



Check-in, Check-out and Deploying

Some elements need to be checked out to make changes, and then checked-in or deployed before those changes are "published" to the K2 Server.

- Forms and Views must be checked in and checked out
- Users will not see any changes you've made until you check in the Form/View
- Other people cannot edit Forms/Views checked out by someone else
- You must check in Forms/Views before creating a package that will contain those items
- The green arrows indicate what is checked out
- SmartObjects and Workflows do not have to be checked in and checked out
- Workflows must be deployed before the changes will be visible on the K2 Server
- SmartObjects created in K2 Studio or K2 for Visual Studio must be deployed to the K2 Server. SmartObjects created in K2 Designer do not need to be deployed, they will be deployed when you click the Finish button.

Note
The Category Browser saves updates to the K2 Server, not to the developer's workstation. This means that if you use the Category Browser to edit a SmartObject, for example, when you save the SmartObject, it is immediately published to the K2 Server and available for other designers to use.

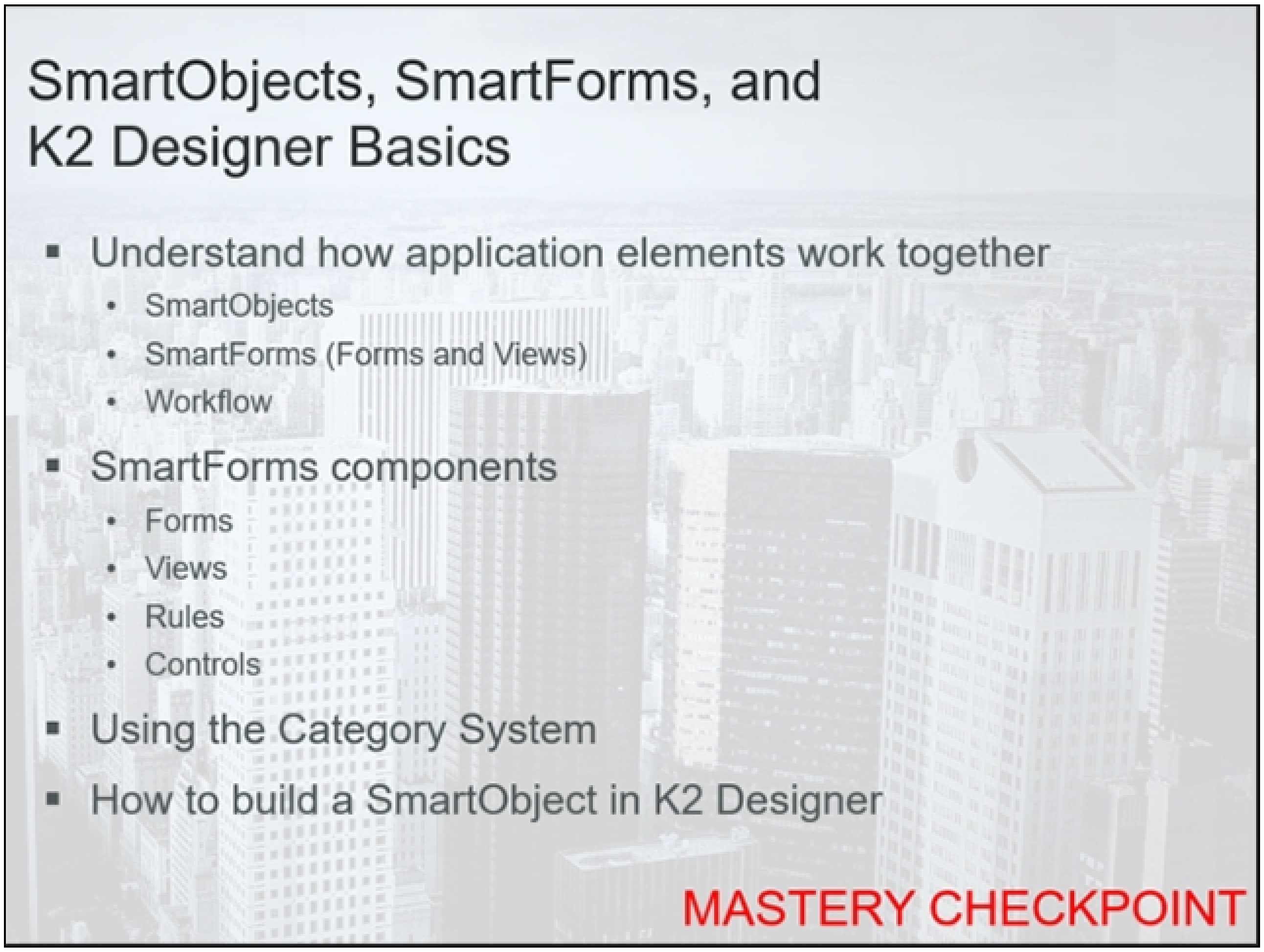
Bear in mind that if you create a SmartObject in K2 Studio, deploy the SmartObject to the K2 Server and then edit the SmartObject further in the K2 Designer, the original SmartObject definition in the K2 Studio project on your machine won't be updated (because the K2 Server cannot modify files local to the developer's workstation). If you were to deploy the original K2 project from your workstation again, it would overwrite the changes you made to the SmartObject in K2 Designer.

Summary

- The Category Browser is the repository of all items in the SmartForms environment
- The items you worked on will be grouped in the "My Items" folder
- You can create Categories ("folders") to group and organize related items together
- Categories can be nested
- The icons show what kind of item it is
- Only the following will show in the category system:
  - Forms
  - Views
  - SmartObjects
  - Workflows designed with the K2 Designer
- Check-in and Check-out:
  - Forms and Views must be checked in and checked out
  - SmartObjects and Workflows do not have to be checked in and checked out
  - Users will not see any changes you've made until you check in the Form/View

- Other people cannot edit Forms/Views checked out by someone else
- You must check in Forms/Views before creating a package that will contain those items
- The green arrows indicate what is checked out

MASTERY CHECKPOINT: SmartObjects, SmartForms and K2 Designer Basics



SmartObjects, SmartForms, and K2 Designer Basics

- Understand how application elements work together
  - SmartObjects
  - SmartForms (Forms and Views)
  - Workflow
- SmartForms components
  - Forms
  - Views
  - Rules
  - Controls
- Using the Category System
- How to build a SmartObject in K2 Designer

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 1 of this module. If you are attending instructor-led training, use this topic as an opportunity to ask any questions around the basics of K2 SmartForms and K2 Designer.

By now you should know:

- How SmartObjects, SmartForms and Workflows work together in a K2 Application
- The basic components of SmartForms (Views, Forms, Rules and Controls)
- Working in the K2 Designer Category System
- Building a SmartObject in K2 Designer

Knowledge-check questions

Q: Do you use Active Directory in your organization?
A: (discussion question)

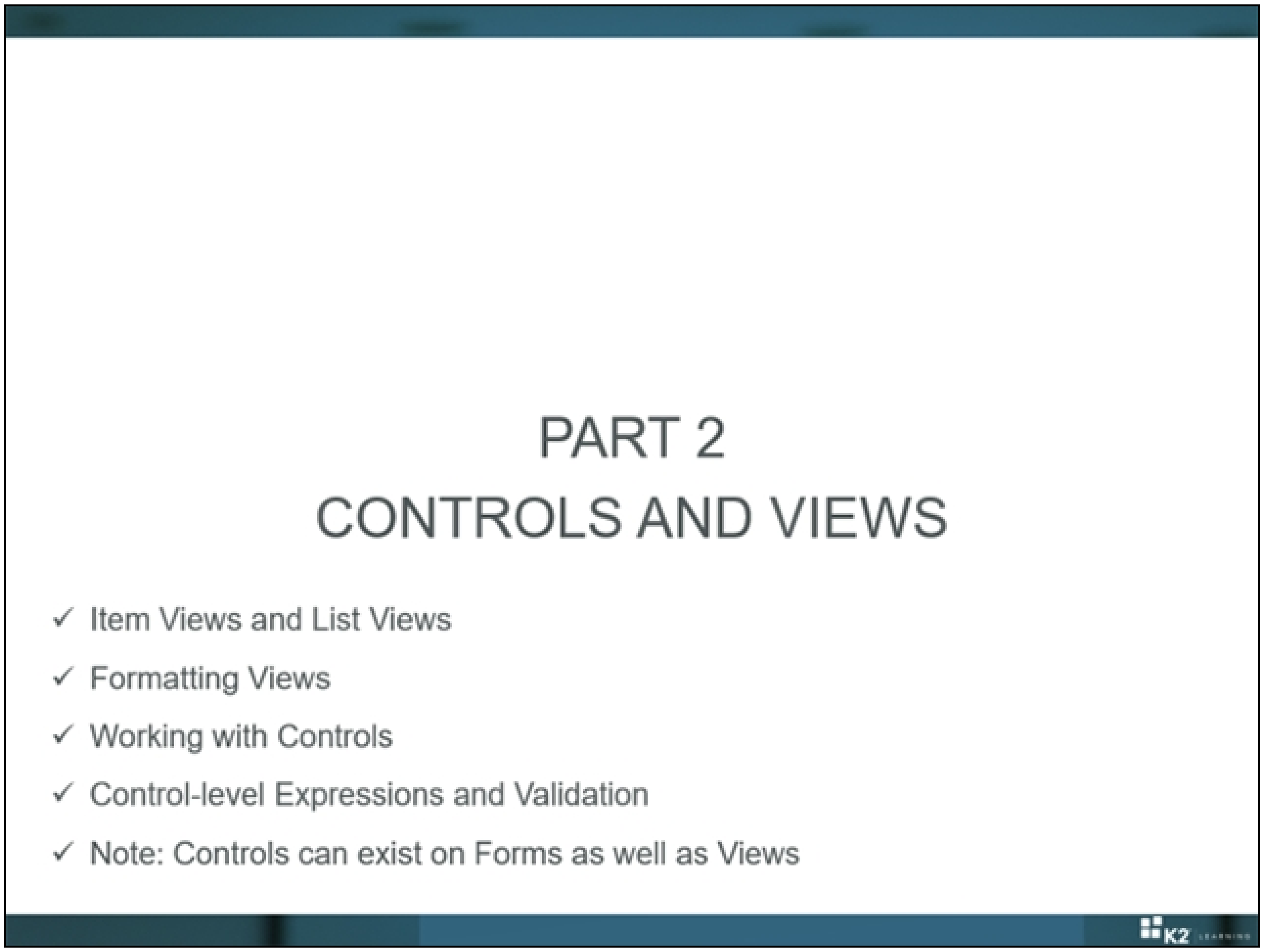
Q: Why would one use folders in the Category System?
Reveal answer**A:** One use case is to keep the same types of components (SmartObjects, Forms, Views, Workflows) grouped together in the same folder. Another use case is to separate the components of specific applications, such as keeping everything that makes up an application all together.

Q: Do you understand the difference between “Views” and “Forms”?
Reveal answer**A:** Forms contain one or more Views. You can re-use the same View on different Forms. Users usually interact with Forms.

Q: Can you re-use the same View on multiple Forms?
Reveal answer**A:** Yes, and this is in fact a recommended approach to reduce the development time for new Forms, because those new Forms may be able to re-use an existing View.

Q: True or False: users will not be able to see the changes you made to a Form until you have checked it in.
Reveal answer**A:** True, only the user who has checked out a Form will see any new changes, you have to check a Form in before other users will see those changes.

PART 2: Controls and Views


PART 2
CONTROLS AND VIEWS

- ✓ Item Views and List Views
- ✓ Formatting Views
- ✓ Working with Controls
- ✓ Control-level Expressions and Validation
- ✓ Note: Controls can exist on Forms as well as Views

K2 LEARNING

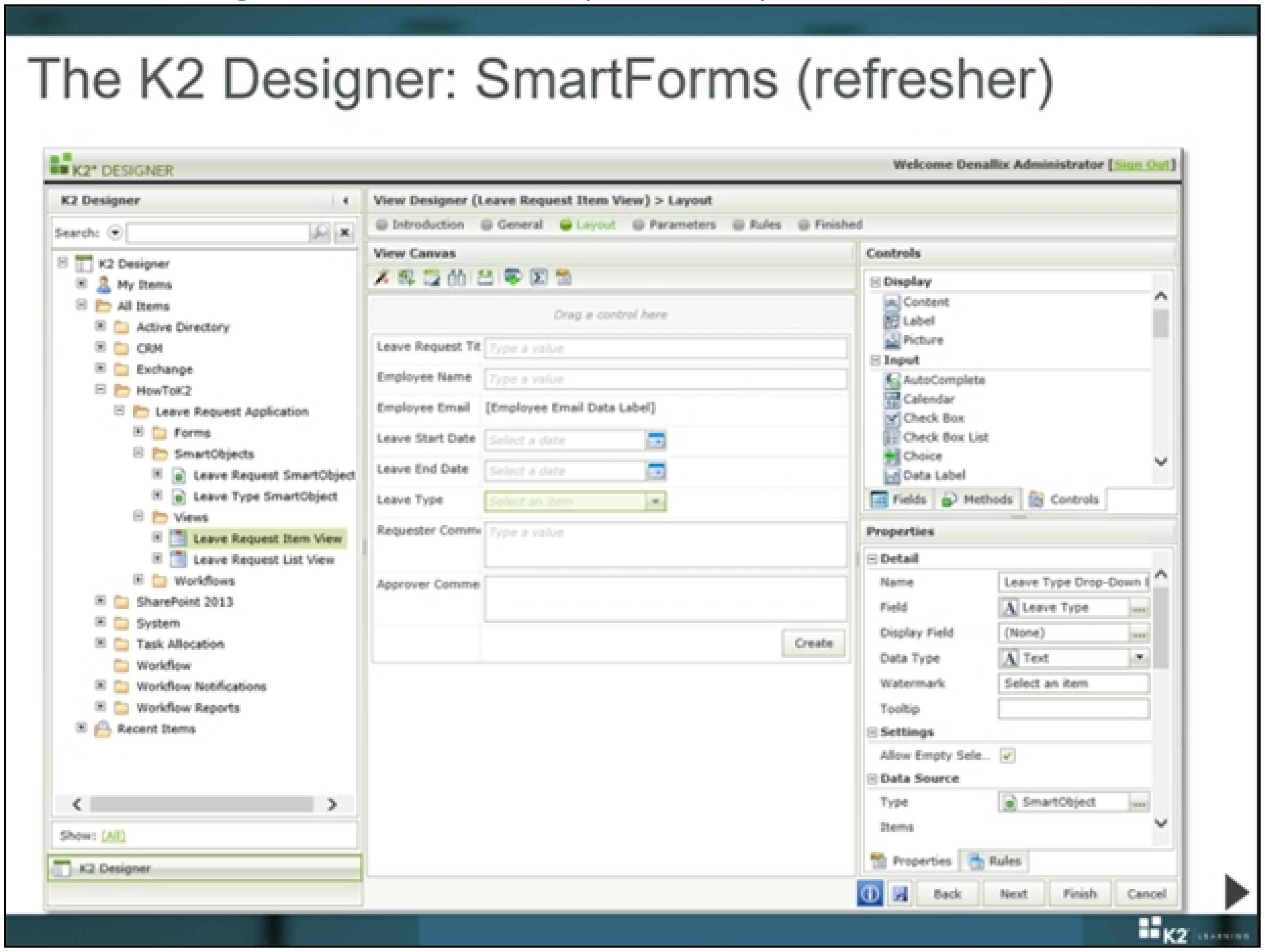
Video

In Part 2 we will look at Views in more detail, specifically List Views and Item Views.

- We will describe how to format Views with the table layout
- We will explain how to work with Controls, Control-level Expressions and Control-level Validations
- At the end of Part 2, we will build two Views
  - A List View to show AD contact information for multiple users (read-only)
  - An Item View to show AD contact information for a specific user (with Controls to allow users to edit the information)
  - These Views will eventually be used on the Forms of our application

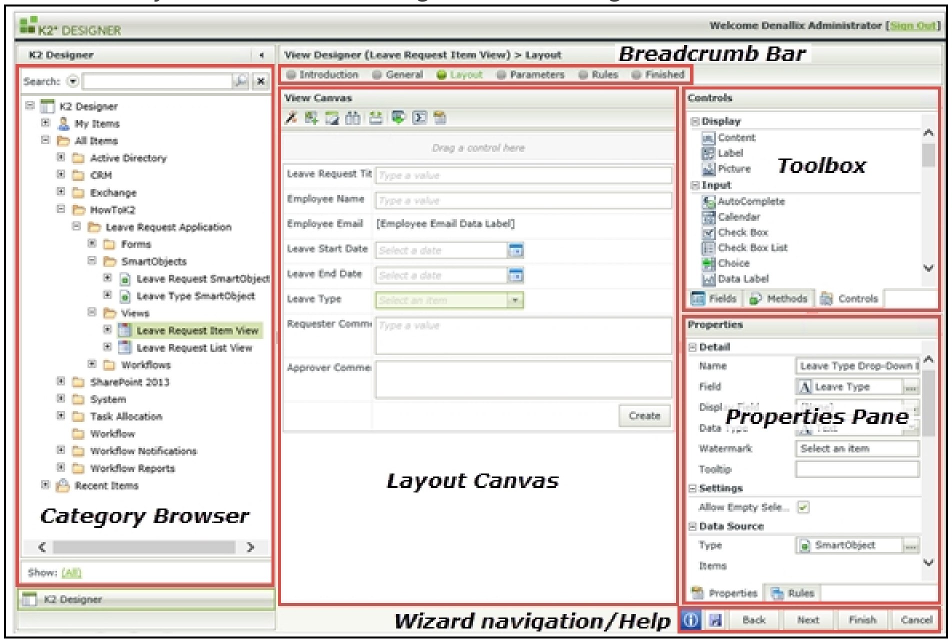
Note
Controls can exist on Forms, too. We are only grouping Views and Controls together in Part 2 because of the way the exercises are structured.

The K2 Designer: SmartForms (refresher)


The K2 Designer: SmartForms (refresher)
K2 DESIGNER
Welcome Denallix Administrator [Sign Out]
View Designer (Leave Request Item View) > Layout
Introduction
General
Layout
Parameters
Rules
Finished
View Canvas
Controls
Properties
Back
Next
Finish
Cancel

The K2 Designer is the only tool you can use to design SmartForms. Let's take a few minutes to get familiar with the Form designer screens when building SmartForms.

The basic layout of the Form designer in K2 Designer



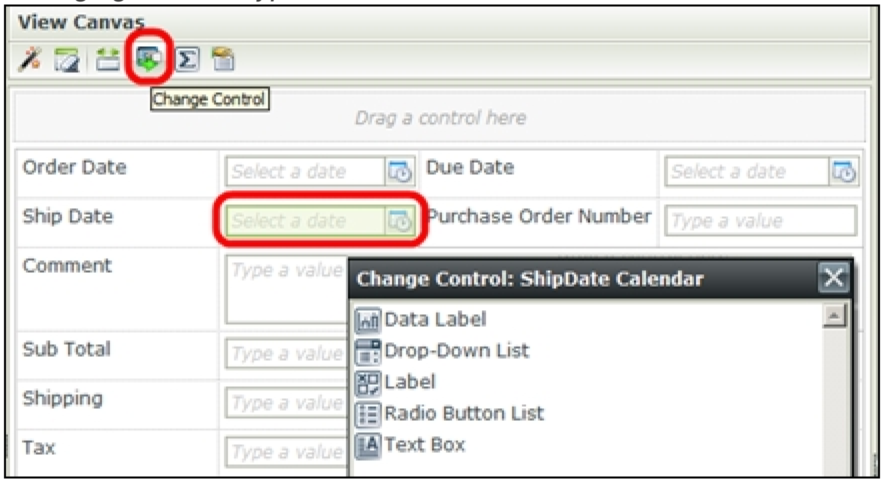
The **Layout Canvas** is the design area where a Form or View layout is designed. Think of this as the “layout pane” or “editing pane”. When designing a SmartForm, this where you will drag and drop Controls and tables in designing your User Interface.

Just above the Canvas, you will find the **Breadcrumb Bar**. This is a very useful feature, which allows you to jump quickly between various configuration screens of the View or Form, such as defining the Settings, Layout and Rules. Remember to use the Breadcrumb Bar: it will make your life easier when designing SmartForms.

The **Toolbox** is where you will find the SmartObject Fields, Methods and Controls that can be added to the design canvas. Think of the Toolbox as a repository of the things you can potentially add to the View or Form. Notice that the Toolbox has three Tabs: **Fields**, **Methods** and **Controls**.

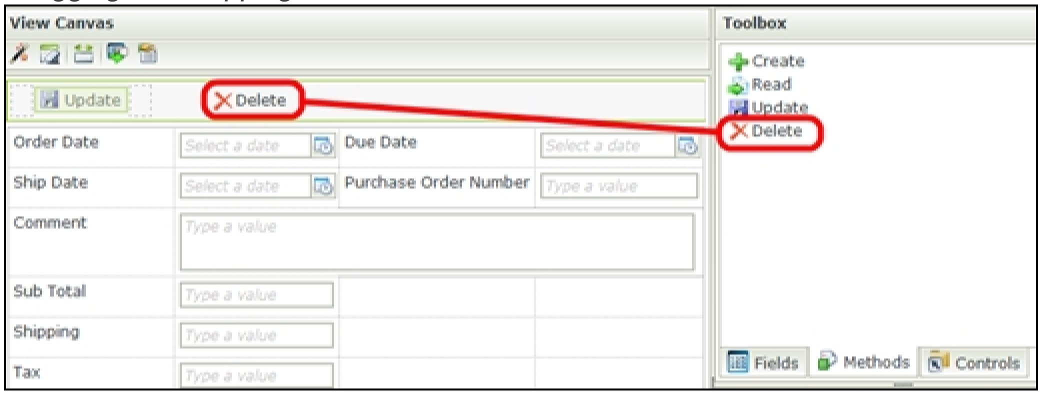
Fields are the properties of the current SmartObject. You can drag these properties onto the design canvas and K2 will automatically generate an appropriate Control for the type of field (for example a Date picker Control for a DateTime property). If required, you can use the **Change Control** menu button to change the Control into some other type.

Changing a control type



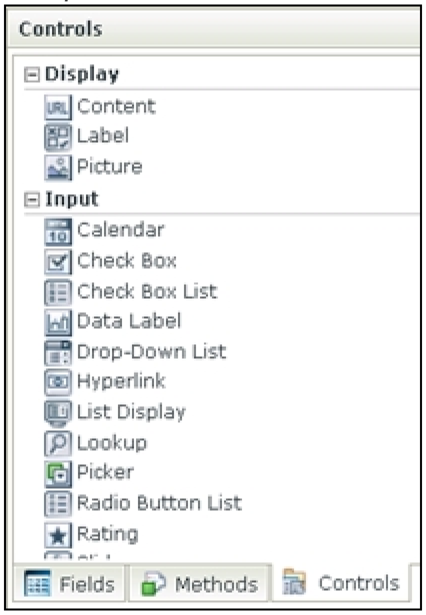
Methods contain the methods that have been defined for the SmartObject. You can drag and drop methods into the View or onto the View Toolbar to add a Control that is automatically wired to execute that method of the SmartObject when it is clicked.

Dragging and dropping a method onto the View



Controls contains a selection of Controls like text boxes, labels, drop-down menus, file attachments, picture boxes and more, that can be dragged and dropped onto the design canvas. You would normally configure the Controls after dropping them on the canvas with the options found in the Properties Pane.

Samples of some Controls

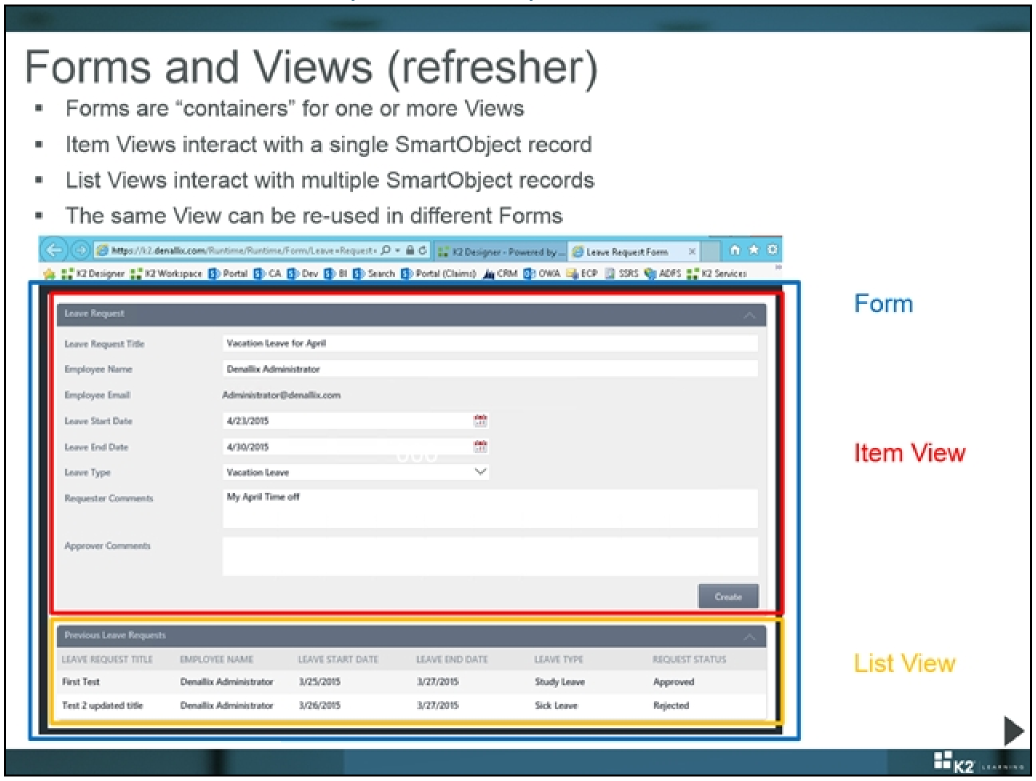


When you select a Control on the View, the Control's properties will appear in the Properties Pane. Here you can set the style of the Control, which field to display, validation Rules and other Rules that will fire when the Control is selected. The available options in the Properties Pane will differ depending on the type of Control selected.

Summary

- K2 Designer is where you design SmartForm Forms and Views
- The Designer is separated into different areas, as shown by the images above
- The Category Browser is essentially a “folder structure” of artifacts
- The Breadcrumb Bar allows you to jump between different configuration screens
- The Design Canvas is where you design the View and Form layouts
- The Toolbox is where you find Control, Method and Field properties from the SmartObject associated with this View
- The Properties Pane is where you configure the item that is selected in the canvas
- Navigation buttons allow you to step through the configuration wizards

Forms and Views (refresher)



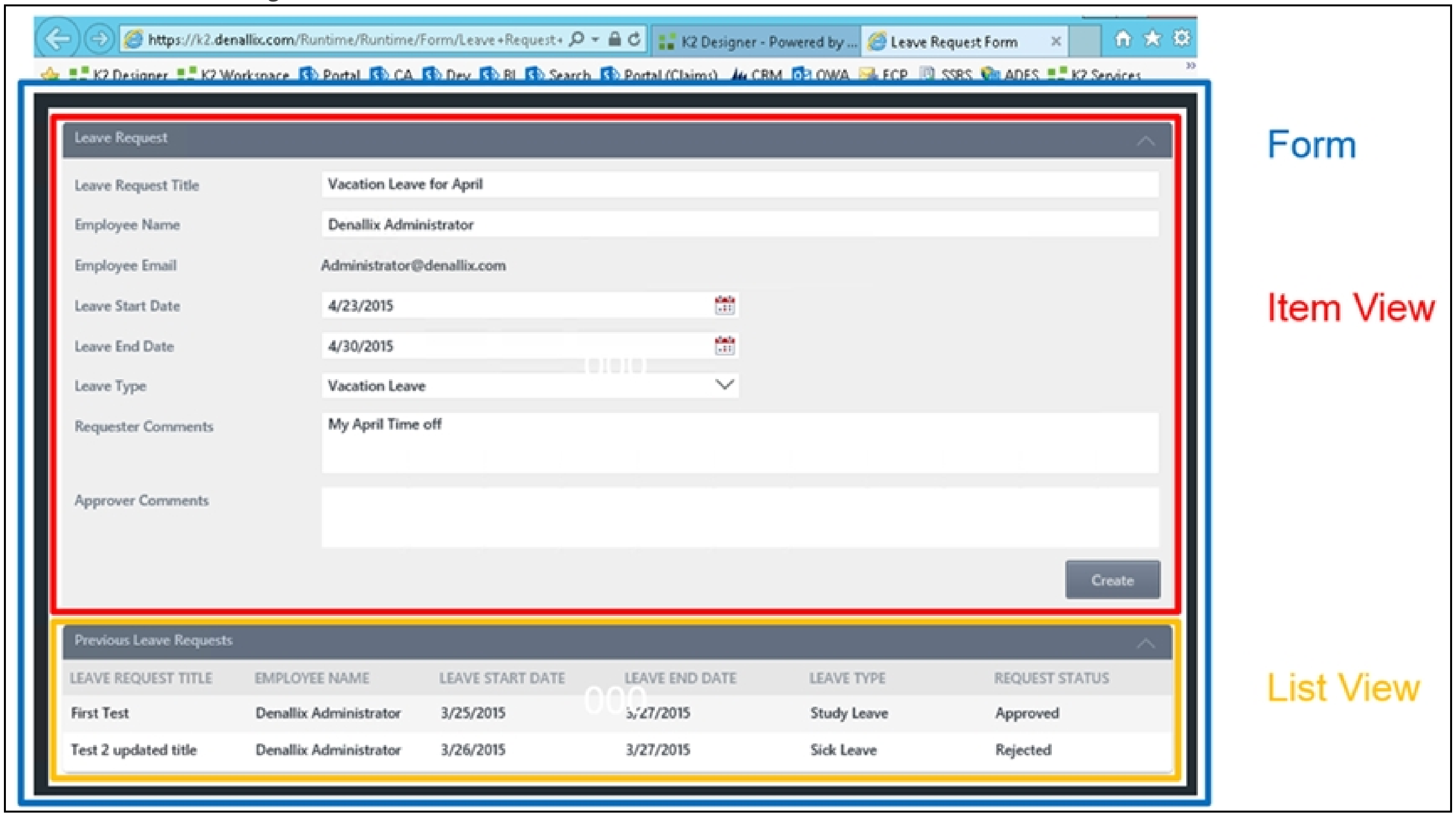
Forms and Views (refresher)

- Forms are "containers" for one or more Views
- Item Views interact with a single SmartObject record
- List Views interact with multiple SmartObject records
- The same View can be re-used in different Forms

Video

As mentioned previously, Forms are essentially the "web pages" that end users interact with, and these Forms are exposed as URLs. A Form is a container for one or more Views, and the majority of design and configuration actually lives on the View level. The benefit of this approach is that you can create re-usable Views that can be used on multiple Forms.

A SmartForm showing the Form with two Views: an Item View and a List View



Leave Request

Leave Request Title	Vacation Leave for April
Employee Name	Denallix Administrator
Employee Email	Administrator@denallix.com
Leave Start Date	4/23/2015
Leave End Date	4/30/2015
Leave Type	Vacation Leave
Requester Comments	My April Time off
Approver Comments	

Previous Leave Requests

LEAVE REQUEST TITLE	EMPLOYEE NAME	LEAVE START DATE	LEAVE END DATE	LEAVE TYPE	REQUEST STATUS
First Test	Denallix Administrator	3/25/2015	3/27/2015	Study Leave	Approved
Test 2 updated title	Denallix Administrator	3/26/2015	3/27/2015	Sick Leave	Rejected

Views are the main components of Forms and as such, you will spend much of your time configuring View layouts and behavior before adding those Views onto a Form. You can add the same View onto different Forms, and this is a nice way of creating re-usable Views and managing those Views in one place. Here is an example: suppose you create a View that displays basic Employee data. You could use this Employee data View on multiple Forms like a Leave Request Form and an Expense Claim Form. If you were to add another data field to the Employee data View (their Department name, for example), this new field will automatically show up on all the Forms where that View is used, because the Forms only point to the View, they do not actually copy the View.

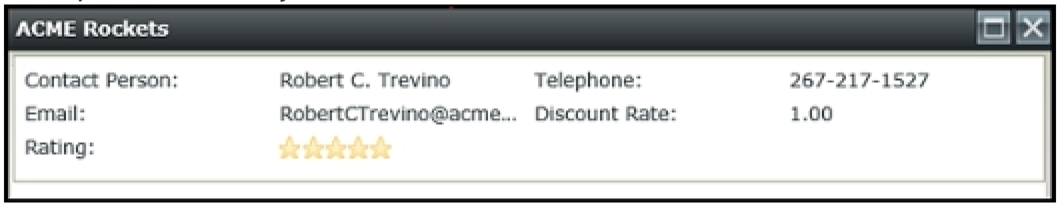
Views are the building blocks of Forms and are used to display data, capture user input and host other controls such as images, buttons, labels and more. Views are normally associated with at least one SmartObject, and may expose several methods of that SmartObject. Secondary SmartObjects can also be used on the View, for example, using a SmartObject to populate a drop-down list on the View. You do not have to associate a View with a SmartObject, however.

As mentioned previously, there are essentially two types of Views: Item Views and List Views:

Item Views

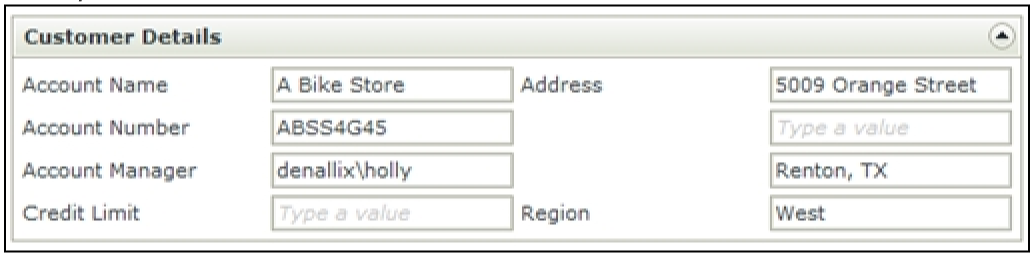
Item Views are used to display or interact with a single SmartObject "record" or "entry". These Views are normally used in conjunction with scalar SmartObject methods like Create, Update, Delete and Save. In the example below, we are displaying custom information for a specific customer.

Example of a read-only Item View


ACME Rockets

Contact Person:	Robert C. Trevino	Telephone:	267-217-1527
Email:	RobertCTrevino@acme...	Discount Rate:	1.00
Rating:	★★★★★		

Example of an editable Item View


Customer Details

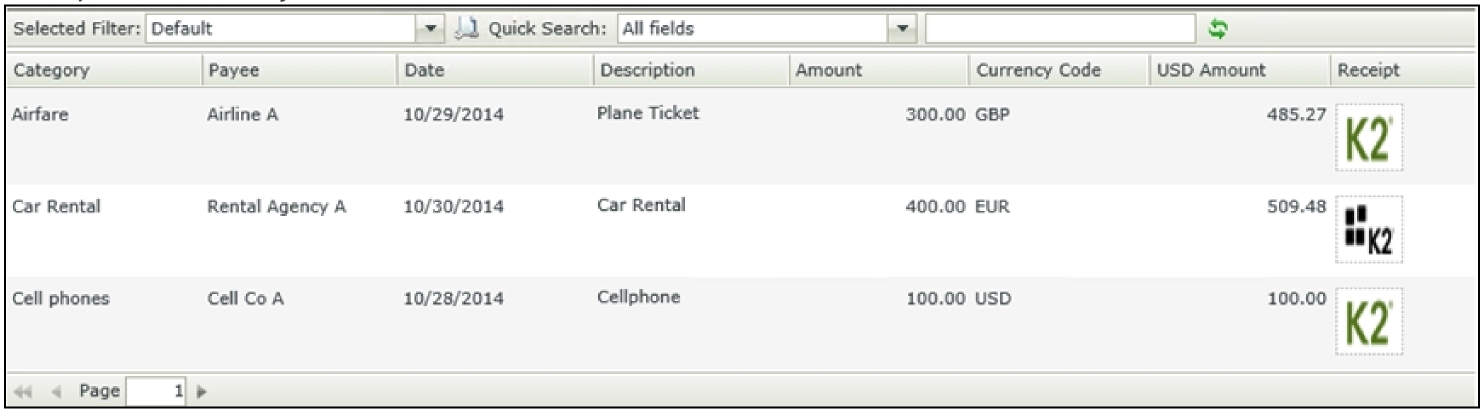
Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

List Views

A List View shows a list of SmartObject "records" or "entries". List Views are used in conjunction with SmartObject List-type methods. Just like Item Views, List Views could be read-only or editable.

Consider the example List View below, which displays a list of Expense Claim line items in a grid-like table.

Example of a read-only List View

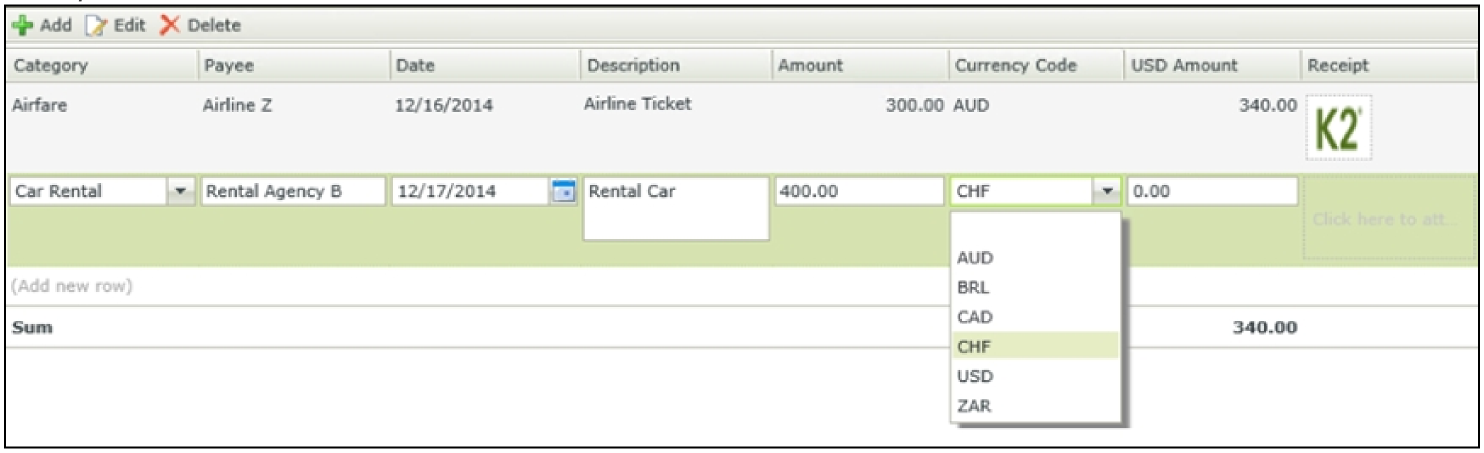

Selected Filter: Default Quick Search: All fields

Category	Payee	Date	Description	Amount	Currency Code	USD Amount	Receipt
Airfare	Airline A	10/29/2014	Plane Ticket	300.00	GBP	485.27	K2
Car Rental	Rental Agency A	10/30/2014	Car Rental	400.00	EUR	509.48	K2
Cell phones	Cell Co A	10/28/2014	Cellphone	100.00	USD	100.00	K2

Page 1

An editable List View is a very powerful feature in SmartForms, and users will love this approach to data entry when adding multiple entries for the same type of SmartObject. Consider the example below: here we are displaying a list of line items for an Expense Claim. The user can edit the List View by adding and removing items on the list and editing existing items in the list. Behind the scenes, K2 will use the SmartObject's appropriate methods (e.g. Create, Save, Delete) to apply the changes to the SmartObject.

Example of an Editable List View

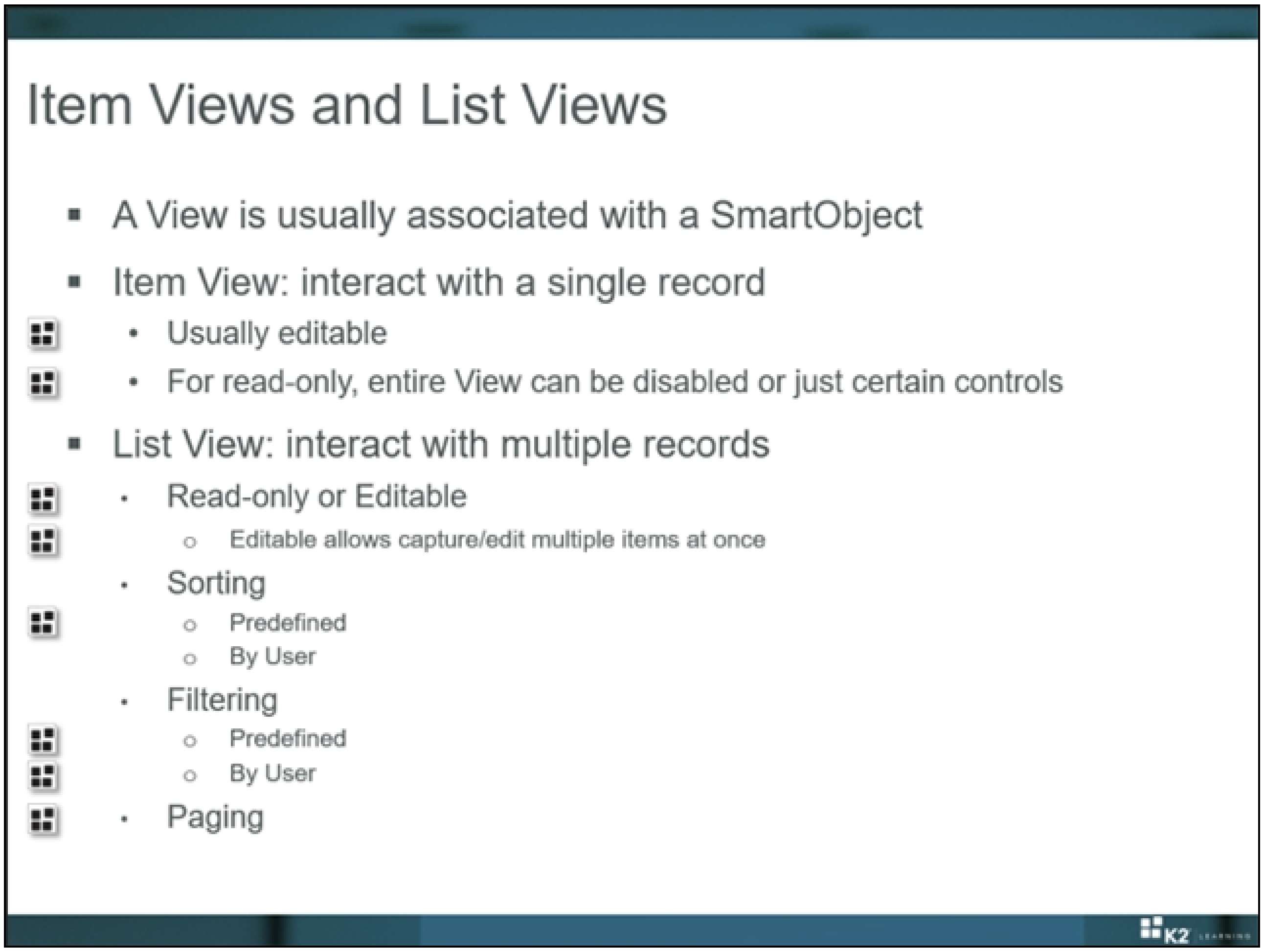


Category	Payee	Date	Description	Amount	Currency Code	USD Amount	Receipt
Airfare	Airline Z	12/16/2014	Airline Ticket	300.00	AUD	340.00	K2
Car Rental	Rental Agency B	12/17/2014	Rental Car	400.00	CHF	0.00	Click here to att...
(Add new row)							
Sum						340.00	

Discussion Points:

- A Form is a container for one or more Views.
- You can re-use the same View on different Forms.
- A Form can contain multiple Views.
- Item Views interact with a single record.
- List Views interact with multiple records.

Item Views and List Views



Item Views and List Views

- A View is usually associated with a SmartObject
- Item View: interact with a single record
  - Usually editable
  - For read-only, entire View can be disabled or just certain controls
- List View: interact with multiple records
  - Read-only or Editable
    - Editable allows capture/edit multiple items at once
  - Sorting
    - Predefined
    - By User
  - Filtering
    - Predefined
    - By User
  - Paging

Video

Let's explore Views in a little more detail. Views are the main components of Forms and as such, you will spend much of your time configuring View layouts and behavior before adding the Views to a Form. You can add the same View to different Forms and this is a nice way of creating re-usable Views and managing those Views in one place. Here is an example: suppose you create a View that displays basic Employee data. You could use this Employee data View on multiple Forms like a Leave Request Form or an Expense Claim Form. If you were to add another field to the Employee data View (their Department name, for example), this new field will automatically show up on all the Forms where that View is used, because the Forms only point to the View, they do not actually copy the View.

Tip
If required, you can create a copy of a Form or View using the "Save As" command. This essentially creates a copy of the View/Form which you can then edit further without affecting the original View or Form.

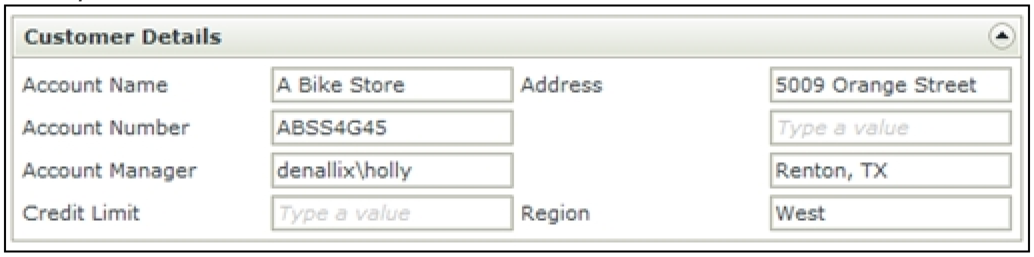
Views are the building blocks of Forms and are used to display data, capture user input and host other Controls such as images, buttons, labels and more. Views are normally associated with at least one SmartObject and may expose several methods of that SmartObject. Secondary SmartObjects can also be used on the View, for example, using a SmartObject to populate a drop-down list. You do not have to associate a View with a SmartObject, however.

As mentioned previously, there are essentially two types of Views: Item Views and List Views

Item View

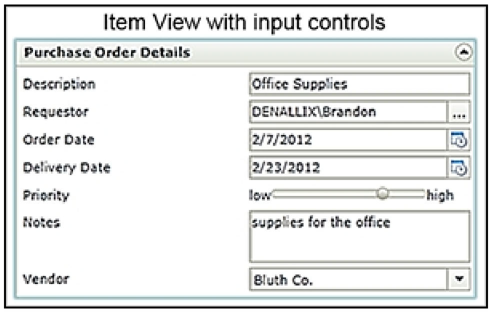
Item Views are used to display or interact with a single SmartObject “record” or “entry”. These Views are normally used in conjunction with scalar SmartObject methods like Create, Update, Delete and Save. In the example below, we are displaying custom information for a specific customer.

Example of an editable Item View



Customer Details

Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABSS4G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West



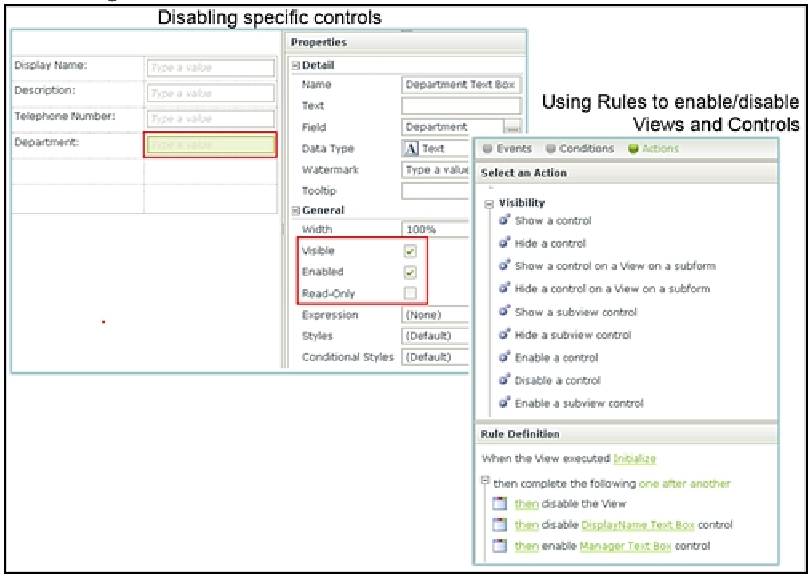
Item View with input controls

Purchase Order Details

Description	Office Supplies
Requestor	DENALLIX\Brandon
Order Date	2/7/2012
Delivery Date	2/23/2012
Priority	low – high
Notes	supplies for the office
Vendor	Bluth Co.

Item Views can be displayed in an editable format as well as a read-only format. The difference is purely in how the Controls in the View are set up: you as the View designer can decide whether to enable specific fields for input or not, using the Control's Enabled property. You can even use Rule Actions to disable all Controls on a View if you want to "lock down" the View on-demand, so that no data entry can be performed.

Disabling Controls and Views



Disabling specific controls

Display Name:	Type a value
Description:	Type a value
Telephone Number:	Type a value
Department:	Type a value

Properties

Detail

Name	Department Text Box
Text	
Field	Department
Data Type	Text
Watermark	Type a value
Tooltip	

General

Width	100%
Visible	✓
Enabled	✓
Read-Only	
Expression	(None)
Styles	(Default)
Conditional Styles	(Default)

Using Rules to enable/disable Views and Controls

Events Conditions Actions

Select an Action

Visibility
- Show a control
- Hide a control
- Show a control on a View on a subform
- Hide a control on a View on a subform
- Show a subview control
- Hide a subview control
- Enable a control
- Disable a control
- Enable a subview control

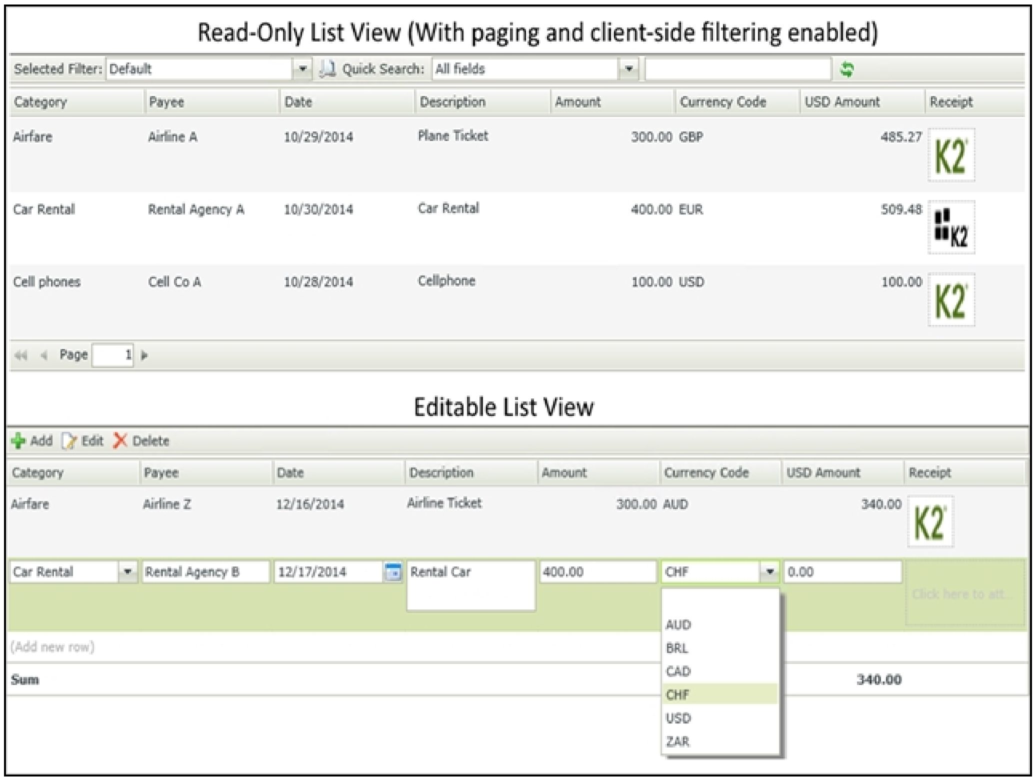
Rule Definition

When the View executed Initialize
then complete the following one after another
- then disable the View
- then disable DisplayName Text Box control
- then enable Manager Text Box control

List View

A List View shows a list of SmartObject "records" or "entries".

Example of List Views



Read-Only List View (With paging and client-side filtering enabled)

Selected Filter: Default | Quick Search: All fields

Category	Payee	Date	Description	Amount	Currency Code	USD Amount	Receipt
Airfare	Airline A	10/29/2014	Plane Ticket	300.00	GBP	485.27	K2
Car Rental	Rental Agency A	10/30/2014	Car Rental	400.00	EUR	509.48	K2
Cell phones	Cell Co A	10/28/2014	Cellphone	100.00	USD	100.00	K2

Page 1

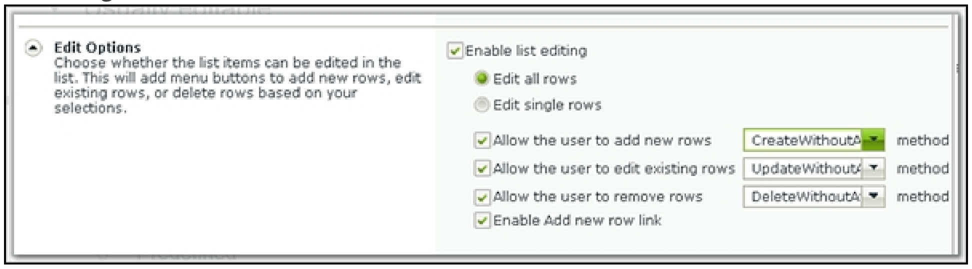
Editable List View

Add | Edit | Delete

Category	Payee	Date	Description	Amount	Currency Code	USD Amount	Receipt
Airfare	Airline Z	12/16/2014	Airline Ticket	300.00	AUD	340.00	K2
Car Rental	Rental Agency B	12/17/2014	Rental Car	400.00	CHF	0.00	Click here to att...
(Add new row)							
Sum						340.00	

To make a List View editable, you only need to set the 'Enable List editing' setting and indicate which methods should be used to Add (Create), Edit (Save/Update) and Delete (Remove) records.

Enabling a List View for Edit mode



Edit Options
Choose whether the list items can be edited in the list. This will add menu buttons to add new rows, edit existing rows, or delete rows based on your selections.

☑ Enable list editing
● Edit all rows
○ Edit single rows
☑ Allow the user to add new rows CreateWithoutA method
☑ Allow the user to edit existing rows UpdateWithoutA method
☑ Allow the user to remove rows DeleteWithoutA method
☑ Enable Add new row link

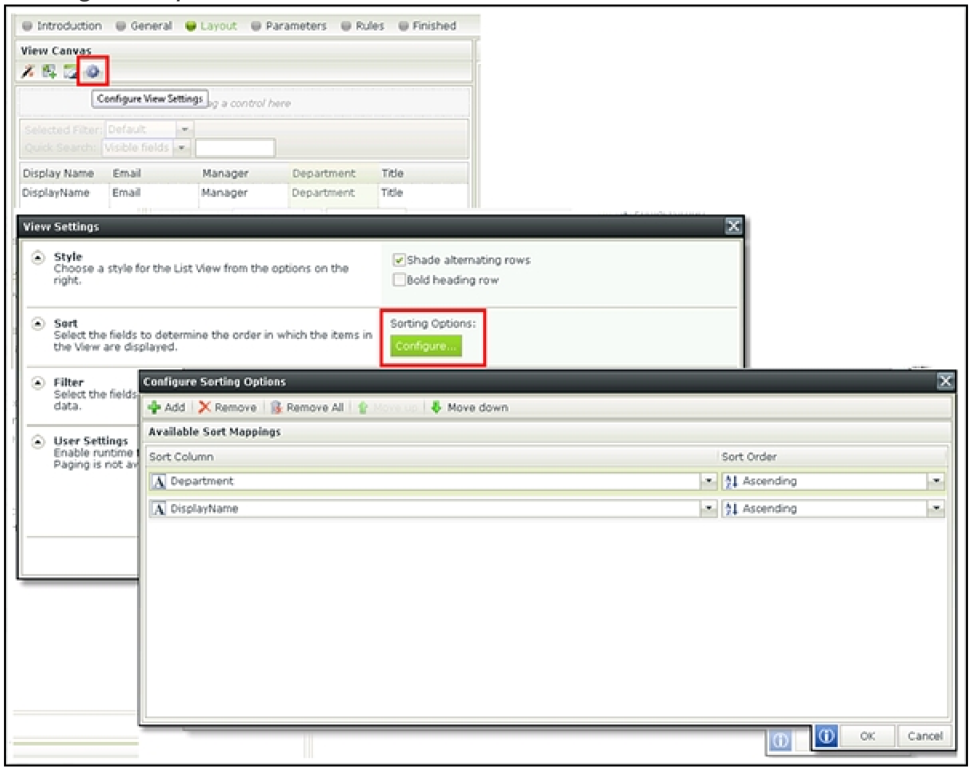
It is possible to search in the list, and paging is supported out of the box. The View designer can adjust these options, along with adding "server-side" Filtering and Sorting options when defining the List View.

Sorting

You can define sort options for a List when configuring the settings for the List View at design time. This will let you specify the sort order for a list at design time and will be the default sort order applied to the list whenever it is refreshed. Think of this as server-side sorting, where the list is sorted before it is presented to the user.

You can sort by multiple columns, in which case the list will be sorted by the first column initially and then by the subsequent columns.

Setting Sort options for a List View



Selected Filter: Default
Quick Search: Visible fields

Display Name	Email	Manager	Department	Title
DisplayName	Email	Manager	Department	Title

View Settings

Style
Choose a style for the List View from the options on the right.
✓ Shade alternating rows
☐ Bold heading row

Sort
Select the fields to determine the order in which the items in the View are displayed.
Sorting Options: Configure...

Filter
Select the fields
data.

User Settings
Enable runtime
Paging is not av

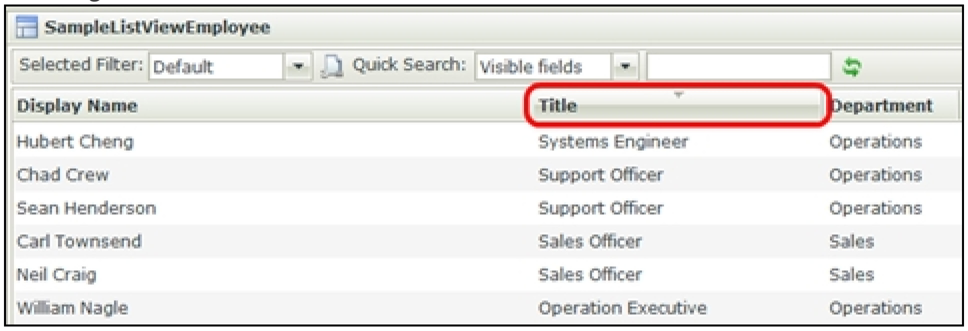
Configure Sorting Options

Available Sort Mappings

Sort Column	Sort Order
Department	Ascending
DisplayName	Ascending

Note that users can also sort lists at runtime by clicking on a column header. Clicking the column header again will reverse the previous sort order.

Clicking a column header at runtime will sort the list



SampleListViewEmployee

Selected Filter: Default   Quick Search: Visible fields

Display Name	Title	Department
Hubert Cheng	Systems Engineer	Operations
Chad Crew	Support Officer	Operations
Sean Henderson	Support Officer	Operations
Carl Townsend	Sales Officer	Sales
Neil Craig	Sales Officer	Sales
William Nagle	Operation Executive	Operations

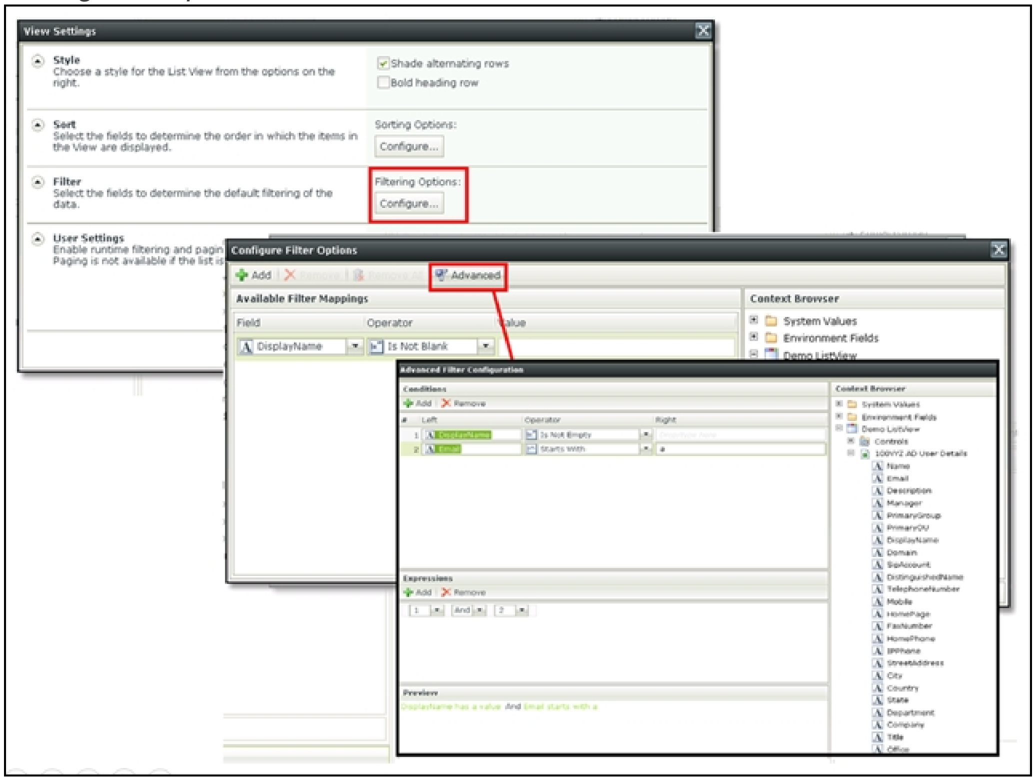
Filtering

There are two approaches you can use to filter List Views: the data can either be filtered before it is presented to the user (also known as “server-side” filtering) or filtered on-demand by the user (also known as “client-side” filtering). The primary difference between these approaches is that when the data is filtered on the server side, the data that has been filtered (out) will never be sent to the user and they will never see the data. If the data is filtered by the user, however, they are actually filtering a local copy of the whole data set, and all of the data is still available on the client side.

To filter the data BEFORE it is presented to the user, you would set the Filter setting for the List View to some statement when editing the View. You can define multiple Filter statements and you can drag in (add) contextual information into the filter. In the example below, we are filtering the list so that the current user will only see employees managed by their own manager (effectively, they are seeing a list of their peers). Because the data is being filtered on the server side, the user will never even see data for other users.

You can also add more complex filter statements with the **Advanced** button.

Setting Filter options for a List View


View Settings

Style — Choose a style for the List View from the options on the right. ☑ Shade alternating rows ☐ Bold heading row

Sort — Select the fields to determine the order in which the items in the View are displayed. Sorting Options: Configure...

Filter — Select the fields to determine the default filtering of the data. Filtering Options: Configure...

Configure Filter Options

Available Filter Mappings

Field	Operator	Value
DisplayName	Is Not Blank	

Advanced Filter Configuration

Conditions

#	Left	Operator	Right
1	DisplayName	Is Not Empty	
2	Email	Starts With	a

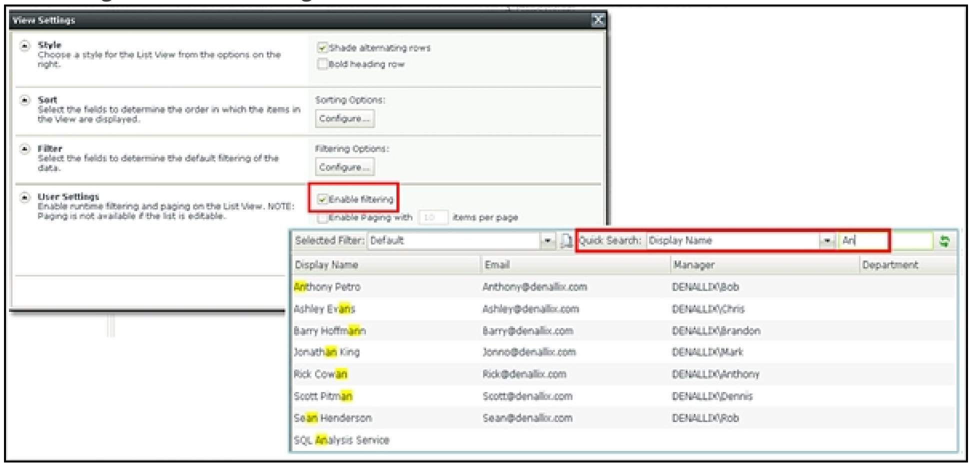
Expressions: 1 And 2

Preview: DisplayName has a value And Email starts with a

Context Browser: System Values, Environment Fields, Demo ListView, Controls, 100VYZ AD User Details: Name, Email, Description, Manager, PrimaryGroup, PrimaryOU, DisplayName, Domain, SipAccount, DistinguishedName, TelephoneNumber, Mobile, HomePage, FaxNumber, HomePhone, IPPhone, StreetAddress, City, Country, State, Department, Company, Title, Office

End users can filter lists on-demand by typing values into the filter statement, as shown below. Remember that the user will only be able to view the data that was passed to them by the server. If the List View has already filtered out some data and sent a reduced data set to the user, the user can only “client-side” filter the reduced data set. Note that you can enable or disable the ability for end users to filter a List View.

Enabling runtime filtering for a List View


View Settings

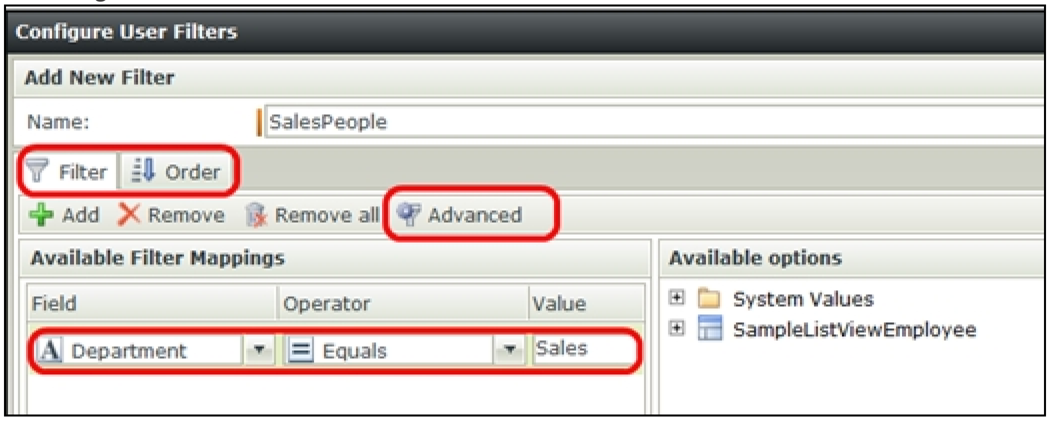
User Settings — Enable runtime filtering and paging on the List View. NOTE: Paging is not available if the list is editable. ☑ Enable filtering ☐ Enable Paging with 10 items per page

Selected Filter: Default Quick Search: Display Name An

Display Name	Email	Manager	Department
Anthony Petro	Anthony@denallix.com	DENALLIX\Bob	
Ashley Evans	Ashley@denallix.com	DENALLIX\Chris	
Barry Hoffmann	Barry@denallix.com	DENALLIX\Brandon	
Jonathan King	Jonno@denallix.com	DENALLIX\Mark	
Rick Cowan	Rick@denallix.com	DENALLIX\Anthony	
Scott Pitman	Scott@denallix.com	DENALLIX\Dennis	
Sean Henderson	Sean@denallix.com	DENALLIX\Rob	
SQL Analysis Service			

Users can also save filters as pre-defined, per-user filters so that they can re-use filters in future. When clicking on the configuration icon for the Selected Filter, they can add simple or advanced filter statements and save the filter definition (they can also add sorting to the new filter statement). Once they have created filters, those filters are available for future use on the same list.

Defining a new client-side filter



Configure User Filters

Add New Filter

Name: SalesPeople

Filter | Order

Add | Remove | Remove all | Advanced

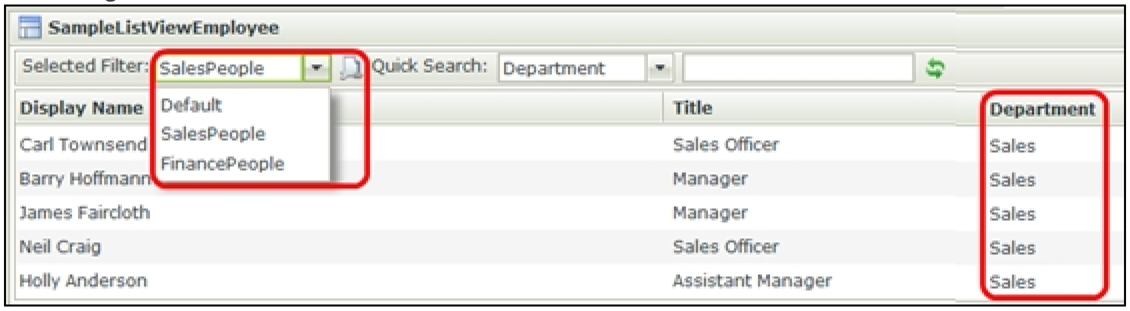
Available Filter Mappings

Field	Operator	Value
Department	Equals	Sales

Available options

- System Values
- SampleListViewEmployee

Re-using client-side filters



SampleListViewEmployee

Selected Filter: SalesPeople (Default, SalesPeople, FinancePeople) Quick Search: Department

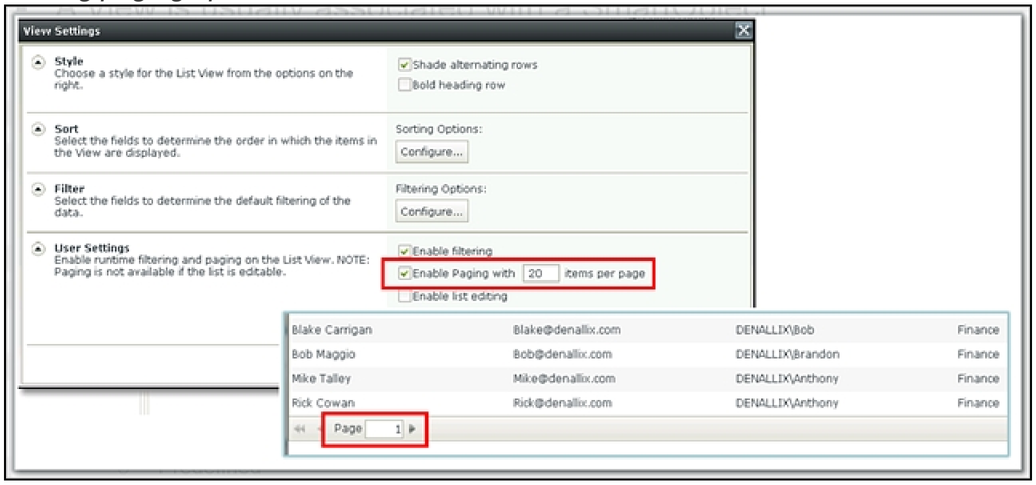
Display Name	Title	Department
Carl Townsend	Sales Officer	Sales
Barry Hoffmann	Manager	Sales
James Faircloth	Manager	Sales
Neil Craig	Sales Officer	Sales
Holly Anderson	Assistant Manager	Sales

Paging

List Views can also use paging so that users are not overloaded with long lists of data. Note that any filtering the user applies is applied to the WHOLE list, not just the items on the page. (Client-side sorting is based on the items on the current page.)

Note
Paging is not available when using an editable list.

Setting paging options for a List View



View Settings

Style
Choose a style for the List View from the options on the right.
- [✓] Shade alternating rows
- [ ] Bold heading row

Sort
Select the fields to determine the order in which the items in the View are displayed.
Sorting Options: Configure...

Filter
Select the fields to determine the default filtering of the data.
Filtering Options: Configure...

User Settings
Enable runtime filtering and paging on the List View. NOTE: Paging is not available if the list is editable.
- [✓] Enable filtering
- [✓] Enable Paging with 20 items per page
- [ ] Enable list editing

Blake Carrigan	Blake@denallix.com	DENALLIX\Bob	Finance
Bob Maggio	Bob@denallix.com	DENALLIX\Brandon	Finance
Mike Talley	Mike@denallix.com	DENALLIX\Anthony	Finance
Rick Cowan	Rick@denallix.com	DENALLIX\Anthony	Finance

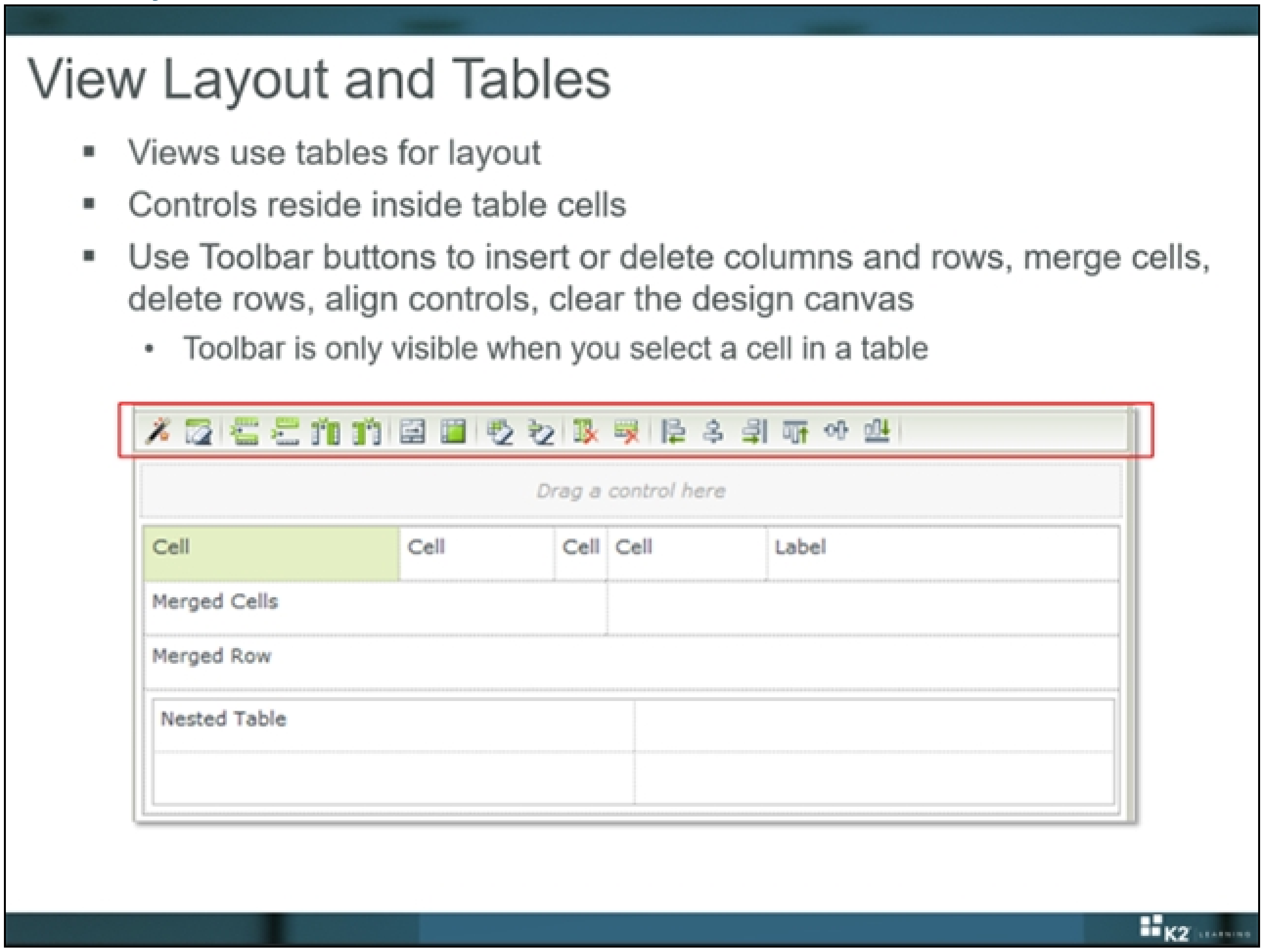
Page 1

Summary

- Item Views: single record, usually associated with Create/Read/Update/Delete methods.
  - Usually are editable
  - Can disable the entire View or specific Controls
    - Enabled/Disabled for View can be set with Rule Actions
    - Enabled/Disabled for Controls can be set read-only at design time or at runtime with Rule Actions

- List Views: multiple records, usually associated with List methods for read-only List Views
    - Editable List Views allow users to capture multiple records at the same time, it is almost like an Excel-style data entry.
    - Sorting and Filtering can be defined when designing the View or can be enabled for end users to do their own filtering.
    - Paging is defined at design time. This can help to improve performance since it can be inefficient to list thousands of items on the same List View all at the same time.

View Layout and Tables



View Layout and Tables

- Views use tables for layout
- Controls reside inside table cells
- Use Toolbar buttons to insert or delete columns and rows, merge cells, delete rows, align controls, clear the design canvas
  - Toolbar is only visible when you select a cell in a table

Views use tables to format the layout of Controls on the View. The most common approach used to format the layout of Views is to adjust the layout of the table on the View, and then drag and drop Controls into the table where appropriate. (Controls are always located within a cell of the layout table.)

When you select a table cell, column or row, a toolbar will appear that allows you to modify the table layout. You may use the Layout Toolbar to make changes if necessary to the table layout. You can add and remove columns and rows, merge cells or align Controls in the table (e.g. aligning a Control to the middle of a column). If needed, you can also nest tables inside other tables. It is also possible to use the table, column and cell Properties pane to set the dimensions of those items.

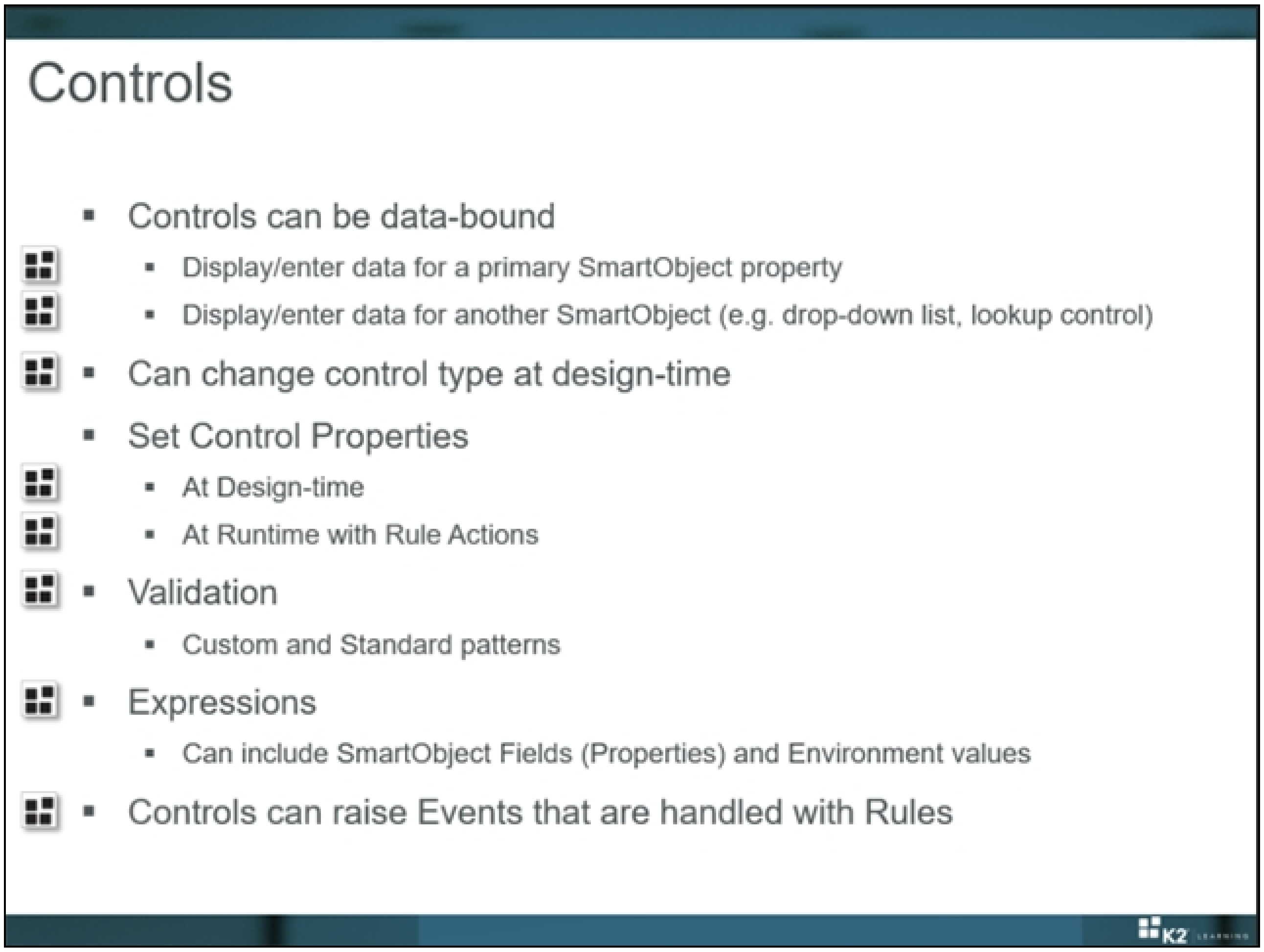
If you are familiar with designing forms using Microsoft InfoPath, the approach is very similar in using tables, columns and views to format the layout of your form, and you have the same ability to merge cells, add columns and align the Controls inside the table cells.

If you are familiar with web development, tables are almost the same as HTML <TABLE> or <DIV> elements.

Summary

- Views use tables with rows, columns and cells to format the layout of Controls on the View
- Controls live inside of the cells of the layout table
- You can use the Toolbar buttons to merge/insert/delete rows and cells in a table
- The toolbar will only appear when you select a cell in a table
- You can have nested tables and use cell/table properties to set the width/height of cells

Controls



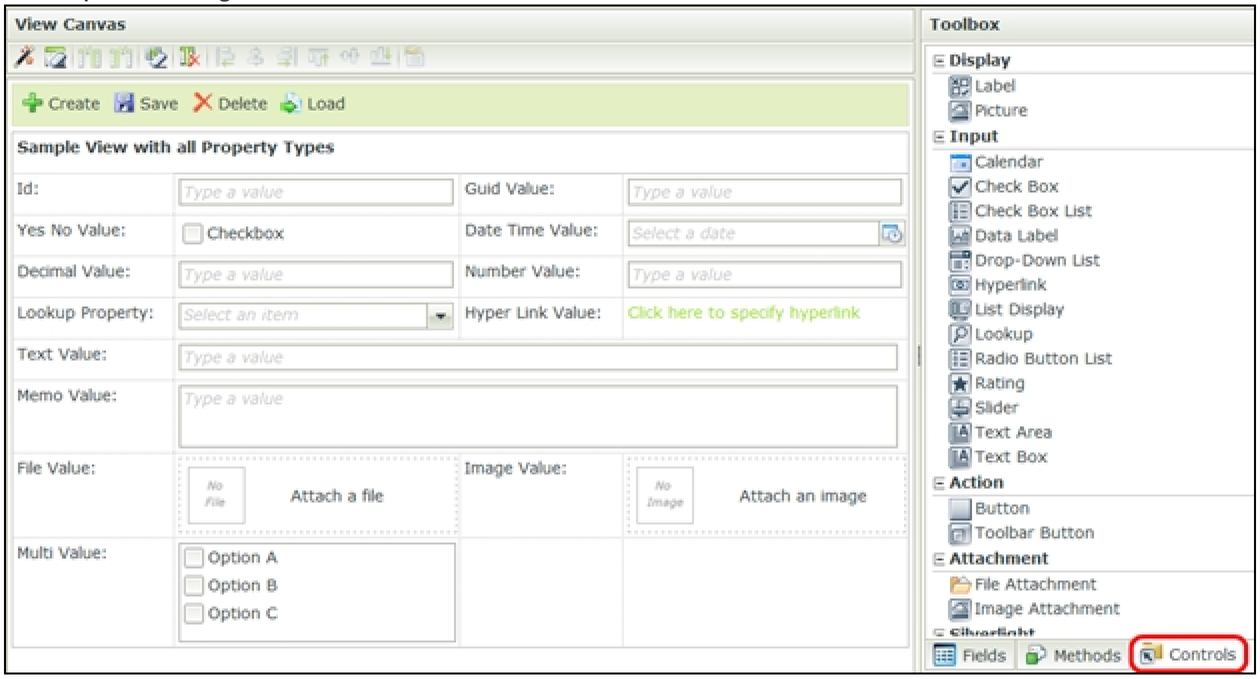
Controls

- Controls can be data-bound
  - Display/enter data for a primary SmartObject property
  - Display/enter data for another SmartObject (e.g. drop-down list, lookup control)
- Can change control type at design-time
- Set Control Properties
  - At Design-time
  - At Runtime with Rule Actions
- Validation
  - Custom and Standard patterns
- Expressions
  - Can include SmartObject Fields (Properties) and Environment values
- Controls can raise Events that are handled with Rules

Video

Controls are used on Views and Forms to display data or values and to capture user input. There are a range of controls available in the Toolbox on the right-hand side of the screen: the screenshot below illustrates a selection of the available Controls.

A sample showing some of the available Controls



View Canvas

Create Save Delete Load

Sample View with all Property Types

Id:	Type a value	Guid Value:	Type a value
Yes No Value:	Checkbox	Date Time Value:	Select a date
Decimal Value:	Type a value	Number Value:	Type a value
Lookup Property:	Select an item	Hyper Link Value:	Click here to specify hyperlink
Text Value:	Type a value		
Memo Value:	Type a value		
File Value:	No File Attach a file	Image Value:	No Image Attach an image
Multi Value:	Option A Option B Option C		

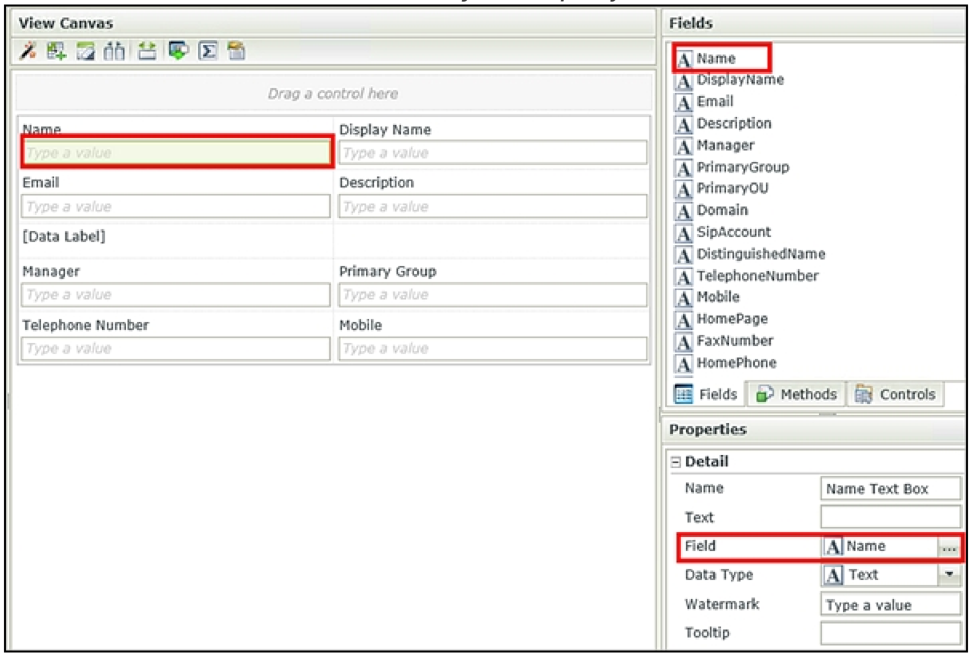
Toolbox

- Display
  - Label
  - Picture
- Input
  - Calendar
  - Check Box
  - Check Box List
  - Data Label
  - Drop-Down List
  - Hyperlink
  - List Display
  - Lookup
  - Radio Button List
  - Rating
  - Slider
  - Text Area
  - Text Box
- Action
  - Button
  - Toolbar Button
- Attachment
  - File Attachment
  - Image Attachment
- Silverlight

Fields Methods Controls

Controls are usually bound to a SmartObject property, and can also be used to display data from another SmartObject (for example using a drop-down list control to show a list of values returned by a SmartObject).

A Control that is bound to a SmartObject Property



View Canvas

Drag a control here

Name	Display Name
Type a value	Type a value
Email	Description
Type a value	Type a value
[Data Label]	
Manager	Primary Group
Type a value	Type a value
Telephone Number	Mobile
Type a value	Type a value

Fields

- Name
- DisplayName
- Email
- Description
- Manager
- PrimaryGroup
- PrimaryOU
- Domain
- SipAccount
- DistinguishedName
- TelephoneNumber
- Mobile
- HomePage
- FaxNumber
- HomePhone

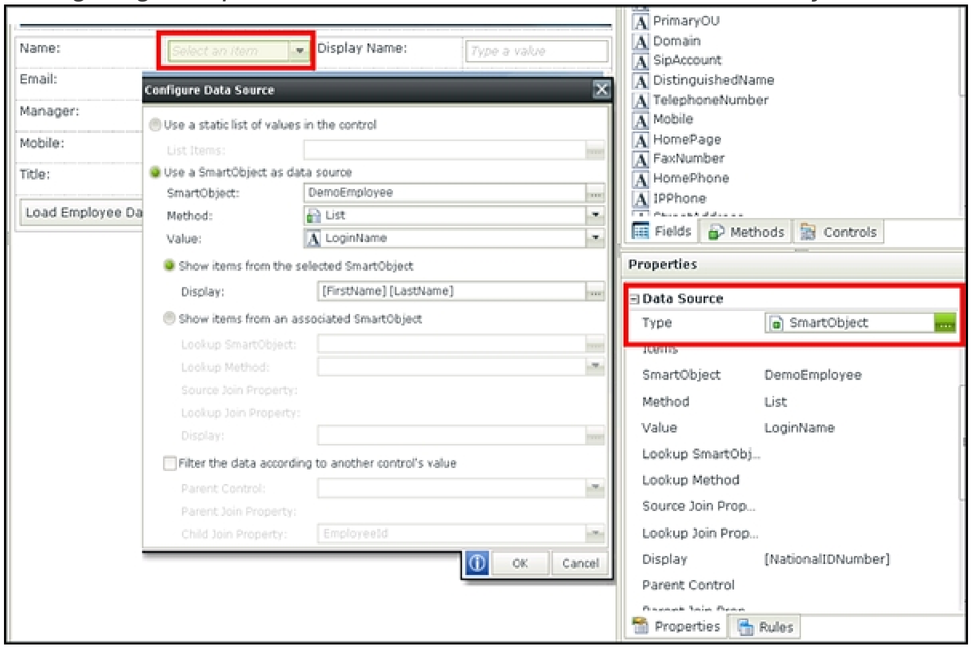
Fields | Methods | Controls

Properties

Detail

Name	Name Text Box
Text	
Field	Name
Data Type	Text
Watermark	Type a value
Tooltip	

Configuring a drop-down list to show data from another SmartObject



Name: Select an item Display Name: Type a value

Email:

Manager:

Mobile:

Title:

Load Employee Da

Configure Data Source

- Use a static list of values in the control
  - List Items:
- Use a SmartObject as data source
  - SmartObject: DemoEmployee
  - Method: List
  - Value: LoginName
  - Show items from the selected SmartObject
    - Display: [FirstName] [LastName]
  - Show items from an associated SmartObject
    - Lookup SmartObject:
    - Lookup Method:
    - Source Join Property:
    - Lookup Join Property:
    - Display:
- Filter the data according to another control's value
  - Parent Control:
  - Parent Join Property:
  - Child Join Property: EmployeeId

OK Cancel

- PrimaryOU
- Domain
- SipAccount
- DistinguishedName
- TelephoneNumber
- Mobile
- HomePage
- FaxNumber
- HomePhone
- IPPhone

Fields | Methods | Controls

Properties

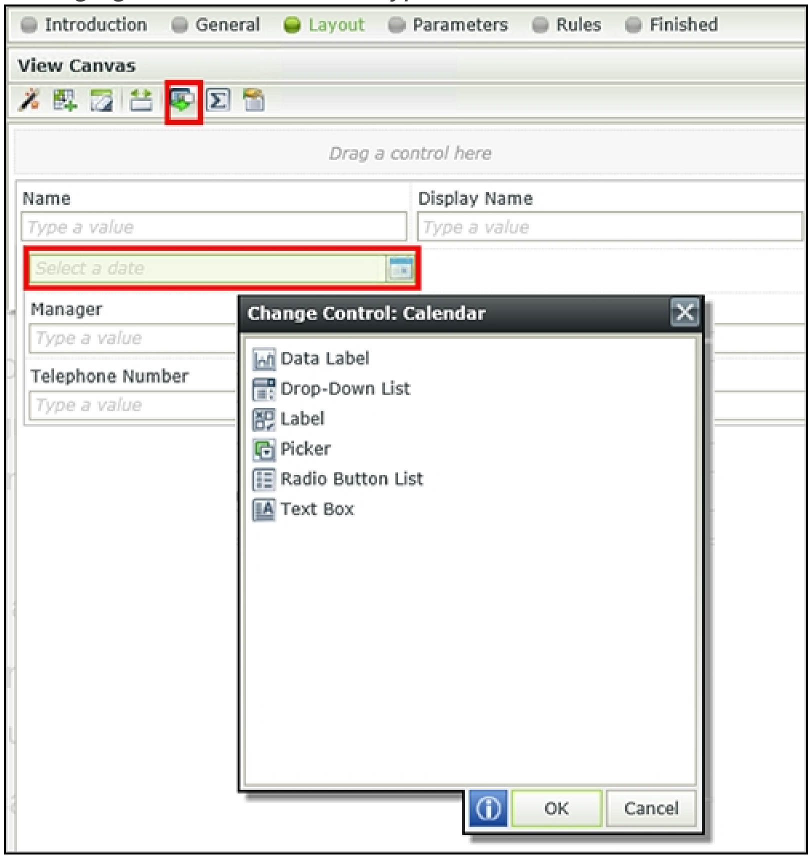
Data Source

Type	SmartObject
Items	
SmartObject	DemoEmployee
Method	List
Value	LoginName
Lookup SmartObj...	
Lookup Method	
Source Join Prop...	
Lookup Join Prop...	
Display	[NationalIDNumber]
Parent Control	

Properties | Rules

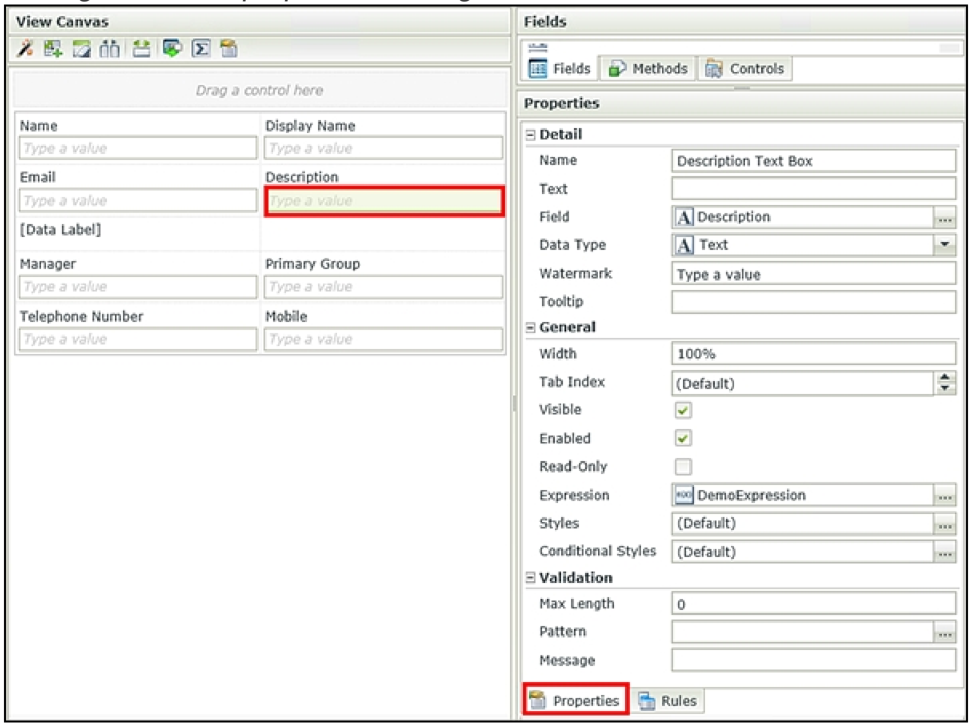
When you generate a View or drag and drop fields from a SmartObject onto a View or Form design canvas, K2 will automatically assign an appropriate type of control for the data type of the field. For example, if you drag a date/time field onto the design canvas, K2 will insert a calendar control onto the design canvas because this is the most common control used to input a date/time field. However, it is possible to change the type of a control at design time using the toolbar, as shown below:

Changing a control to another type of control



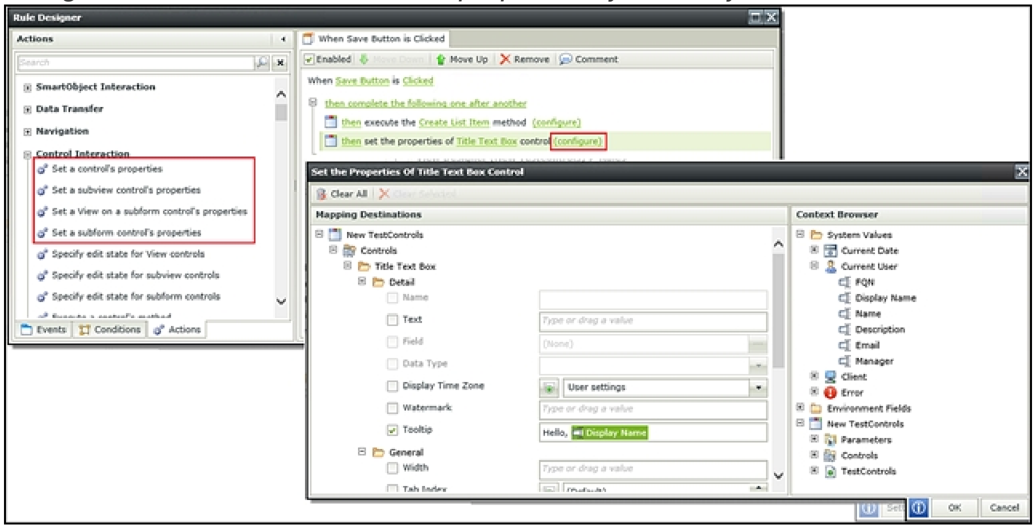
You can set Control Properties at design-time by editing the properties of the control in the Properties pane, or you can set the properties of controls at runtime with Rules. For example, you may want to define a Rule that sets a certain control to read-only if a particular value is selected from a drop-down list.

Setting a Control's properties at design-time



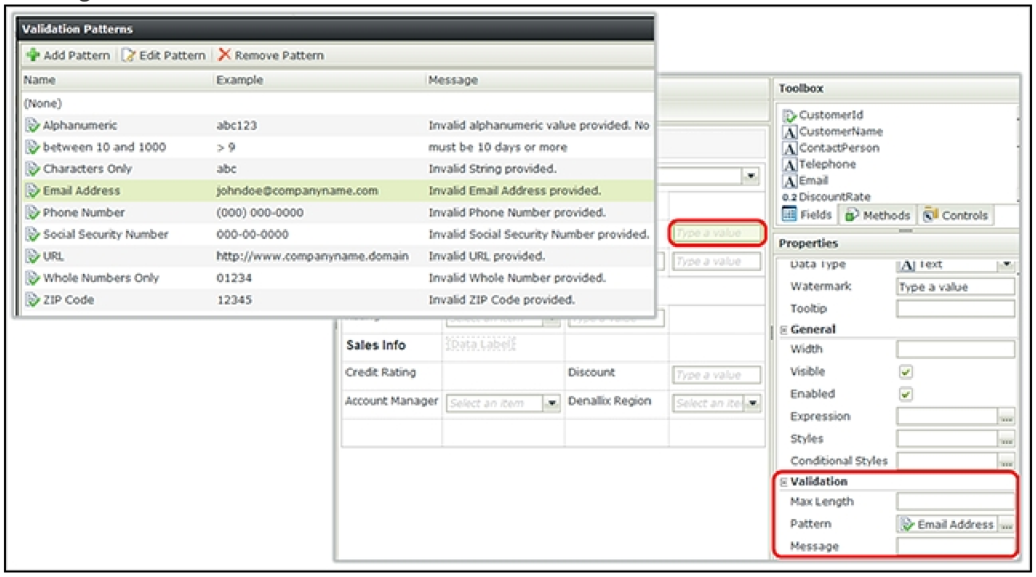
Properties	
Detail	
Name	Description Text Box
Text	
Field	Description
Data Type	Text
Watermark	Type a value
Tooltip	
General	
Width	100%
Tab Index	(Default)
Visible	✓
Enabled	✓
Read-Only	
Expression	DemoExpression
Styles	(Default)
Conditional Styles	(Default)
Validation	
Max Length	0
Pattern	
Message	

Using Rule Actions to set a control's properties dynamically at runtime


When Save Button is Clicked
then complete the following one after another
then execute the Create List Item method (configure)
then set the properties of Title Text Box control (configure)

You can set Validation Properties on Controls that can match the value entered in the control with a particular pattern or format. For example, you can select the **Email Address** pattern to verify that the value entered by the user is in the correct format for an email address. You can also define the maximum length allowed for input as well as a custom message if the validation is not successful. You can define custom patterns for other types of validation expressions as well.

Adding validation to a Control

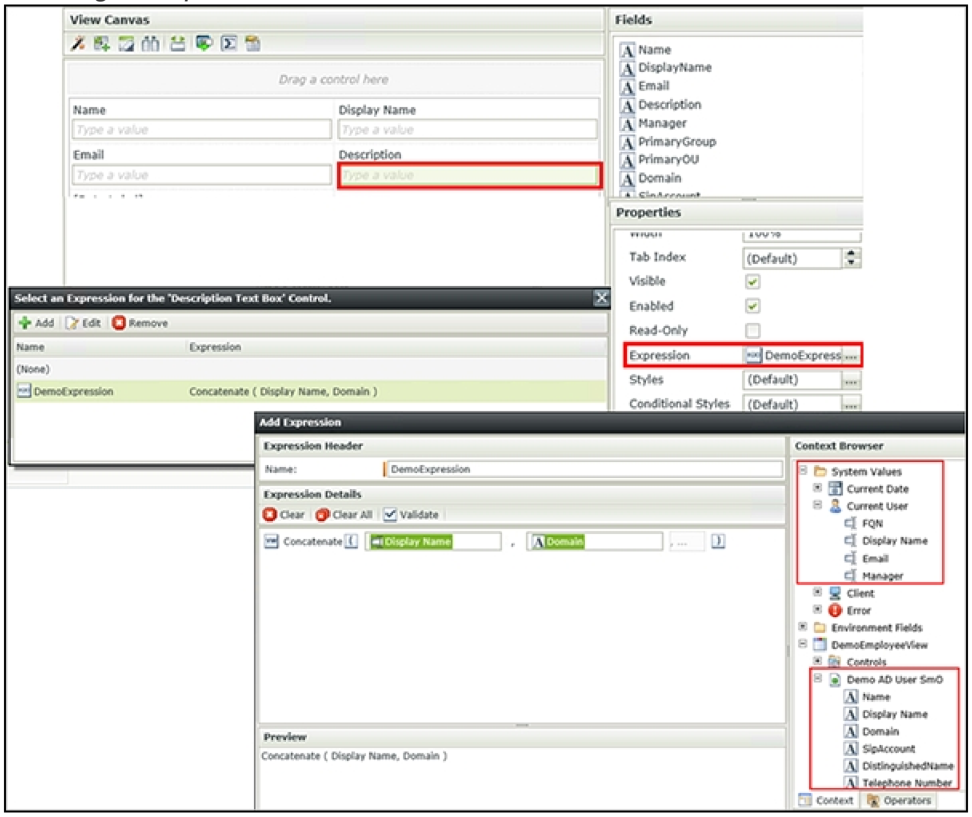

Validation Patterns

Name	Example	Message
(None)		
Alphanumeric	abc123	Invalid alphanumeric value provided. No
between 10 and 1000	> 9	must be 10 days or more
Characters Only	abc	Invalid String provided.
Email Address	johndoe@companyname.com	Invalid Email Address provided.
Phone Number	(000) 000-0000	Invalid Phone Number provided.
Social Security Number	000-00-0000	Invalid Social Security Number provided.
URL	http://www.companyname.domain	Invalid URL provided.
Whole Numbers Only	01234	Invalid Whole Number provided.
ZIP Code	12345	Invalid ZIP Code provided.

Validation
Max Length
Pattern: Email Address
Message

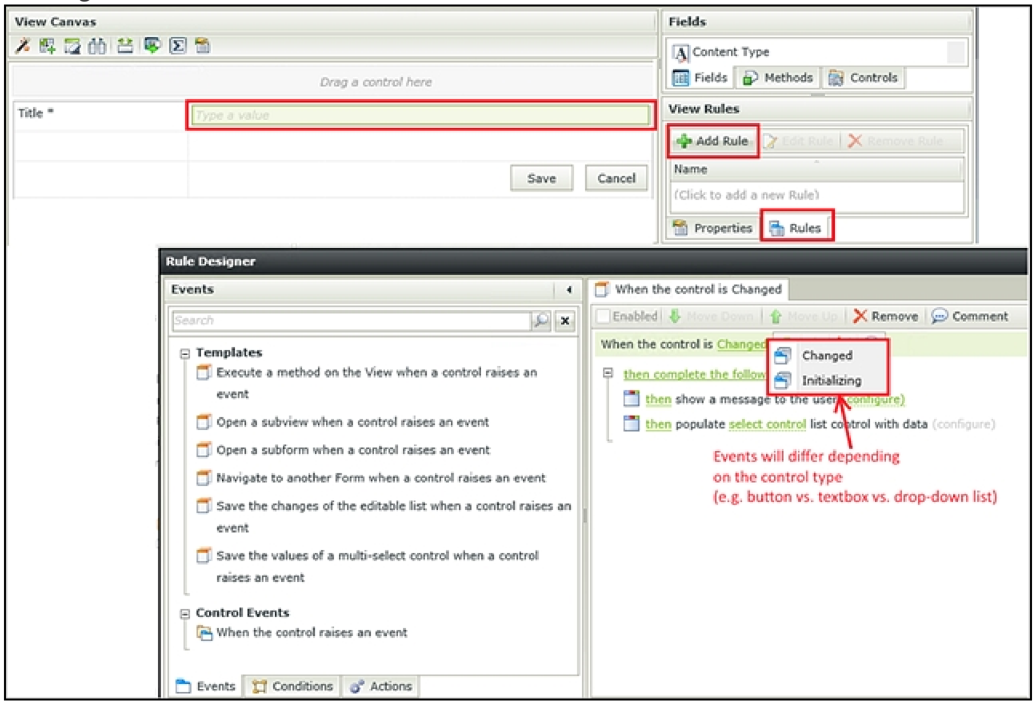
Expressions can be used to calculate a value for a Control. For example, you may want to concatenate the values of two fields together and display them in one Control. To achieve this, use the **Expression** Property and define an expression using the available **Operators** in the Context Browser. You can also include values from the current environment (such as the current username or current date) that are available in the System Values group in the Context Browser, as shown below.

Defining an expression for a Control



Controls can raise Events that are handled with Rules. This allows you to build some processing logic into a View when something happens on a control (for example, refreshing the View content if a specific control is changed by the user). Different types of controls will have different events.

Adding a Rule for a control event

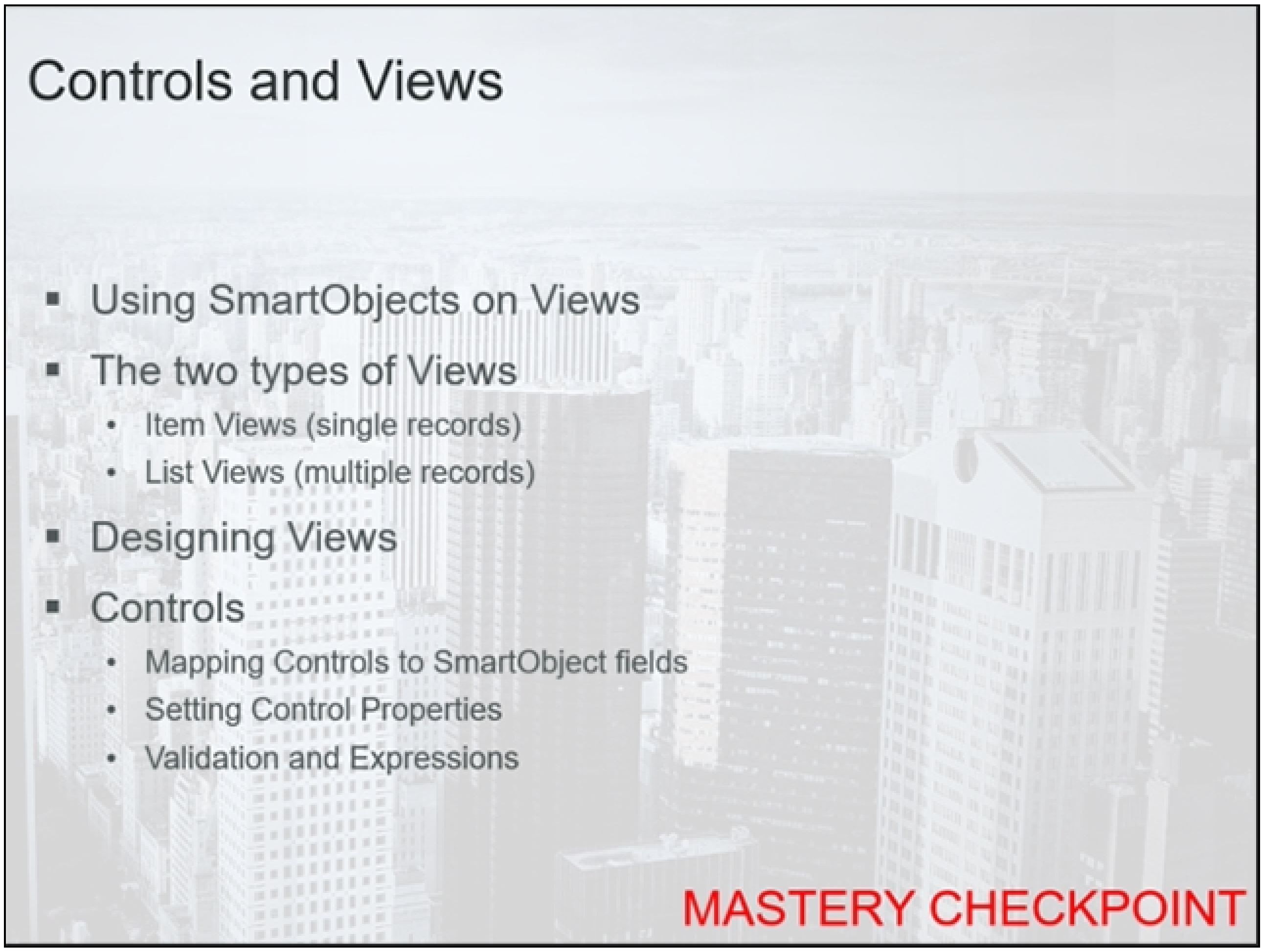

View Canvas
Drag a control here
Title *
Type a value
Save
Cancel
Fields
Content Type
Fields
Methods
Controls
View Rules
Add Rule
Edit Rule
Remove Rule
Name
(Click to add a new Rule)
Properties
Rules
Rule Designer
Events
Search
Templates
Execute a method on the View when a control raises an event
Open a subview when a control raises an event
Open a subform when a control raises an event
Navigate to another Form when a control raises an event
Save the changes of the editable list when a control raises an event
Save the values of a multi-select control when a control raises an event
Control Events
When the control raises an event
Events
Conditions
Actions
When the control is Changed
Enabled
Move Down
Move Up
Remove
Comment
When the control is Changed
Changed
Initializing
then complete the following
then show a message to the user (configure)
then populate select control list control with data (configure)
Events will differ depending on the control type (e.g. button vs. textbox vs. drop-down list)

Summary

- Controls are often bound to SmartObject Properties, but they do not have to be data-bound.
- Control types can be changed (within limits). Select a Control and use the Toolbar to change the Control to a different type (e.g. changing a text-box to a label).

- Control Properties can be set at design-time or at runtime with Rule Actions.
- Validation is based on regular expressions and you can use standard and custom patterns.
- Expressions can be used to perform manipulation or other functions on Controls.
- Controls can raise Events that you can handle with Rules (for example, when a button is clicked - do something, or when a field changes - do something).

MASTERY CHECKPOINT: Controls and Views



Controls and Views

- Using SmartObjects on Views
- The two types of Views
  - Item Views (single records)
  - List Views (multiple records)
- Designing Views
- Controls
  - Mapping Controls to SmartObject fields
  - Setting Control Properties
  - Validation and Expressions

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 2 of this module: Views and Controls. If you are attending instructor-led training, use this as an opportunity to answer any questions around Views and Controls.

By now you should know:

- How to build Views that use SmartObjects as the data source
- How to design Views using the layout pane (design canvas)
- The two types of Views (List Views and Item Views)
- How to work with Controls
  - How to map Controls to SmartObject Properties
  - Setting Control Properties at design-time
  - Working with Control Validation and Expressions
  - Working with different types of Controls

Knowledge-check questions

Q: What is the main difference between List Views and Item Views?
Reveal answer**A:**Item Views work with a single record, List Views work with multiple records.

Q: Can you change a Control to a different type? (e.g. changing a data label to a text box)?
Reveal answer**A:**Yes, you can change Controls to other types (within some limits)

Q: What is the difference between a data-bound Control and a non-bound Control?
Reveal answer**A:** A data-bound Control is associated with a particular property of a SmartObject. A non-bound Control is not.

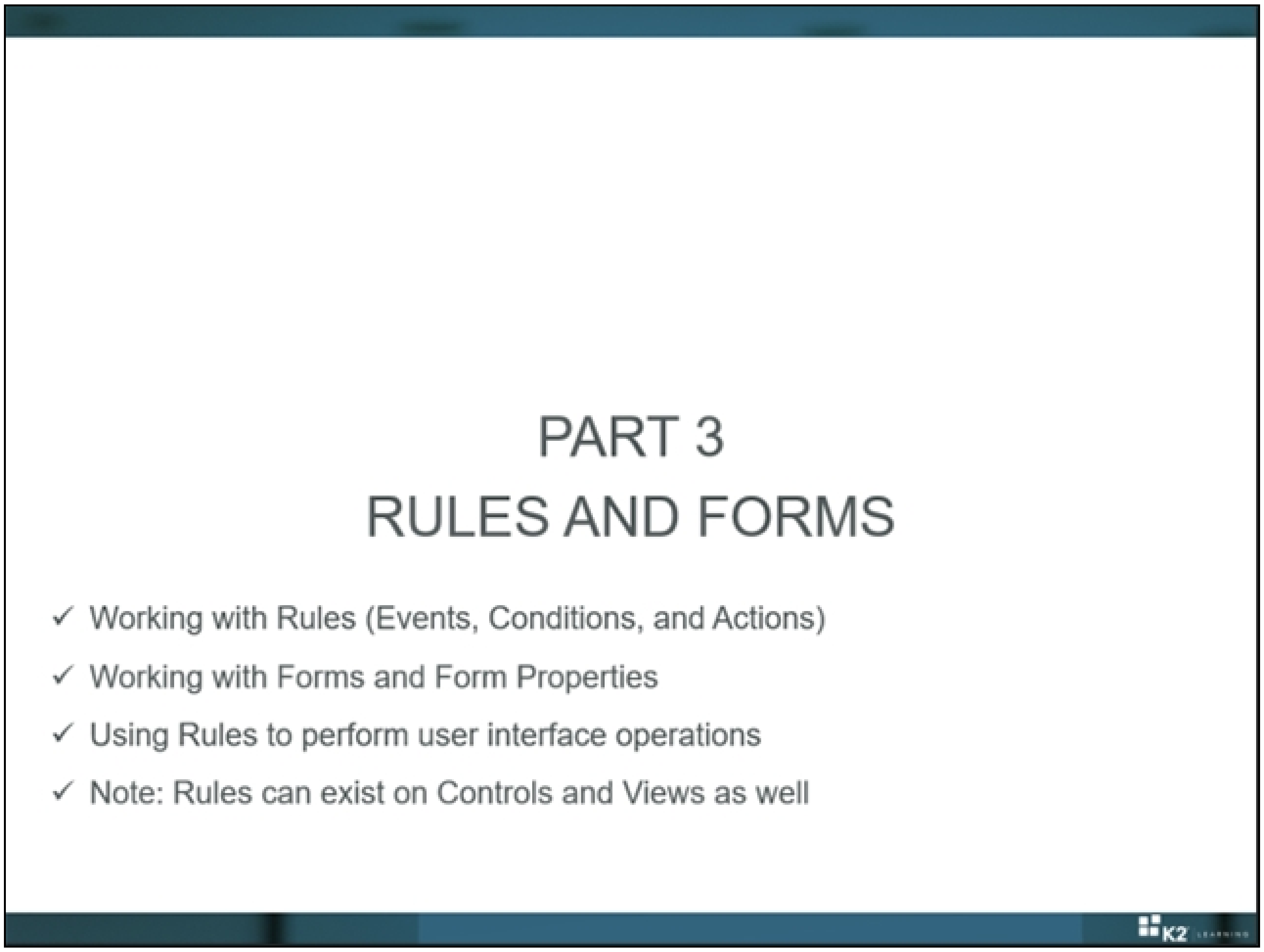
Q: Can you put a button Control on a View?
Reveal answer**A:**Yes, and you can put buttons on Forms as well.

Q:Can Controls exist on Forms, or only on Views?

Reveal answer**A:**Controls can exist on Forms as well as Views.

Q:What is the difference between a Validation and an Expression?
Reveal answer**A:** A Validation tests whether the data entered by the user is acceptable, such as a valid e-mail address. Expressions are used to set the value of a Control, such as concatenating text together in a text box.

PART 3: Rules and Forms



PART 3

RULES AND FORMS

- ✓ Working with Rules (Events, Conditions, and Actions)
- ✓ Working with Forms and Form Properties
- ✓ Using Rules to perform user interface operations
- ✓ Note: Rules can exist on Controls and Views as well

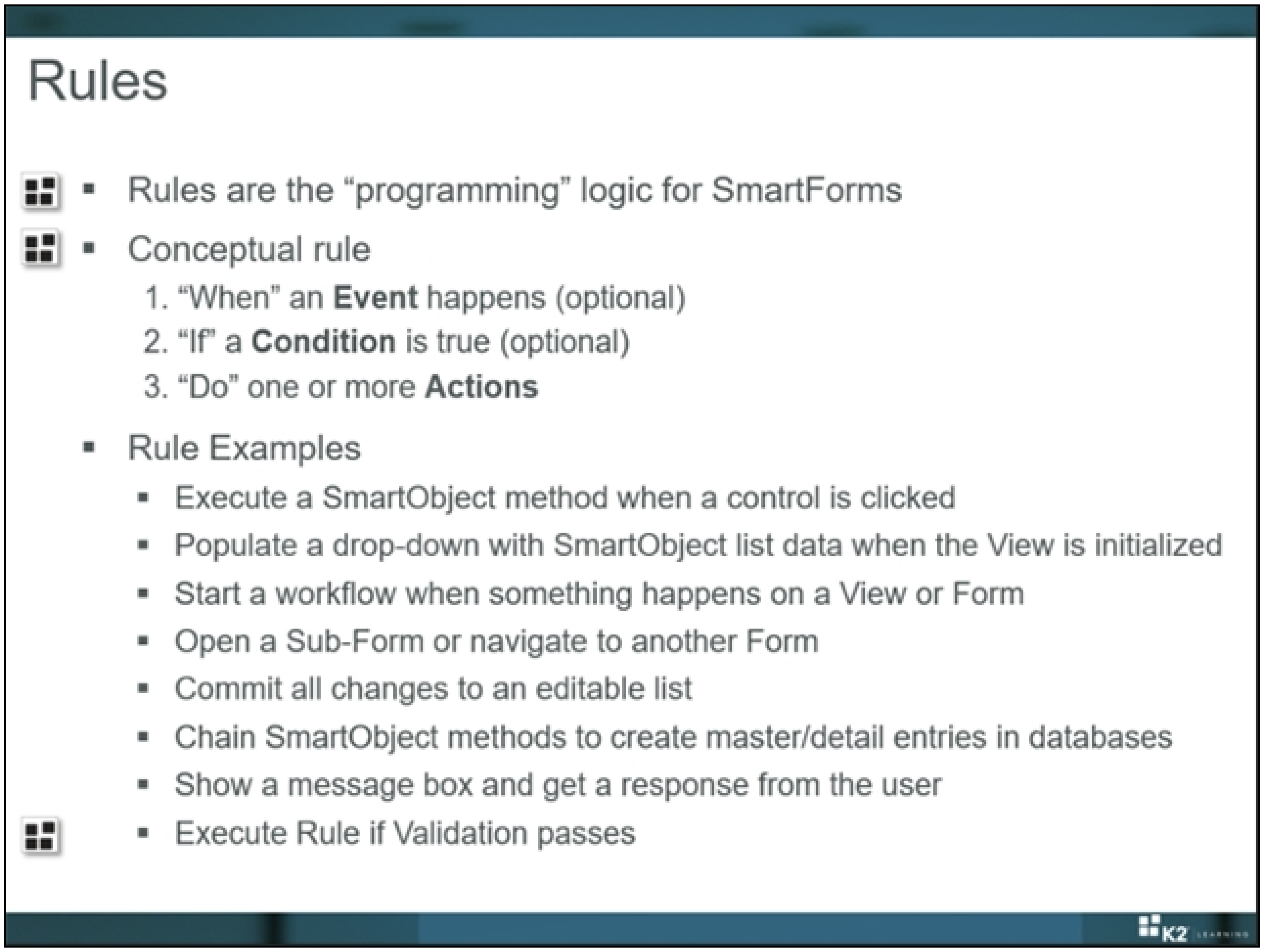
Video

In Part 3 we will look at Rules and Forms in more detail.

- We will describe the components of Rules (Events, Conditions and Actions)
- We will explain how to work with Forms and Form Properties such as themes and parameters
- At the end of Part 3, we will build two Forms
  - One Form will host the two Views we created in Part 2 (this Form is used to list details for all AD Users)
  - The other Form will be used for the Approval task in the Workflow (and will only use the Item View to show the details for a single record)

Note
Rules can exist on Forms, Views and Controls, too. We are only grouping Rules and Forms together in Part 3 because of the way the exercises are structured.

Rules



Rules

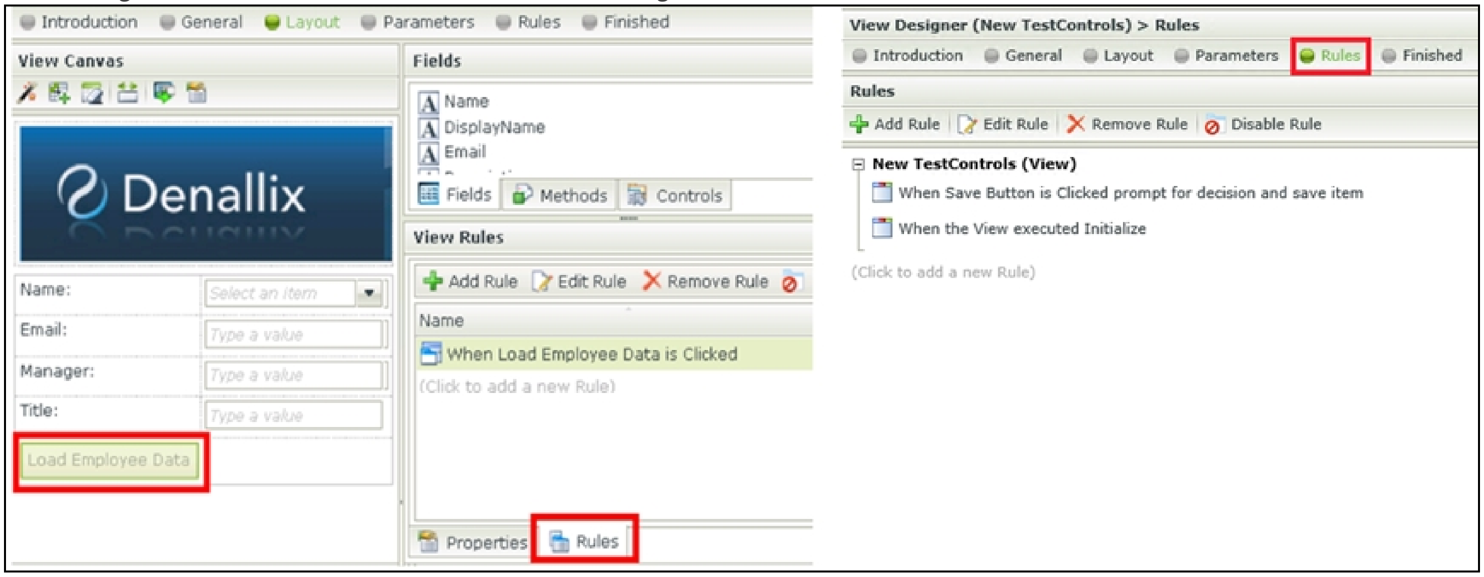
- Rules are the "programming" logic for SmartForms
- Conceptual rule
  1. "When" an **Event** happens (optional)
  2. "If" a **Condition** is true (optional)
  3. "Do" one or more **Actions**
- Rule Examples
  - Execute a SmartObject method when a control is clicked
  - Populate a drop-down with SmartObject list data when the View is initialized
  - Start a workflow when something happens on a View or Form
  - Open a Sub-Form or navigate to another Form
  - Commit all changes to an editable list
  - Chain SmartObject methods to create master/detail entries in databases
  - Show a message box and get a response from the user
  - Execute Rule if Validation passes

Video

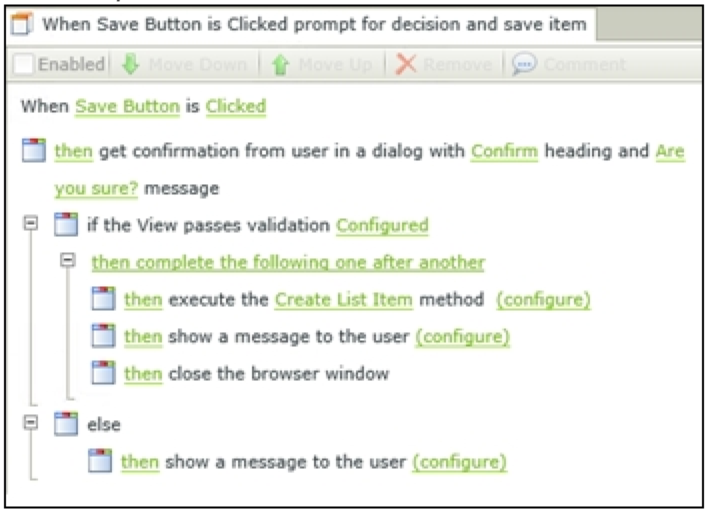
Most user interfaces require some kind of programming or processing support. For example, saving updates made to a record into a database when the user clicks the "Save" button, or starting a Workflow when the user "submits" a Form. SmartForms use Rules to perform these "programming" actions. (We use "programming" in quotes because there is actually no coding required when implementing the processing tasks. Instead, the Form designer will use graphical, wizard-based configuration tools to assemble the programming logic.)

The screenshot below shows how to access the Rules property pane for a Control and the Rules for a View or a Form. A Control, View or Form may have multiple Rules, and most often each Rule is bound to a particular Event (but they don't have to be).

Accessing Control Rules and View Rules in the Designer



An example of a Rule


When Save Button is Clicked prompt for decision and save item

Enabled | Move Down | Move Up | Remove | Comment

When Save Button is Clicked

then get confirmation from user in a dialog with Confirm heading and Are you sure? message

if the View passes validation Configured

then complete the following one after another

then execute the Create List Item method (configure)

then show a message to the user (configure)

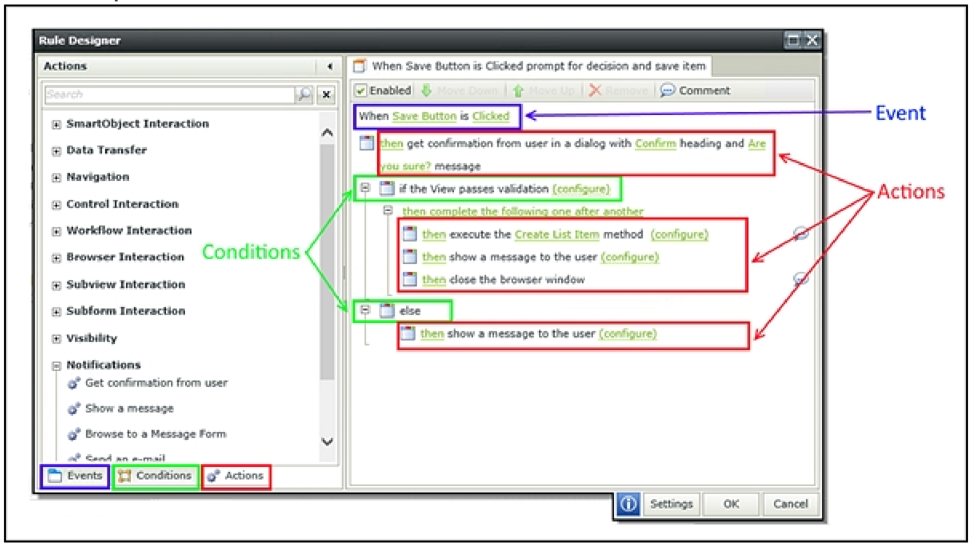
then close the browser window

else

then show a message to the user (configure)

Rules are made up of three components: Events, Conditions and Actions.

The components of a Rule: Event, Conditions and Actions



When Save Button is Clicked prompt for decision and save item

Enabled | Move Down | Move Up | Remove | Comment

When Save Button is Clicked

then get confirmation from user in a dialog with Confirm heading and Are you sure? message

if the View passes validation (configure)

then complete the following one after another

then execute the Create List Item method (configure)

then show a message to the user (configure)

then close the browser window

else

then show a message to the user (configure)

Events

Events define WHEN the Rule should “fire”, and are normally associated with an event on a Control, View or Page. Examples of events include ”when a button is clicked”, “when a control’s value changes” or “when the View is initialized”. In this example, we want the Rule to fire when a control (a Button) on the View raises an event (is clicked). Note that Rules do not have to be bound to Events, and you can define "eventless" Rules. These Rules are normally called by another Rule instead of an Event. Depending on the Control you clicked or the type of View (List View or Item View) being designed, you may have different Events available in the Events list. To help speed up your development time, the most common Rules for the Control you select are also displayed under the Templates heading. These Templates are just pre-defined Rules that you can select from, or you can define Rules from scratch using the available Events in the wizard.

Conditions

Conditions control IF a Rule is allowed to execute. These Conditions are normally some kind of true/false result, and as with the Events designer, you will have a range of pre-defined Conditions to choose from. You can also use the Advanced Condition configuration tool to define a more advanced Condition for the Rule. By default, a Rule is set to execute “always” (in other words, if you do not define a Condition, the Rule will always continue to the Actions part).

In the example Rule above, we defined a Condition so that the Actions would only execute if the Form passes validation.

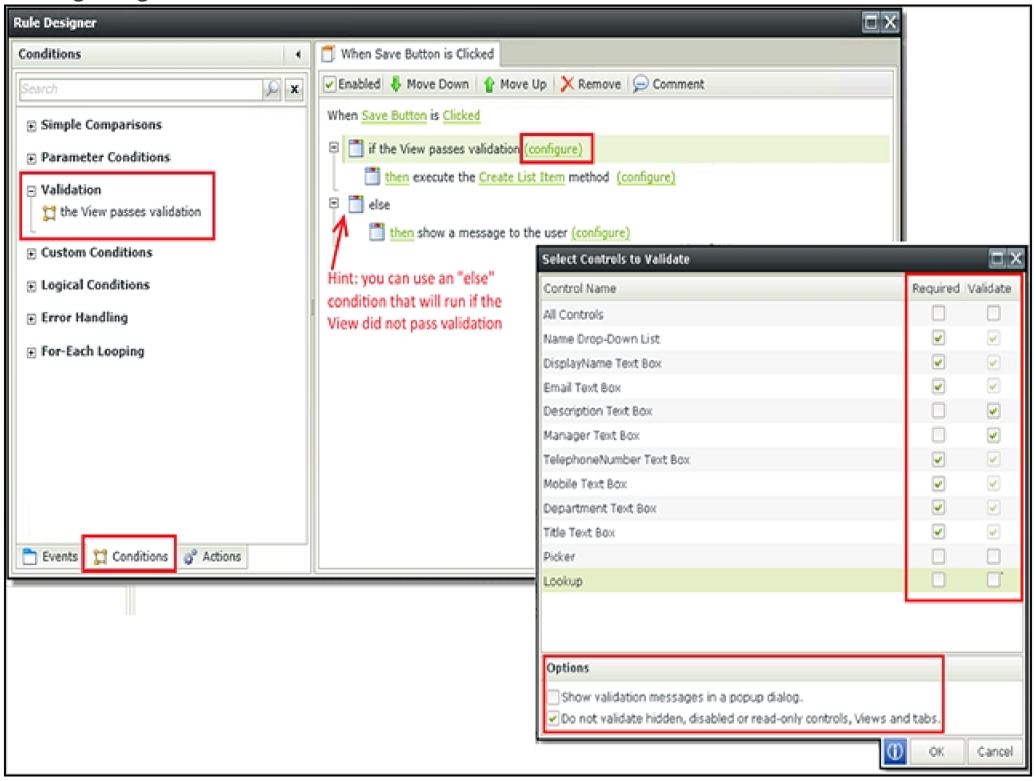
Actions

Actions define WHAT the Rule does, in other words, the work that is performed by the Rule. As with the other Rule components, when defining a Rule you will have a selection of operations to choose from and you can select multiple operations in the same Action. Rules always have to have at least one Action, but can contain multiple Actions.

Configuring Validation

When configuring a "passes Validation" Condition for a Rule, you can select which Controls should be validated, where the validation error should be shown and whether or not to ignore hidden Controls. Additionally, you can add an "else" Condition which will execute if the Form or View did not pass validation. The screenshot below shows how this is implemented.

Configuring a Rule with Validation



When Save Button is Clicked

if the View passes validation (configure)

then execute the Create List Item method (configure)

else

then show a message to the user (configure)

Hint: you can use an "else" condition that will run if the View did not pass validation

Select Controls to Validate

Control Name	Required	Validate
All Controls	☐	☐
Name Drop-Down List	✓	✓
DisplayName Text Box	✓	✓
Email Text Box	✓	✓
Description Text Box	☐	✓
Manager Text Box	☐	✓
TelephoneNumber Text Box	✓	✓
Mobile Text Box	✓	✓
Department Text Box	✓	✓
Title Text Box	✓	✓
Picker	☐	☐
Lookup	☐	☐

Options

☐ Show validation messages in a popup dialog.

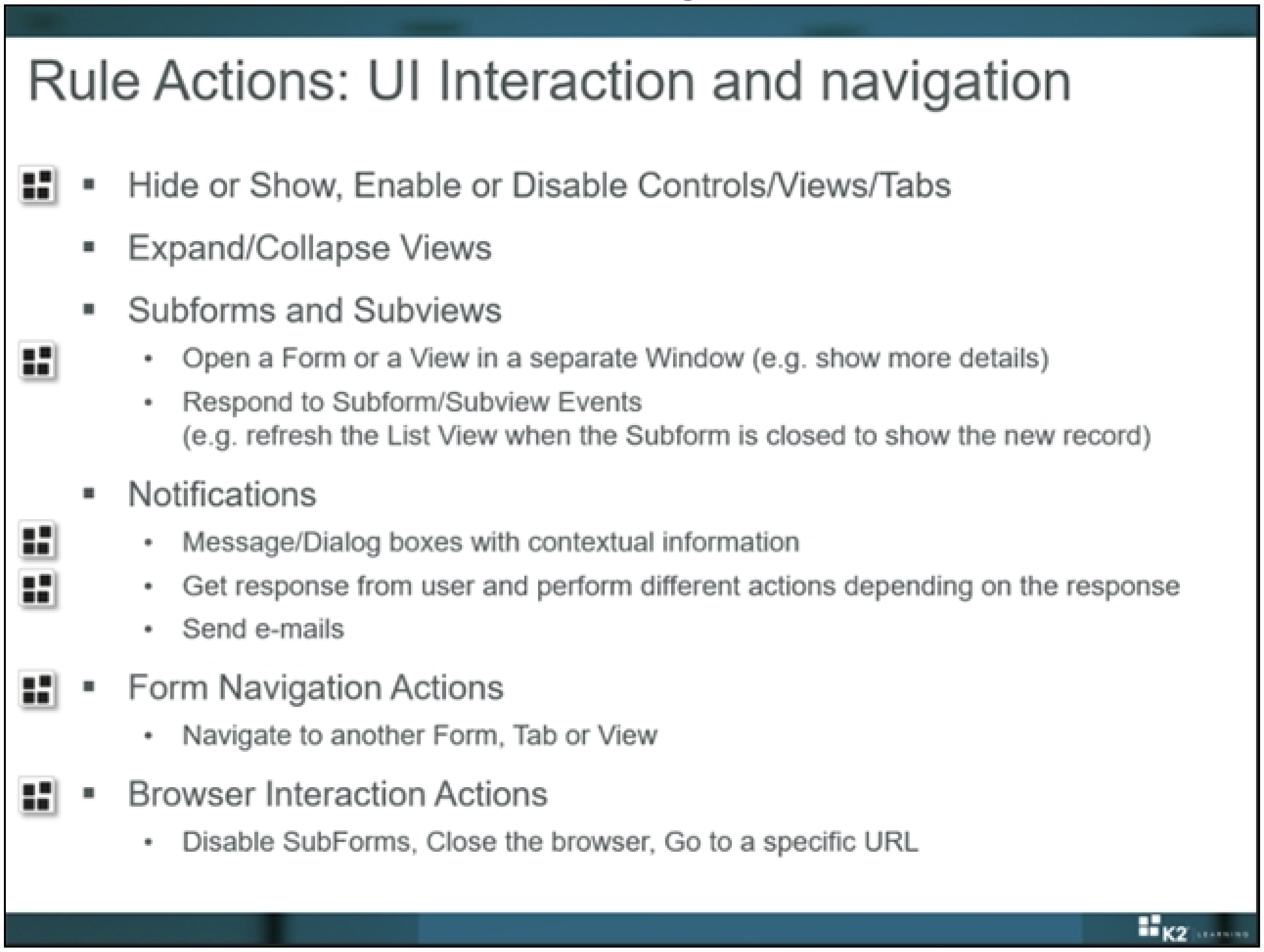
✓ Do not validate hidden, disabled or read-only controls, Views and tabs.

OK Cancel

Summary

- Rules are the “programming” behind Forms, Views and Controls
- A Rule has 3 parts: the Event (optional), Condition (optional) and Action(s)
- Rules can have multiple Actions
- Rules can be re-used with the “Execute another Rule” Action
- Rules can use Conditions to define that the Rule only continues if the Form passes validation

Rule Actions: UI Interaction and navigation



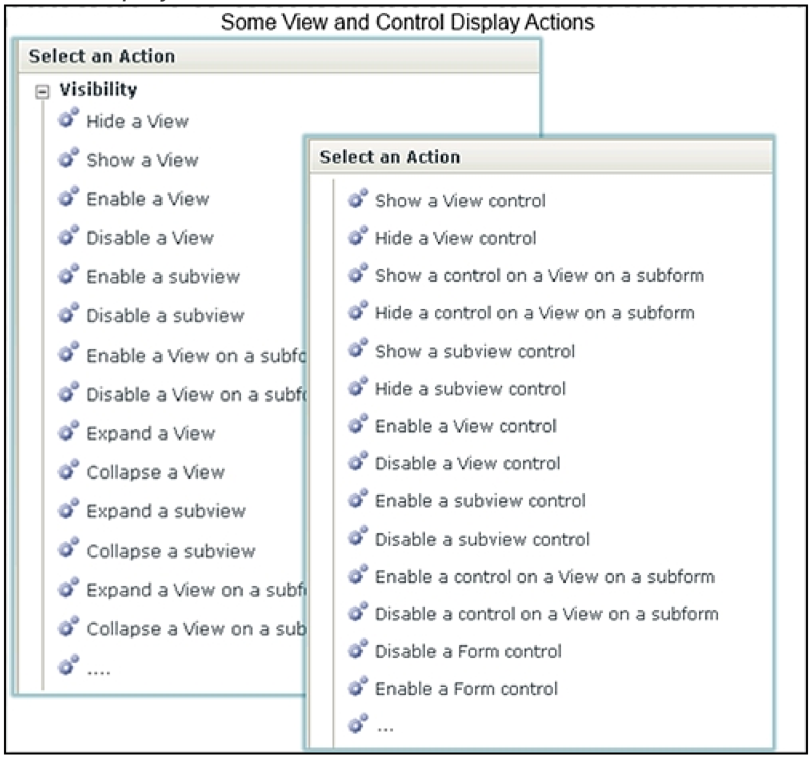
Rule Actions: UI Interaction and navigation

- Hide or Show, Enable or Disable Controls/Views/Tabs
- Expand/Collapse Views
- Subforms and Subviews
  - Open a Form or a View in a separate Window (e.g. show more details)
  - Respond to Subform/Subview Events (e.g. refresh the List View when the Subform is closed to show the new record)
- Notifications
  - Message/Dialog boxes with contextual information
  - Get response from user and perform different actions depending on the response
  - Send e-mails
- Form Navigation Actions
  - Navigate to another Form, Tab or View
- Browser Interaction Actions
  - Disable SubForms, Close the browser, Go to a specific URL

Video

Rules can be quite useful in formatting the user interface and interactions. When building user interfaces you often need to use dialog boxes, messages, pop-up forms and so on to build a friendly user experience. In K2 smartforms, there is a wide selection of Rule Actions and Events that can help you create advanced user interfaces.

Some display Actions



Sub-forms and Sub-views

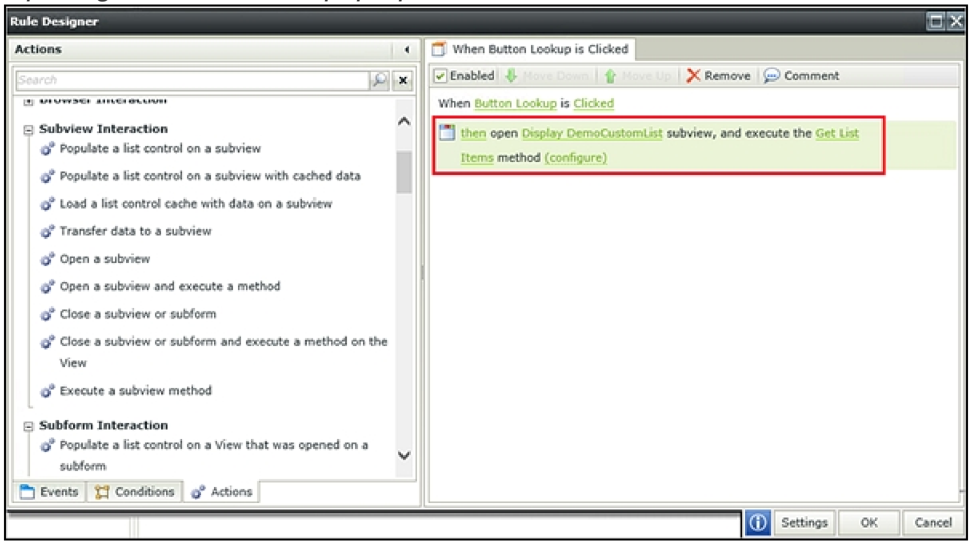
Sub-form Actions are used to open a Form in a separate window, while Sub-view Actions are used to open a View in a separate window. Apart from the Form/View difference, the Action behaves similarly for both.

Here is an example: suppose you want to create a button on a Form that the user can click to launch a "More Details.." Form. The button can have a Rule that will use the "open a Form in a Sub-form and execute a method on the sub-form" Action to load a specific Form, transfer data to that Form and then run a method on that Form to populate a View with data.

There is a large selection of Actions and Events that relate to Sub-forms and Sub-views, and you can perform very advanced Actions with these Events and Actions. Describing each approach is outside the scope of this course, but for now just remember that this functionality exists and feel free to play with it.

Sub-form Actions are also often used to show a customized dialog box or let the user select a value from a rich user interface. Your main Form can even respond to events raised by the sub-form, for example, when the user closes the sub-form, refresh the data on the main Form since the user may have entered new data in the sub-form. This is especially useful in an “Add Comment” scenario: perhaps the main Form shows a list of comments for a customer. The user may click on a button to “Add Comment”, a Sub-form or Sub-view dialog box pops up where the user can enter their comment, and when the dialog box is closed, the list of Comments for the customer is refreshed so that the user can see their new comment.

Opening a Sub-view as a pop-up box


Rule Designer

Actions

Search

Browser Interaction

Subview Interaction
- Populate a list control on a subview
- Populate a list control on a subview with cached data
- Load a list control cache with data on a subview
- Transfer data to a subview
- Open a subview
- Open a subview and execute a method
- Close a subview or subform
- Close a subview or subform and execute a method on the View
- Execute a subview method

Subform Interaction
- Populate a list control on a View that was opened on a subform

Events | Conditions | Actions

When Button Lookup is Clicked

Enabled | Move Down | Move Up | Remove | Comment

When Button Lookup is Clicked

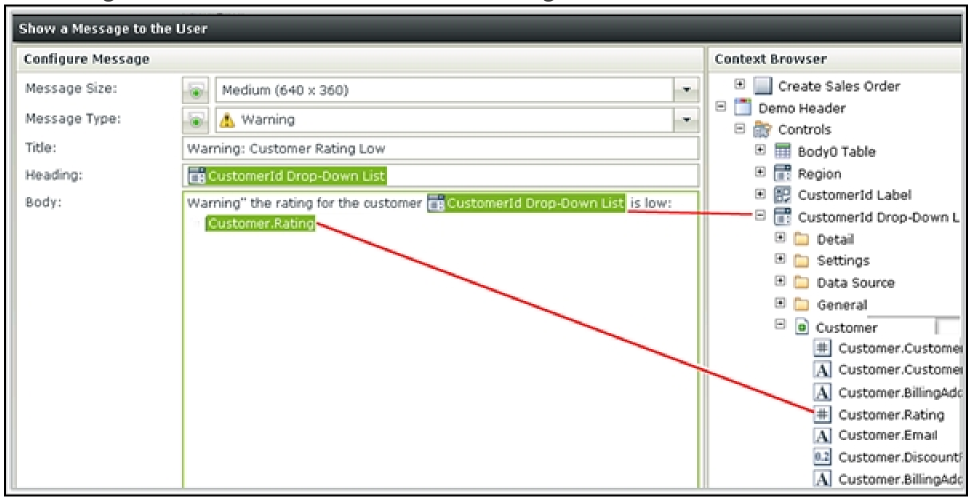
then open Display DemoCustomList subview, and execute the Get List Items method (configure)

Settings | OK | Cancel

Notifications

You can also use Notifications to show a “dialog box” or otherwise notify a user. There are several notification actions available to show dialog boxes and get confirmation from a user, show configurable message boxes, browse to a message Form or send E-mails. You can also use the Context Browser to include contextual data on the message screen, as shown below.

Including contextual information in a message box


Show a Message to the User

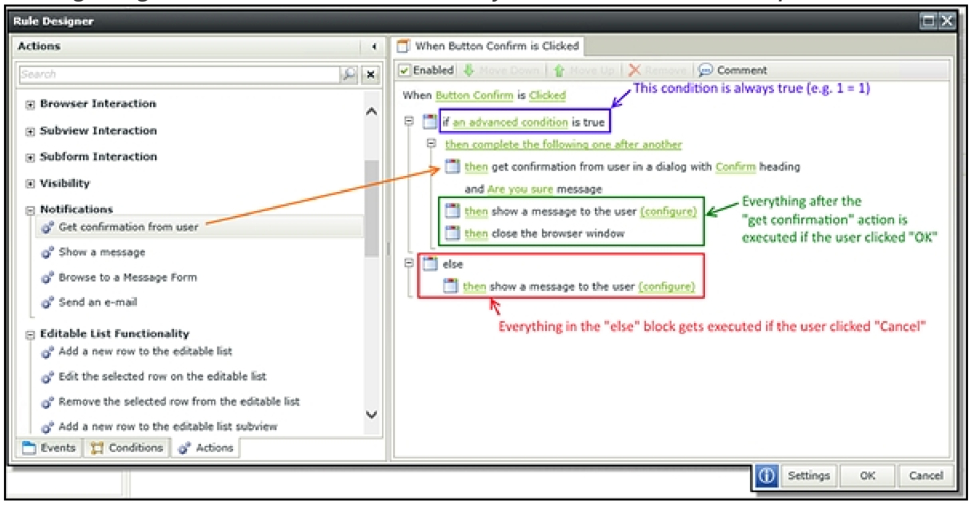
Configure Message

Message Size:	Medium (640 x 360)
Message Type:	Warning
Title:	Warning: Customer Rating Low
Heading:	CustomerId Drop-Down List
Body:	Warning" the rating for the customer CustomerId Drop-Down List is low: Customer.Rating

Context Browser
- Create Sales Order
- Demo Header
  - Controls
    - Body0 Table
    - Region
    - CustomerId Label
    - CustomerId Drop-Down L
      - Detail
      - Settings
      - Data Source
      - General
      - Customer
        - Customer.Custome
        - Customer.Custome
        - Customer.BillingAdd
        - Customer.Rating
        - Customer.Email
        - Customer.Discount
        - Customer.BillingAdd

And you can even perform different actions, depending on whether the user clicks "OK" or "Cancel" in the confirmation message screen. The screenshot below shows how to configure Rules to achieve this behavior.

Configuring a Rule to execute differently based on a user's response


Rule Designer
Actions
Search
Browser Interaction
Subview Interaction
Subform Interaction
Visibility
Notifications
Get confirmation from user
Show a message
Browse to a Message Form
Send an e-mail
Editable List Functionality
Add a new row to the editable list
Edit the selected row on the editable list
Remove the selected row from the editable list
Add a new row to the editable list subview
Events
Conditions
Actions
When Button Confirm is Clicked
Enabled
Move Down
Move Up
Remove
Comment
Settings
OK
Cancel

When Button Confirm is Clicked

- if an advanced condition is true
  - then complete the following one after another
    - then get confirmation from user in a dialog with Confirm heading and Are you sure message
    - then show a message to the user (configure)
    - then close the browser window
- else
  - then show a message to the user (configure)

This condition is always true (e.g. 1 = 1)

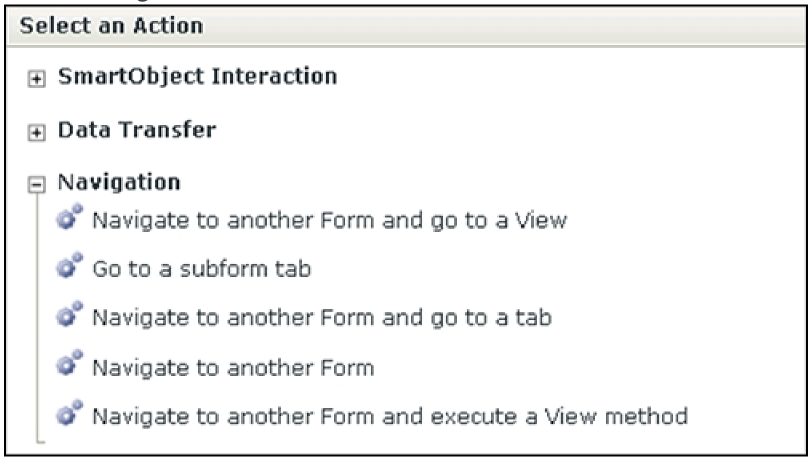
Everything after the "get confirmation" action is executed if the user clicked "OK"

Everything in the "else" block gets executed if the user clicked "Cancel"

Navigation

Navigation actions can be used to send the user to a different Form, Tab or View. (The difference between Sub-forms and Navigation is that sub-forms are shown as new windows, while navigation moves the current window to a different Form.) You can even configure a Rule to navigate to a different Form, pass a value from the current Form into the new Form and then execute a View method on the new Form.

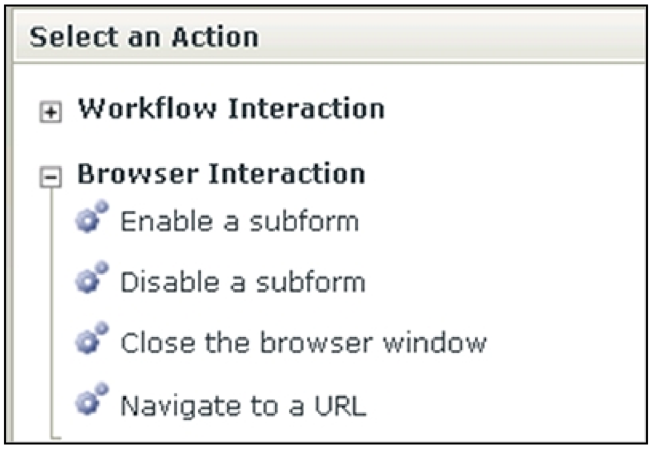
Form Navigation Actions


Select an Action
SmartObject Interaction
Data Transfer
Navigation
Navigate to another Form and go to a View
Go to a subform tab
Navigate to another Form and go to a tab
Navigate to another Form
Navigate to another Form and execute a View method

Browser Interaction

The available Browser Interaction Actions allow you to interact with the user's actual web browser, for example to close the browser, navigate to a specific URL or enable/disable the current Form.

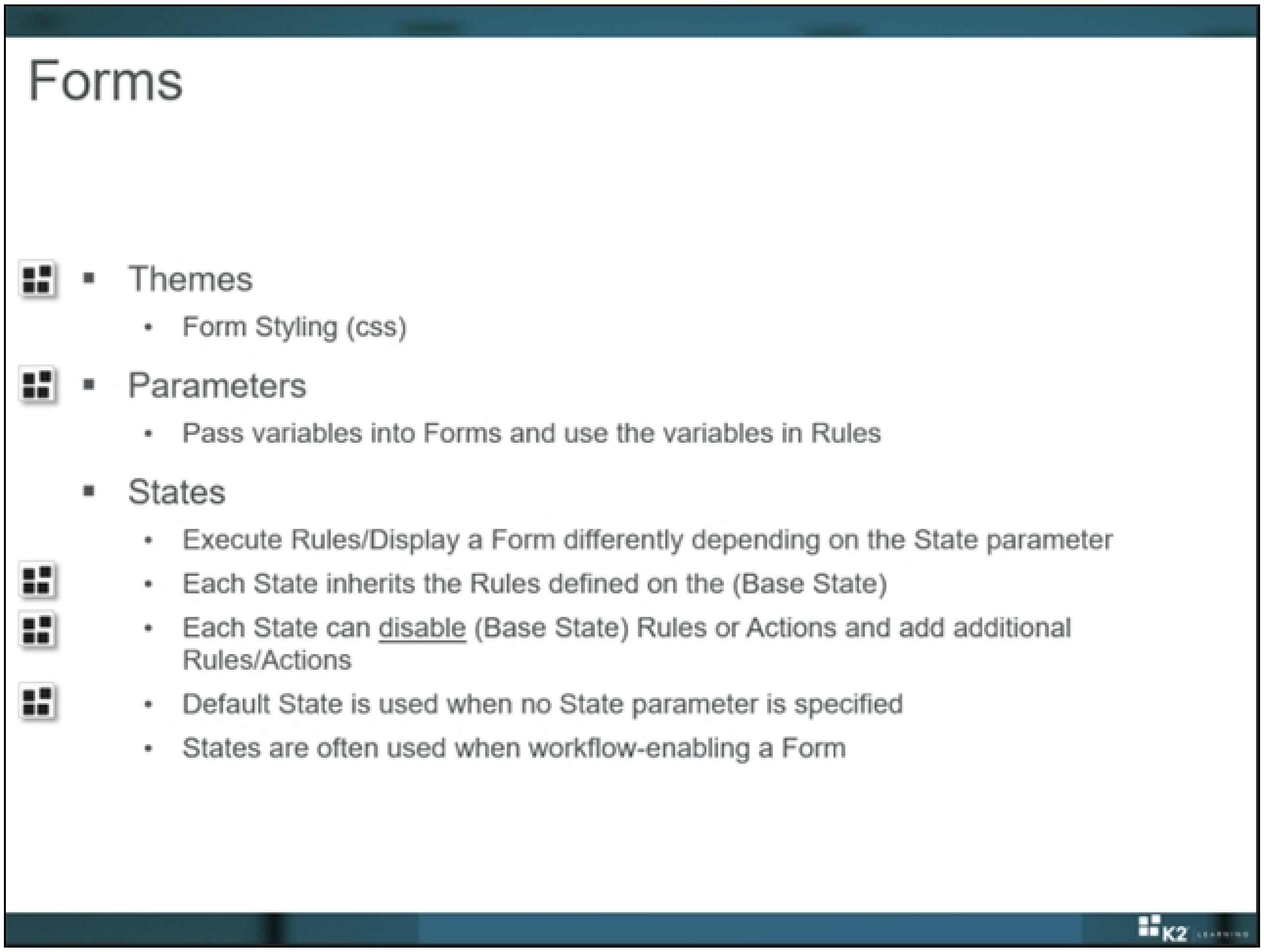
Browser Interaction Actions



Summary

- You can use Rule Actions to perform UI interaction and navigation operations such as showing other Views, dialogs, getting confirmation from users or going to another Form.
- You can use Rule Actions to show/hide, expand/collapse and enable/disable Views.

Forms



Forms

- Themes
  - Form Styling (css)
- Parameters
  - Pass variables into Forms and use the variables in Rules
- States
  - Execute Rules/Display a Form differently depending on the State parameter
  - Each State inherits the Rules defined on the (Base State)
  - Each State can disable (Base State) Rules or Actions and add additional Rules/Actions
  - Default State is used when no State parameter is specified
  - States are often used when workflow-enabling a Form

End users usually interact with a Form, which hosts one or more Views. This means that a View must be added to a Form before you will be able to share the (Form) User Interface with end users (unless you're using a Sub-view).

A Form can host one of more Views, and a View can be re-used on multiple Forms. This last point is especially important to bear in mind: because Views can be re-used on Forms, you can define generic Views (“Customer Information” for example) and then re-use this same View on multiple Forms (“Customer Order”, “Customer Complaint”, “Customer Service Request” and so on.) This can save a lot of time when creating User Interfaces, and we encourage you to create generic Views for common enterprise data.

Apart from Views, Forms can also host Form-level Controls and Rules, and Form-level Rules have access to all the Controls and Rules on the Views within the Form. Remember that a View cannot access other Views, but if two Views are hosted on the same Form, the Form can access everything in those two Views.

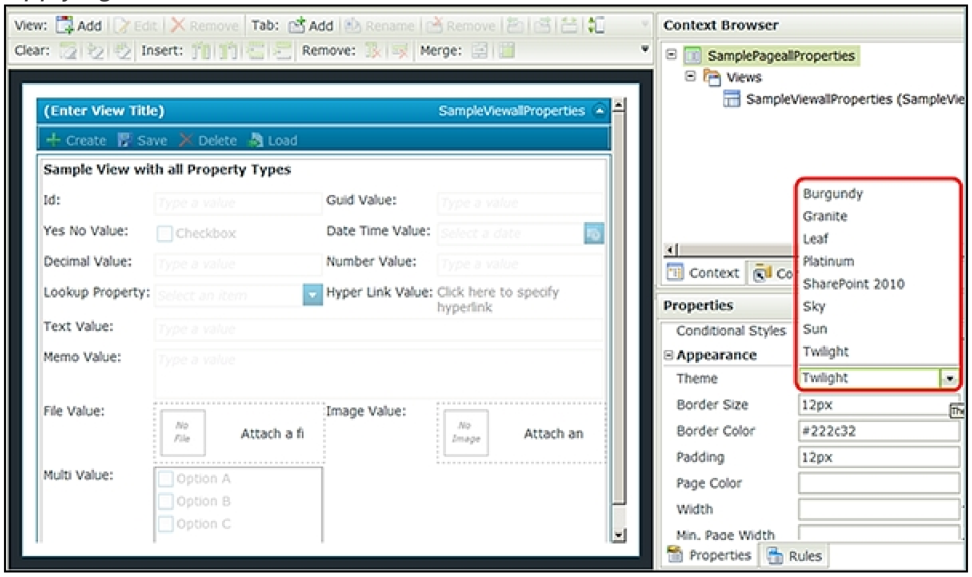
Forms have additional properties and components, so let's take a look at those.

Themes

Themes are used to format the display and styles of all controls on a Form. If you are familiar with web development, think of themes as style sheets. (Style sheets are commonly used to format the overall look of a page, such as colors and fonts.) When designing a Form, you can use the Theme drop-down on the Form’s Properties tab to select a different display style for the Form.

You can also set a Form's Theme dynamically at runtime by passing a Query String parameter in the format **_theme=[ThemeName]**

Applying a Theme to a Form


View: Add Edit Remove Tab: Add Rename Remove
Clear: Insert: Remove: Merge:
(Enter View Title) SampleViewallProperties
Create Save Delete Load
Sample View with all Property Types
Context Browser
SamplePageallProperties
Views
SampleViewallProperties (SampleVie
Burgundy
Granite
Leaf
Platinum
SharePoint 2010
Sky
Sun
Twilight
Context
Properties
Rules

Id:	Type a value
Guid Value:	Type a value
Yes No Value:	Checkbox
Date Time Value:	Select a date
Decimal Value:	Type a value
Number Value:	Type a value
Lookup Property:	Select an item
Hyper Link Value:	Click here to specify hyperlink
Text Value:	Type a value
Memo Value:	Type a value
File Value:	No File Attach a fi
Image Value:	No Image Attach an
Multi Value:	Option A, Option B, Option C

Properties

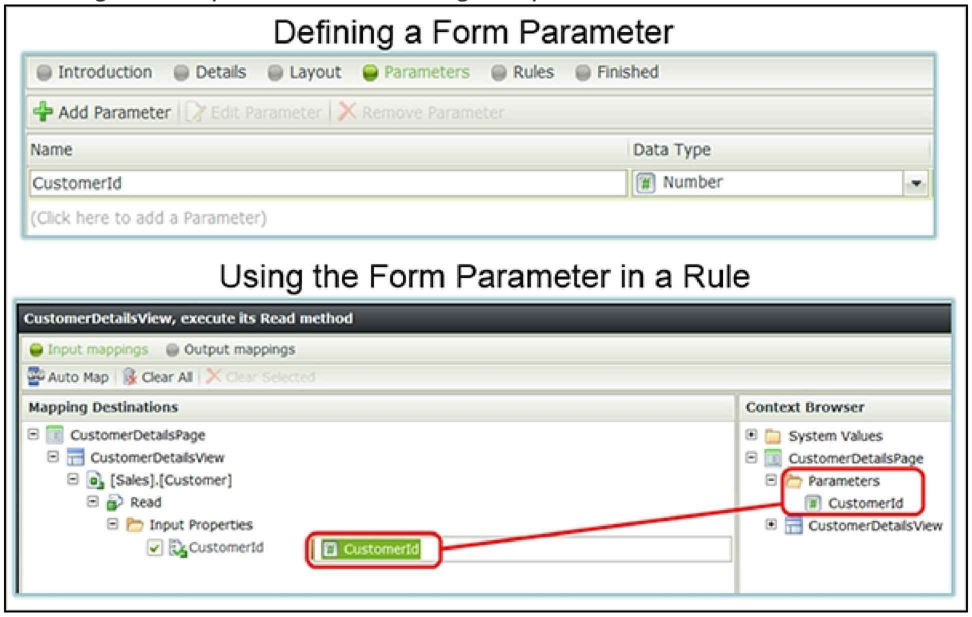
Conditional Styles	
Appearance	
Theme	Twilight
Border Size	12px
Border Color	#222c32
Padding	12px
Page Color	
Width	
Min. Page Width	

Parameters

Parameters are used to pass values into a Form at runtime. These parameter values are often used by Form-level Rules to populate a property and then execute a SmartObject method. Here is an example: suppose you have a Customer Details Form that displays customer information. This Form uses a View that exposes the Customer SmartObject's Read method. The Read method requires a Customer ID as an input parameter. Now suppose you want to be able to open the Form with a Customer ID, so that as soon as the Form opens, the View is populated with a specific customer's details. To achieve this, you would define a Form Parameter and then use the Parameter in a Form-level Rule, passing the parameter value to the input property of the View's Read method.

Note
You can also define parameters on Views.

Defining a Form parameter and using that parameter in a Rule



Name	Data Type
CustomerId	Number
(Click here to add a Parameter)	

Form States

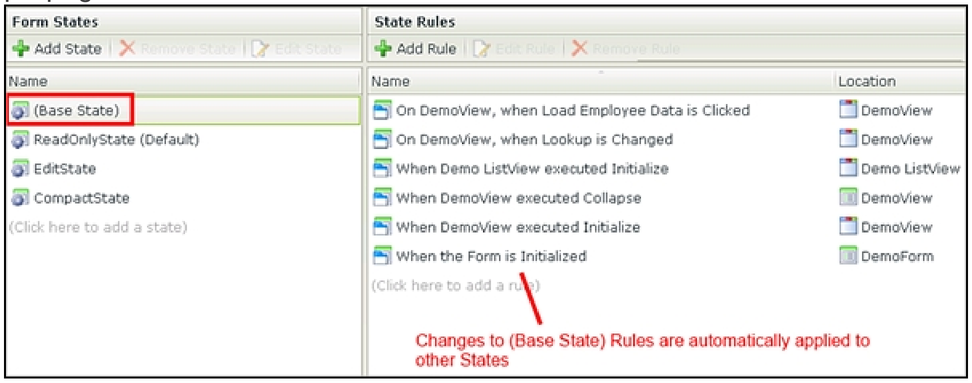
States are a way to re-use the same Form but apply different Rules depending on the State that the Form is currently in. Here is an example: suppose you have a complex Expense Claim Approval Form that is used in an Expense Claim Approval workflow. You would like to use the same Form to start the workflow and during the subsequent approval tasks, but the Rules on the Form are slightly different depending on whether the Form is being used for submission or for approval.

This is where Form States are useful. You can define multiple Form States for the same Form, and then configure additional Rules that are specific to the current Form State. For example, if the Form State is “submission” then enable all the input controls on the Form and when the submit button is clicked, create the Expense Claim record and start the Approval Workflow. If the Form State is “Approval”, then open the worklist item with the Form Serial Number, and disable all the input controls on the Form except for the manager’s approval decision drop-down menu.

The State of a Form is usually passed in as a Query String parameter called "**_State=[StateName]**"

States are usually used in conjunction with Rules, so you would define additional States for a Form on the Rules wizard page, as shown below.

All Forms have a (Base State) with a collection of Rules. Any changes to the (Base State) Rules are automatically propagated to the other States as well.



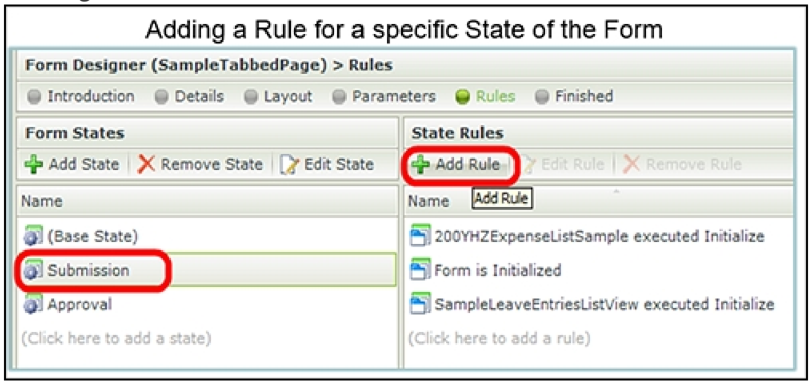
Name
(Base State)
ReadOnlyState (Default)
EditState
CompactState
(Click here to add a state)

Name	Location
On DemoView, when Load Employee Data is Clicked	DemoView
On DemoView, when Lookup is Changed	DemoView
When Demo ListView executed Initialize	Demo ListView
When DemoView executed Collapse	DemoView
When DemoView executed Initialize	DemoView
When the Form is Initialized	DemoForm
(Click here to add a rule)	

Changes to (Base State) Rules are automatically applied to other States

You can add additional Rules to specific States that will only execute when the Form is in that State. Alternatively, you can also disable the Rules executed by the Base State if you prefer.

Adding a Rule to a State

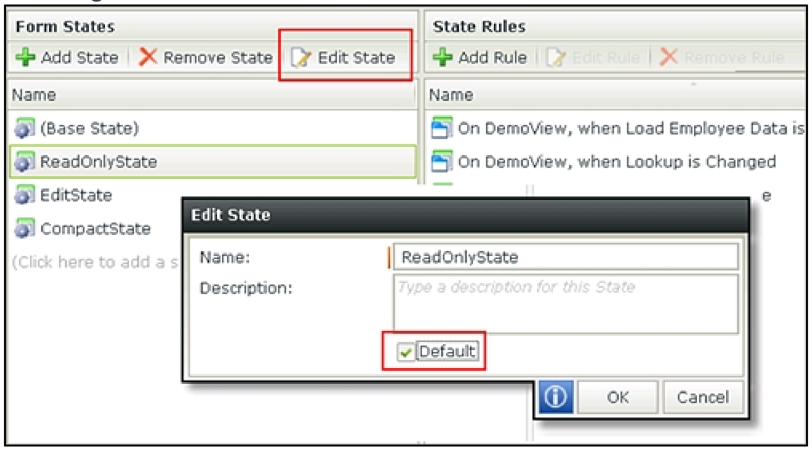

Adding a Rule for a specific State of the Form

Name
(Base State)
Submission
Approval
(Click here to add a state)

Name
200YHZExpenseListSample executed Initialize
Form is Initialized
SampleLeaveEntriesListView executed Initialize
(Click here to add a rule)

You can mark a specific State as the Default State, in which case the Form will render in this State if no State parameter was passed in.

Marking a State as the Default State



Name: ReadOnlyState
Description: Type a description for this State
Default ✓

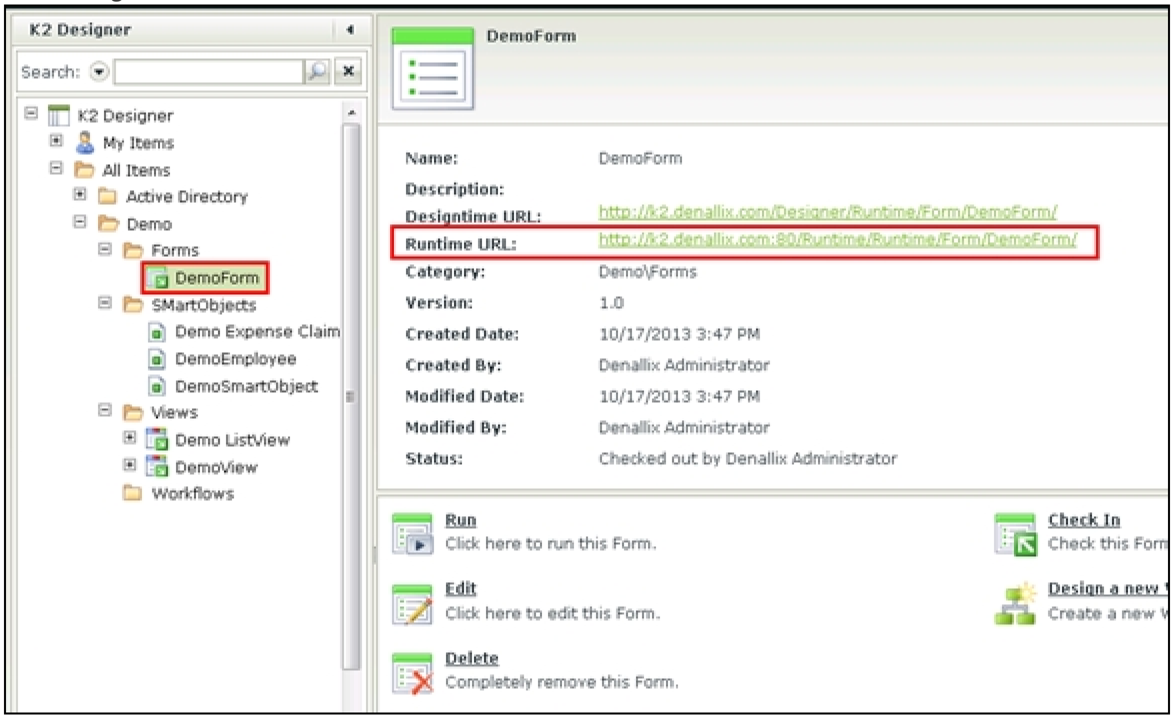
Displaying Forms to End Users

As we have mentioned earlier, end users usually interact with Forms, but you may be wondering how the end users actually do this. Remember that Forms are just web pages, which means that end-users can interact with Forms using a web browser. The key is in giving the users the URL required to open the Form, or configuring another web page to open the Form in an iFrame. Let’s look at the most common approaches used:

Opening a Form with a URL from a client device or computer

Users can open Forms directly with a URL. You can retrieve the URL for a Form by opening a Form’s properties from the Category System and then copying the Runtime URL property, as shown below:

Obtaining the runtime URL for a Form



Name:	DemoForm
Description:	
Designtime URL:	http://k2.denallix.com/Designer/Runtime/Form/DemoForm/
Runtime URL:	http://k2.denallix.com:80/Runtime/Runtime/Form/DemoForm/
Category:	Demo\Forms
Version:	1.0
Created Date:	10/17/2013 3:47 PM
Created By:	Denallix Administrator
Modified Date:	10/17/2013 3:47 PM
Modified By:	Denallix Administrator
Status:	Checked out by Denallix Administrator

Once you have the URL, you can share this with users or add it as a hyperlink to another web page. Forms that use Parameters or States may need extra parameters added to the Form URL so that it renders correctly at runtime.

Displaying a Form in SharePoint with a Web Part

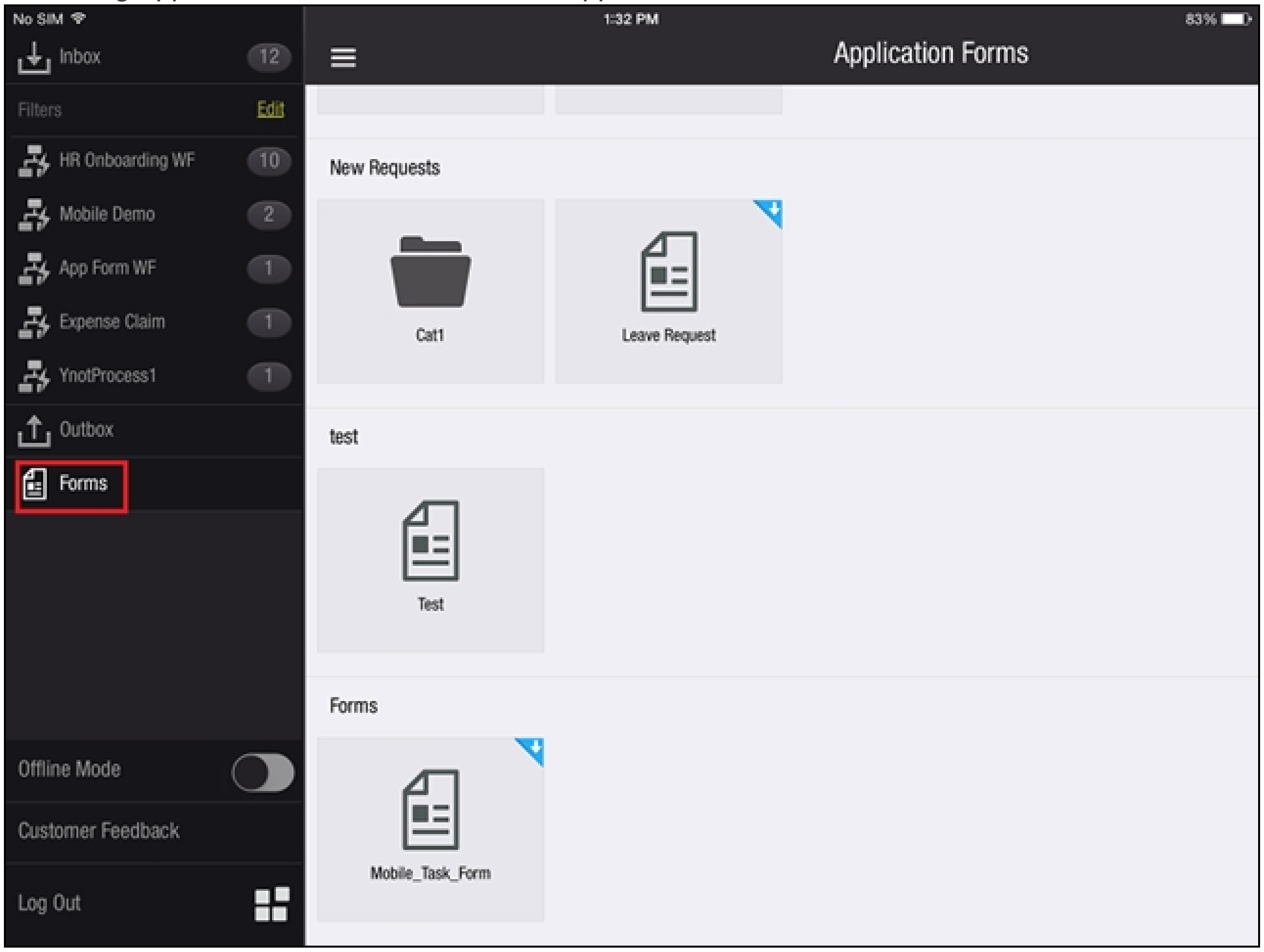
If your organization uses SharePoint, it is very simple to show K2 smartforms in your SharePoint environment. K2 provides the K2 smartforms Viewer Web Part that you can use to display a Form in a SharePoint page. Since SmartForms are just web-based forms, you can also use the Standard SharePoint Page Viewer Web Part and just configure the web part with the Form URL as explained above.

Bear in mind that K2 smartforms are NOT actually deployed, uploaded or published to SharePoint. You do not need to publish or otherwise activate or enable the Forms themselves in SharePoint. Instead, the Forms are published to a K2 smartforms server as normal, and the Web Parts are just iFrame controls that show the target Form presented by the SmartForms server.

Using the K2 mobile applications

Users can also access Forms from K2's mobile applications like the iOS or Android mobile applications. To do so, the Forms must be enabled as Application Forms in the Form properties pane.

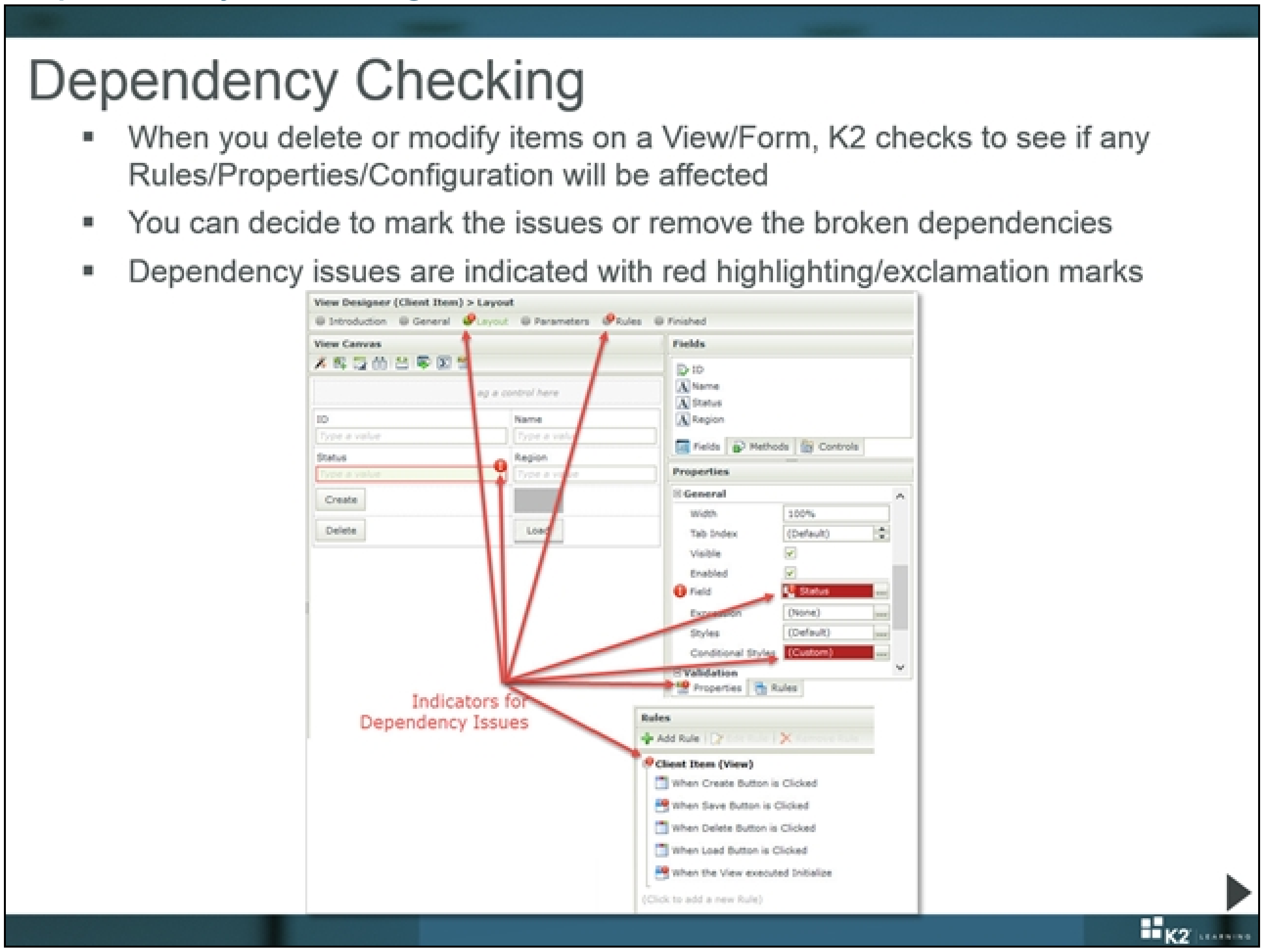
Accessing Application Forms in the K2 Mobile application



Summary

- Forms are containers for Views.
- Themes are applied to Forms (at design time) and can change the look/feel (styling) of the Form.
- States are used to execute the same Form in different ways, and is most often used when the same Form is used in different steps of a Workflow.
- All States inherit Rules from the Base State and changes on the Base State are propagated to the other States. Other States can disable the Base State Rules and add their own Rules as well.
- End users can access Forms by their URL or through the K2 mobile application.

Dependency Checking



Video

Dependency Checking

- When you delete or modify items on a View/Form, K2 checks to see if any Rules/Properties/Configuration will be affected
- You can decide to mark the issues or remove the broken dependencies
- Dependency issues are indicated with red highlighting/exclamation marks

View Designer (Client Item) > Layout
Introduction
General
Layout
Parameters
Rules
Finished
View Canvas
Fields
Properties
Rules
Indicators for Dependency Issues
Client Item (View)
When Create Button is Clicked
When Save Button is Clicked
When Delete Button is Clicked
When Load Button is Clicked
When the View executed Initialize
K2 LEARNING

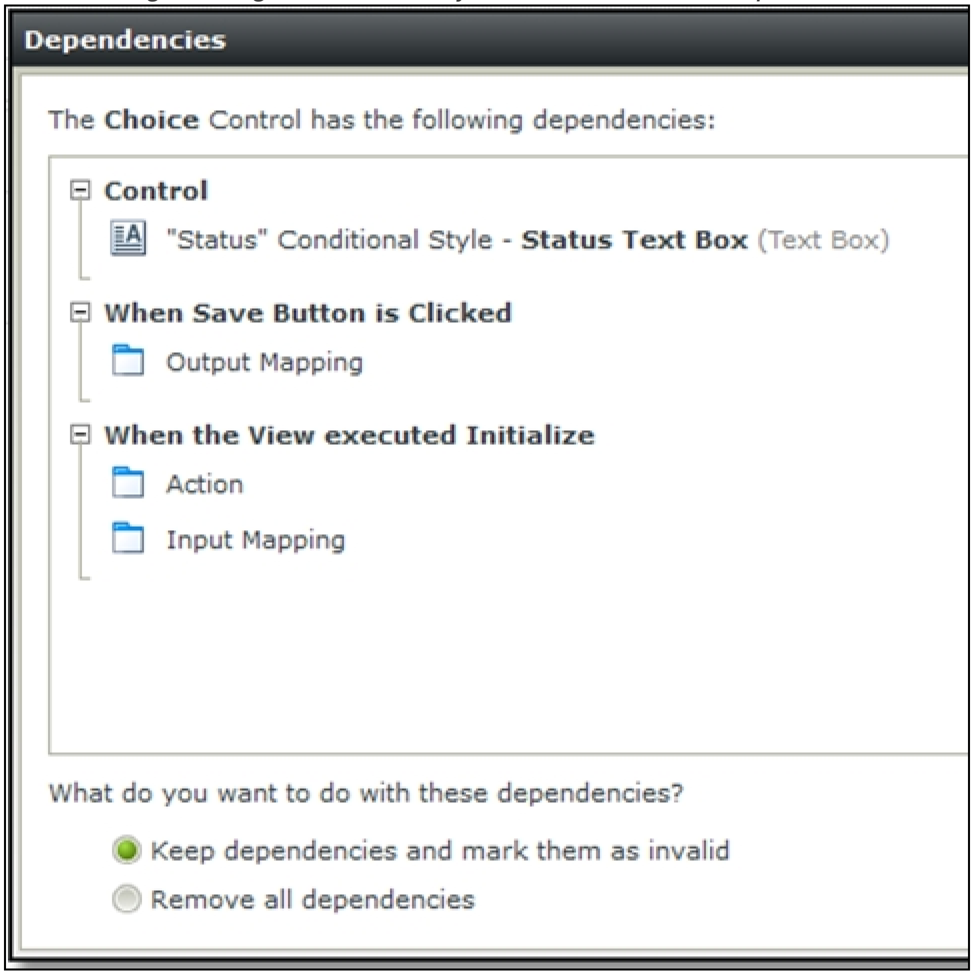
When you build Forms and Views, add and configure Rules and Controls and map to SmartObjects or workflows, there may be many dependencies between these items under the covers. For example: If a view contains a text box and the text box is associated with a rule, a dependency exists between the text box and the associated rule. If the text box is deleted, the rule can't execute because of the missing dependency.

To help you manage these dependencies, the K2 Designer tool provides dependency checking to help identify and address dependency issues on Views and Forms through visual indicators that show where issues have been identified.

When you delete an item from a View or Form, K2 will attempt to identify dependencies for that item and warn you of the issue. You can then decide to keep dependencies and mark them as invalid, which will flag the identified issues with red indicators, or you can decide to remove all dependencies, which will automatically delete any dependencies for the item being removed.

These dependency checks also assist when you are packaging and deploying applications between environments, because you would typically have to address any dependency issues before you create a package of Forms and Views.

The warning message shown when you delete items with dependencies



Dependencies

The **Choice** Control has the following dependencies:

Control

"Status" Conditional Style - **Status Text Box** (Text Box)

When Save Button is Clicked

Output Mapping

When the View executed Initialize

Action

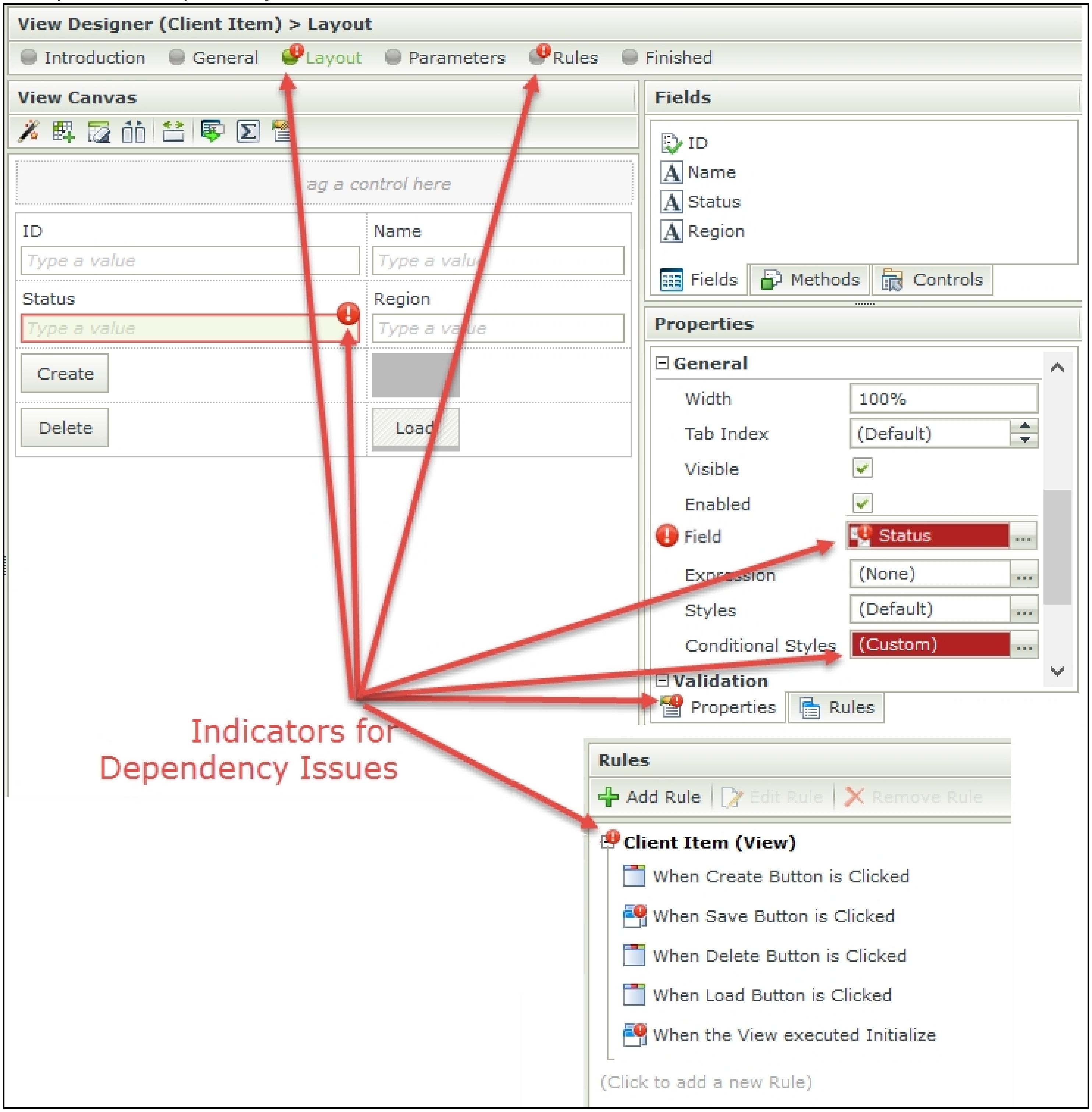
Input Mapping

What do you want to do with these dependencies?

[✓ Keep dependencies and mark them as invalid]

Remove all dependencies

Examples of the Dependency Issue indicators



Summary

- K2 will perform dependency verification when you edit or delete items on Forms or Views
- This helps you to “fix” issues when items are modified or deleted on Forms
- Use the red exclamation marks to find and fix issues

MASTERY CHECKPOINT: Rules and Forms

Rules and Forms

- Adding and configuring Rules
- Events, Conditions, and Actions in Rules
- Using Rules for user interface interaction
- Adding Views to Forms
- Setting Form Properties

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 3 of this module: Rules and Forms. If you are attending instructor-led training, use this as an opportunity to answer any questions around Rules and Forms.

By now you should know:

- How to work with and configure Rules
- The Rules components (Events, Conditions, Actions)
- How to add Views to Forms
- How to set Form Properties
- How to expose Forms to users

Knowledge-check questions

Q: Thinking about Rules, can you describe Events, Conditions and Actions?
Reveal answer**A:**
Events: something that happens (e.g. a button is clicked, a Control is changed). (WHEN)
Conditions: a conditional test that will allow or prevent the Rule Actions from firing (for example “amount is over 2000”) (IF)
Actions: the operations that the Rule performs (e.g. calling a SmartObject method, setting a field value) (DO)

Q: True or False: a Form-level Rule can set Control values in a View that exists in that Form.
Reveal answer**A:** True

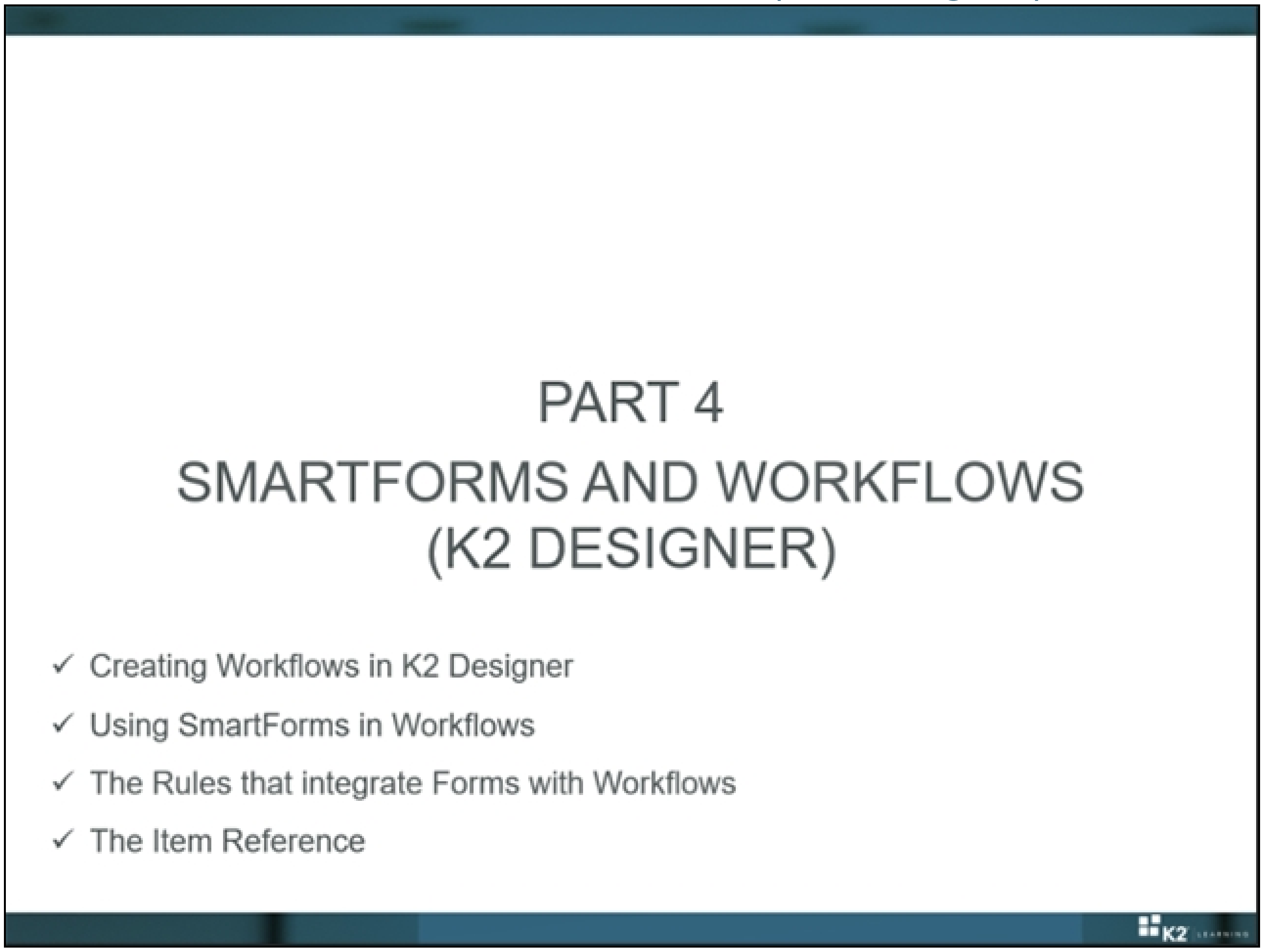
Q: True of False: a View-level Rule can set control values in another View that exists in the same Form.
Reveal answer**A:** False. Views can only “see” themselves, Forms can see themselves and all their Views.

Q: Can you define a Rule on the View level?
Reveal answer**A:** Yes

Q: Can you define a Rule for a specific Control?
Reveal answer**A:** Yes

Q: True or False: a Form can contain multiple Views, and those Views could be List Views and/or Item Views.
Reveal answer**A:** True

PART 4: SmartForms and Workflows (K2 Designer)



PART 4

SMARTFORMS AND WORKFLOWS (K2 DESIGNER)

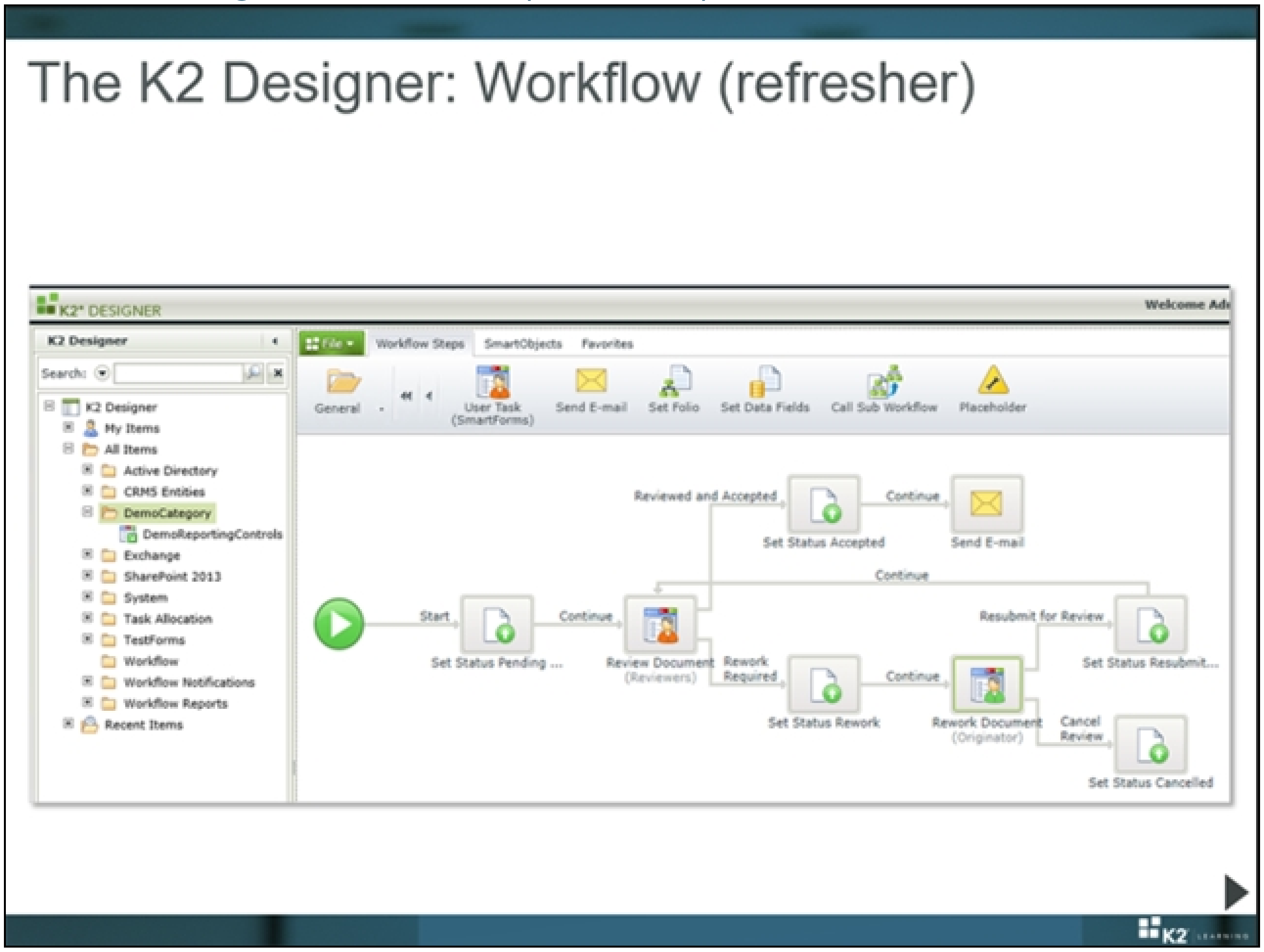
- ✓ Creating Workflows in K2 Designer
- ✓ Using SmartForms in Workflows
- ✓ The Rules that integrate Forms with Workflows
- ✓ The Item Reference

Video

In Part 4 we will look at Workflows in more detail. Note that we will only be working with Workflows built with K2 Designer in this module, the next module will look at Workflows built with K2 Studio.

- How to use SmartForms to start a Workflow and to complete a user task in a Workflow
- We will look at the Rule Actions that make the integration between Workflows and Forms work
- Finally we will investigate the Item Reference ("primary SmartObject record associated with the application") concept
- At the end of Part 4, we will build a Workflow that is integrated with Forms. One Form will start the Workflow, the other will be used for the user review task.

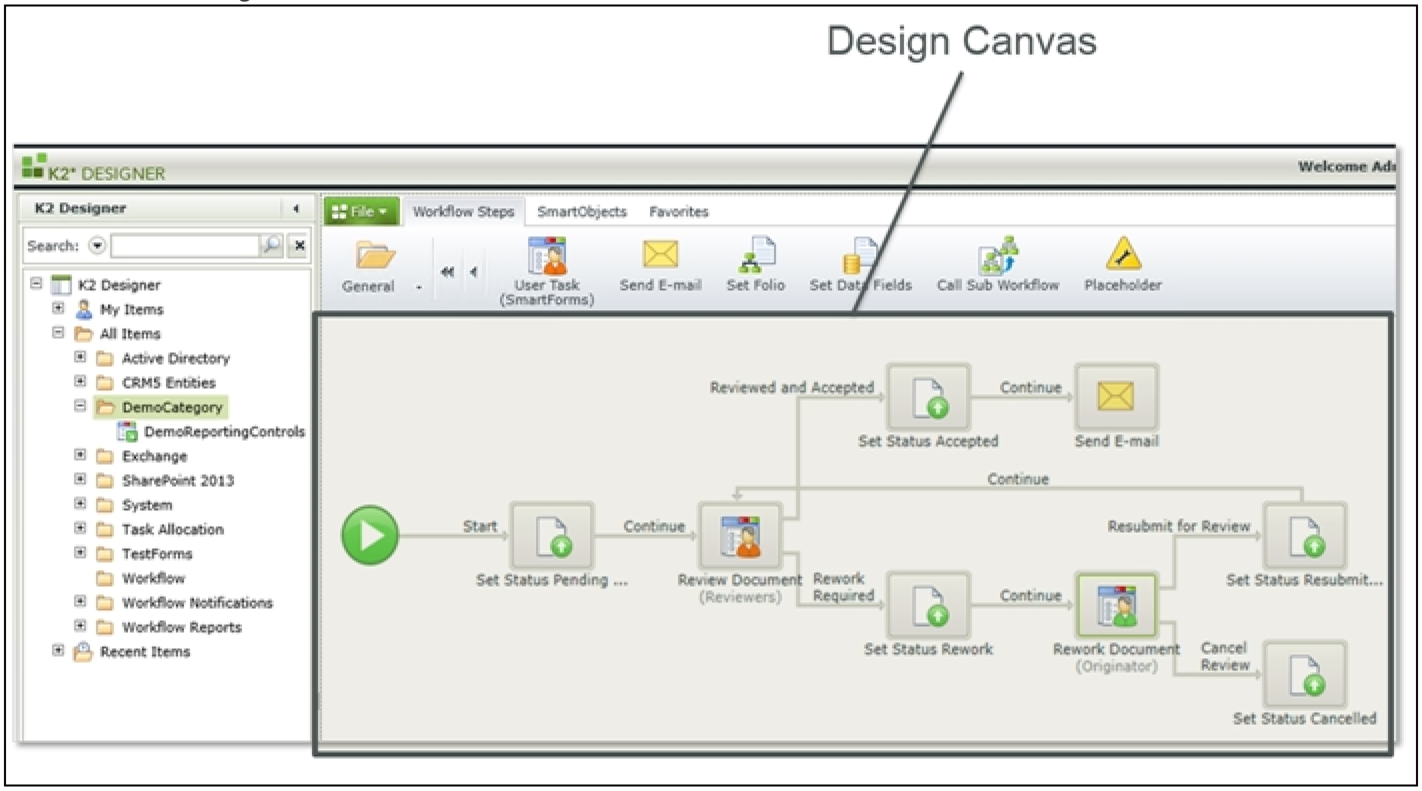
The K2 Designer: Workflow (refresher)



Video

In addition to K2 Studio or K2 for Visual Basic, a common tool for building Workflows is K2 Designer. The Workflow designer looks a little different to the Forms and SmartObjects designers, but the basic operation is very simple: you can add steps to the Workflow by dragging and dropping a wizard onto the Workflow design canvas as a step, and then configuring the wizard by running through the wizard screens. This Workflow design tool does not allow you to drag and drop steps anywhere you want: steps (events) must be dropped into blank steps and K2 will automatically determine the layout of the Workflow for you. You can replace existing steps with other steps, and use Outcomes (described later) to add additional paths to the Workflow.

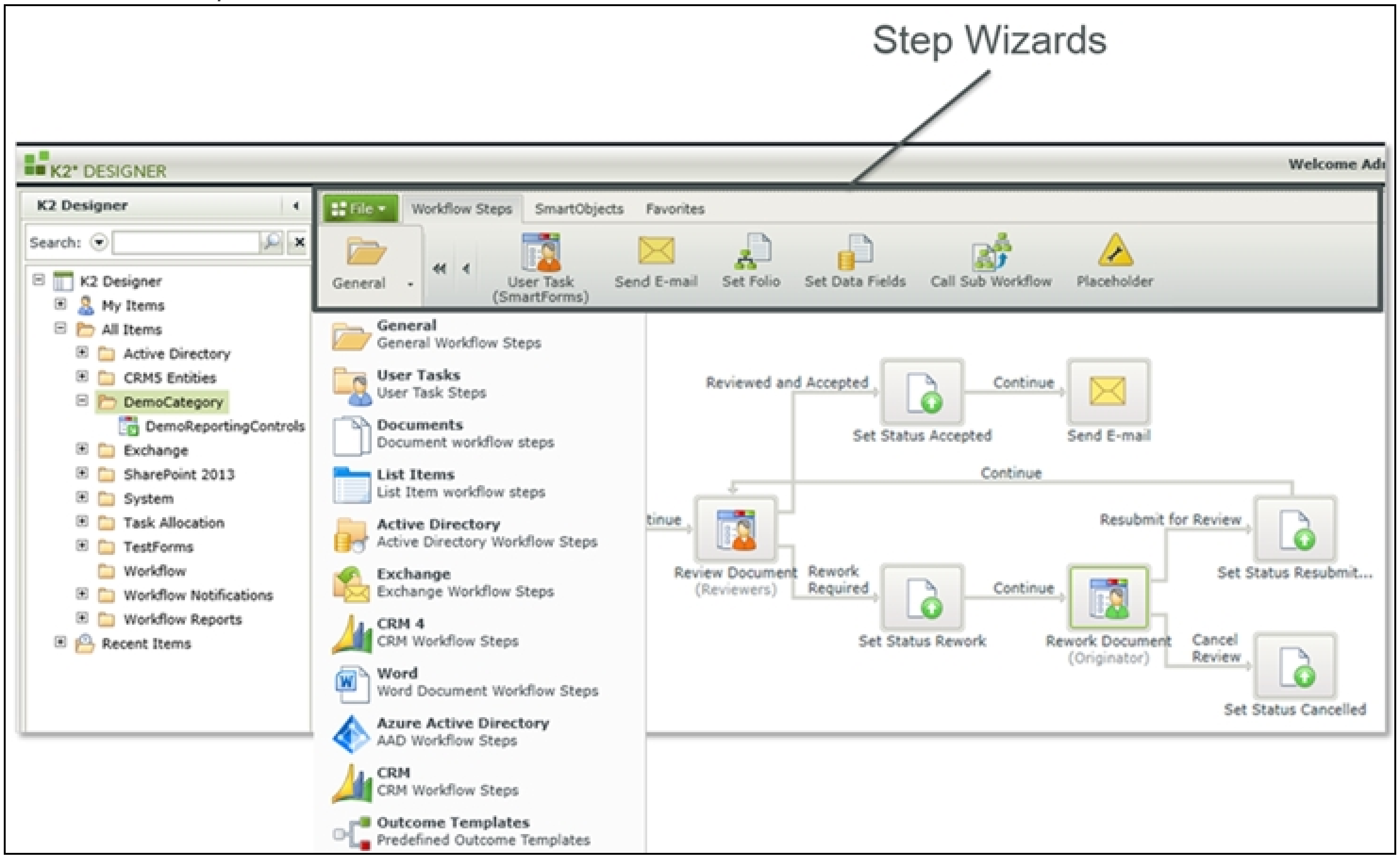
The Workflow design canvas



The available Workflow step wizards are grouped into categories. To select a different type of wizard, click on the category selection button (left-hand side of the steps toolbar) and then select a different category of wizards to show the available steps.

Note
The available wizards will depend on the version of K2 being used and potentially which components were activated in your K2 environment. The screenshot below shows samples of the available wizard types.

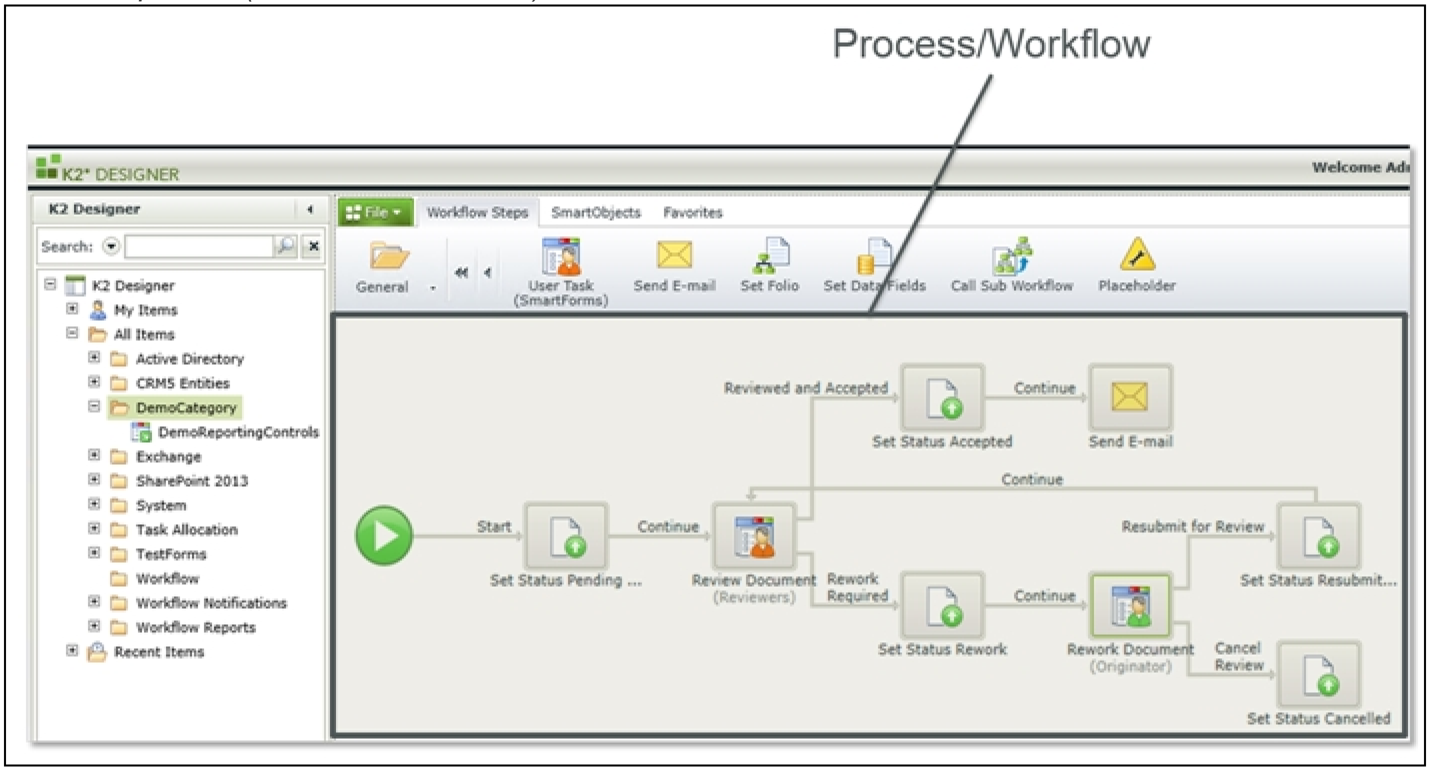
The Workflow Steps toolbar



A process (also known as a Workflow) refers to the entire process designed in the K2 Designer tool. A process will have at least a Start step and one Participant or Server step.

Processes usually contain multiple Steps, which are joined with Outcomes. The steps and outcomes define the flow of the process. From a K2 smartforms perspective, SmartForms can be used to start a Workflow and can be used as the user interface for user tasks in a Workflow.

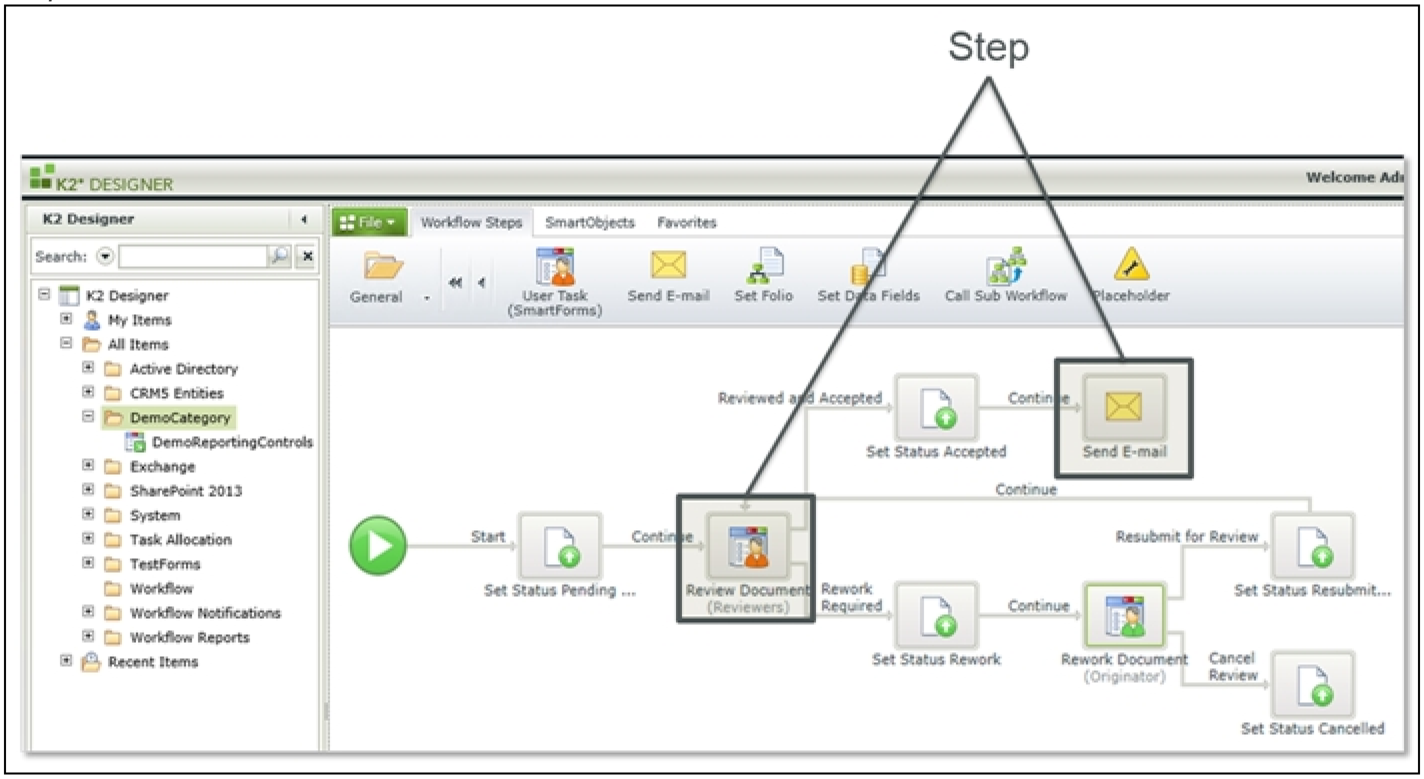
The entire process (also called "Workflow")



A Step is a unit of work in a process, which is performed by a human or system. When considering a process design, whenever work moves from one person to another or when the K2 server needs to perform some work, a step will be created for each of these work items.

In the diagram below, two steps are highlighted. In this case, one step (Review Document) is performed by a user, and the other step (Send E-mail) is performed by the K2 server.

Steps in a Workflow

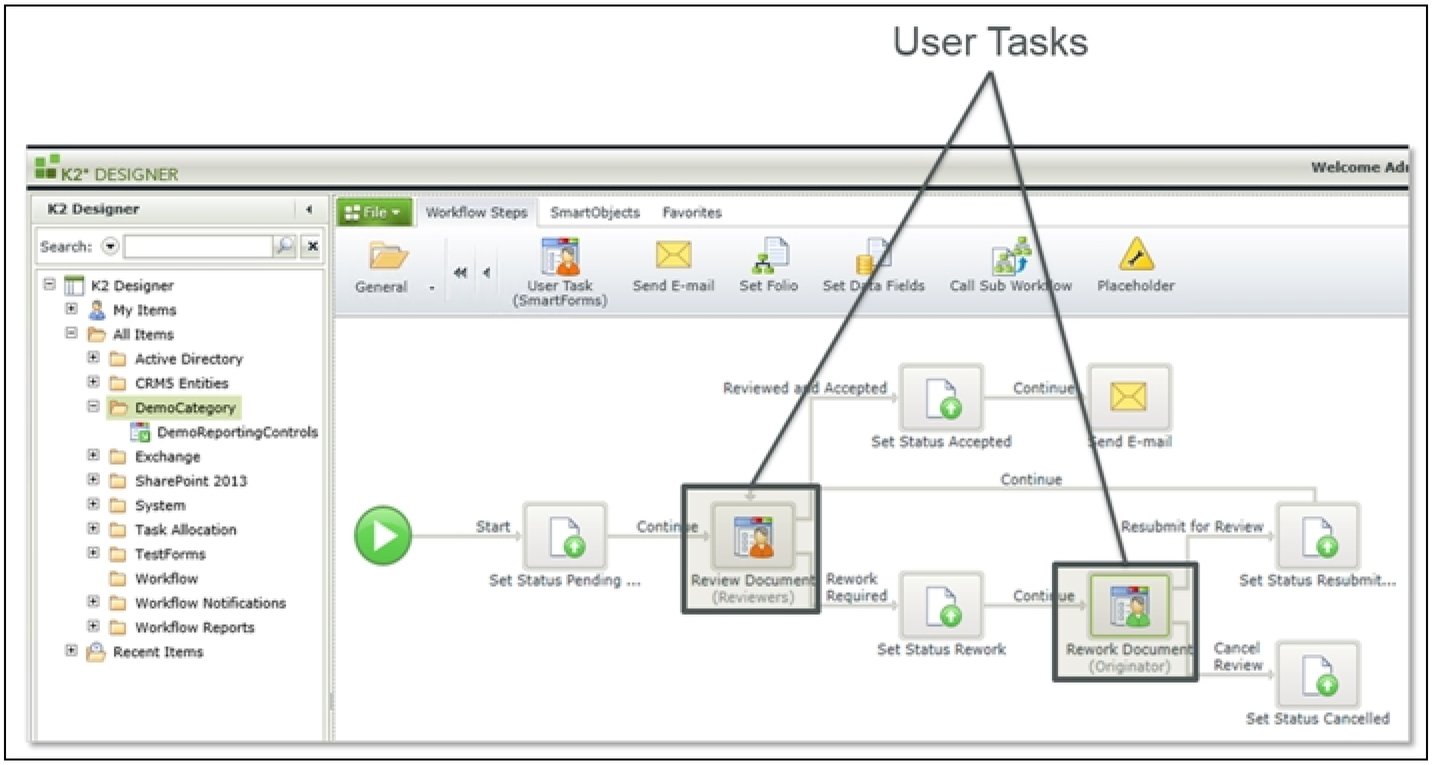


A User Task (also known as a Client Event) is a task that is performed by a human. All tasks that are performed by humans must be allocated to a Participant (also known as a Destination), because a human must perform the task.

In the diagram below, two user tasks have been highlighted. The task with the green user icon is a task that will be performed by the Originator of the process (the person who started the process initially). The task with the red user icon is a task that will be assigned to a Participant, in this case a reviewer for the document.

From a K2 Designer perspective, Forms can be used as the user interfaces for user tasks in a Workflow and can be used to start a Workflow. You merely need to point the Workflow designer to the Form you want to use for a user task, and the designer will do the rest. You can use separate Forms for each step of the Workflow, or make use of States and re-use the same Form for multiple steps in the Workflow.

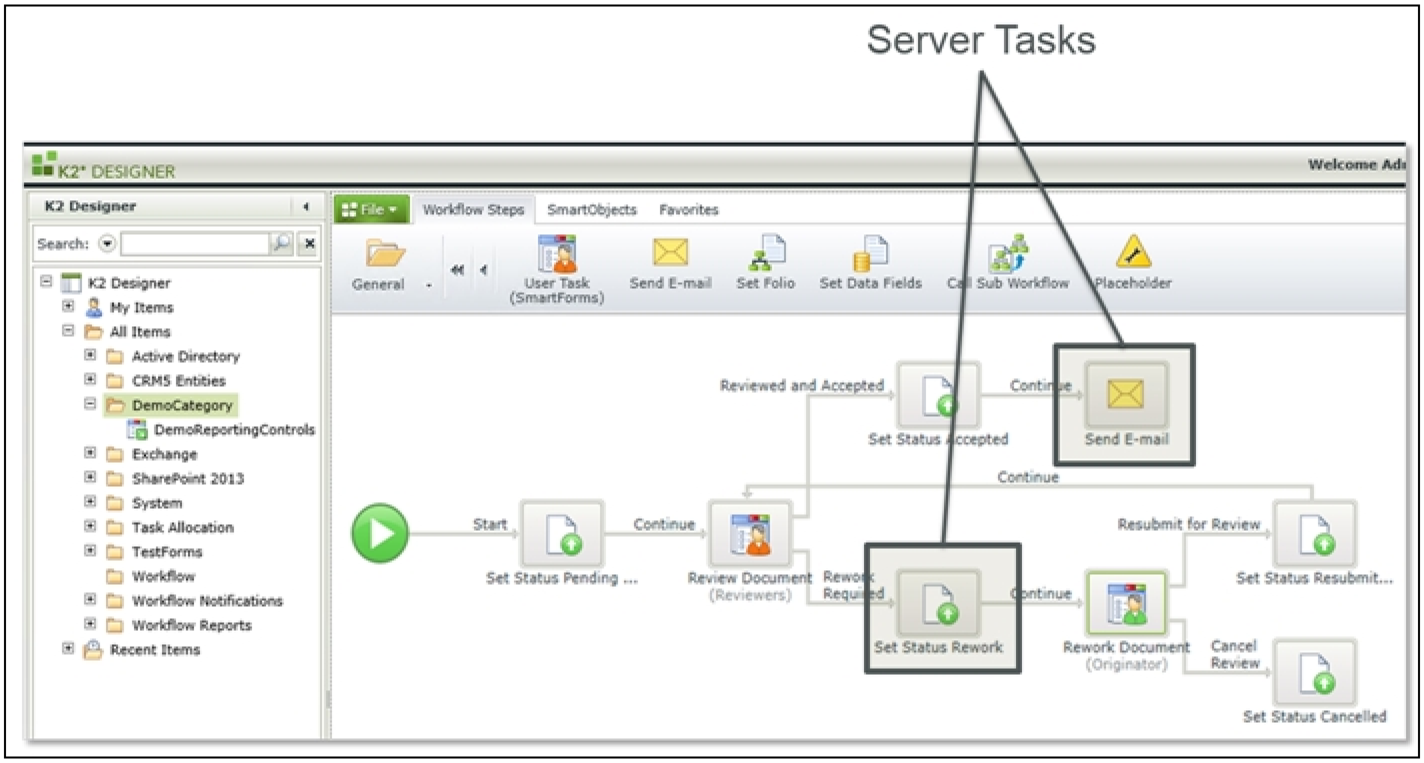
User tasks in a Workflow are indicated with user icons



Server tasks are tasks that are performed by the K2 server.

In the diagram below, two server tasks have been highlighted: the task with an envelope icon represents an E-mail that the server will send, while the task with a page-arrow icon represents a task where the K2 server is executing a SmartObject method to update a database with some value.

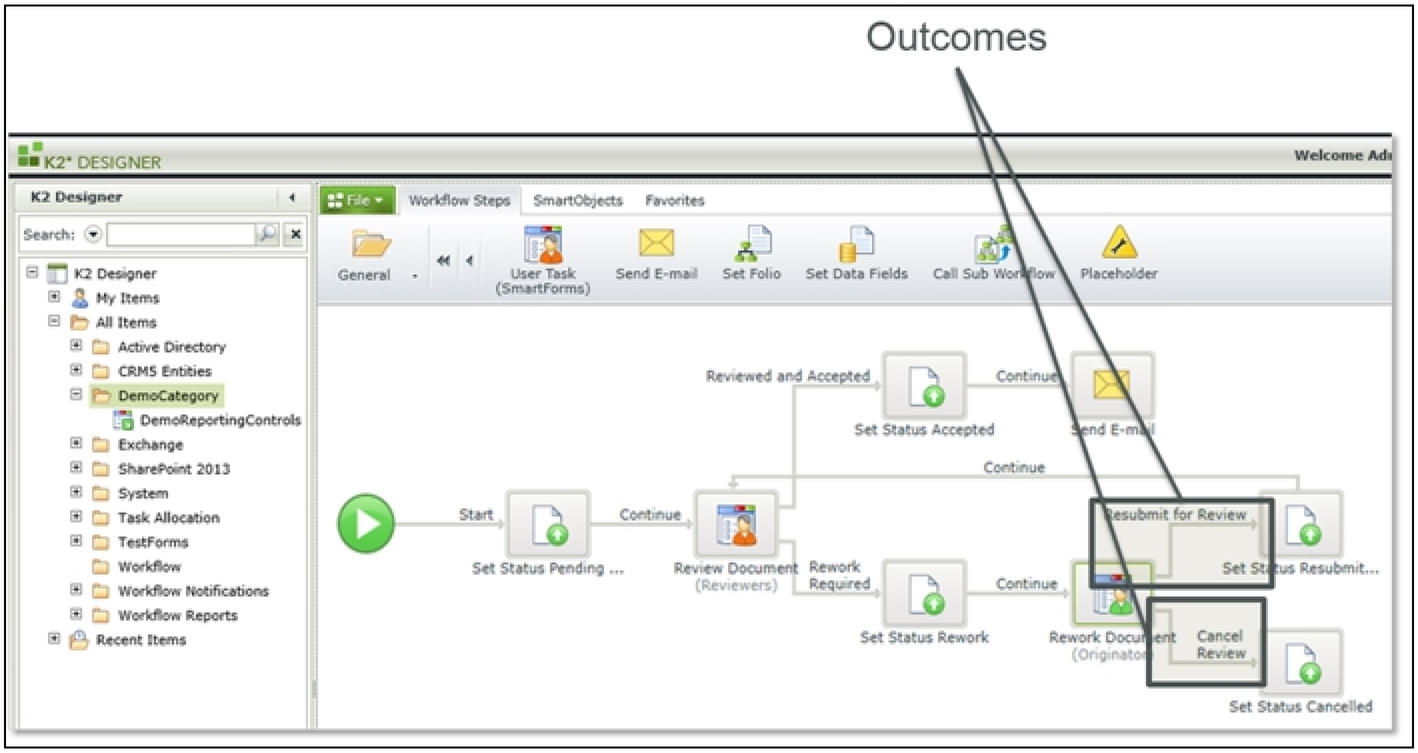
Server tasks in a Workflow



Outcomes (also known as Lines) are the possible paths in a Workflow. Outcomes may be based on user decisions (Actions), based on some comparison of values performed by the K2 server, or even based on a combination of the two. Sometimes an outcome is simply a "Continue" outcome where the path is always followed.

In the diagram below, two outcomes from the same task have been highlighted. The “Resubmit for Review” outcome is executed when the originator has finished re-working their request, and the "Cancel Review" outcome is followed when the user decides to cancel their request. While user actions are often associated with outcomes, an outcome does not have to depend on a user's decision. A step in the Workflow could also have multiple outcomes (for example one for each of the three possible choices a user could make on a task) and could even have parallel outcomes (in other words, the Workflow will split into parallel paths that are executed at the same time).

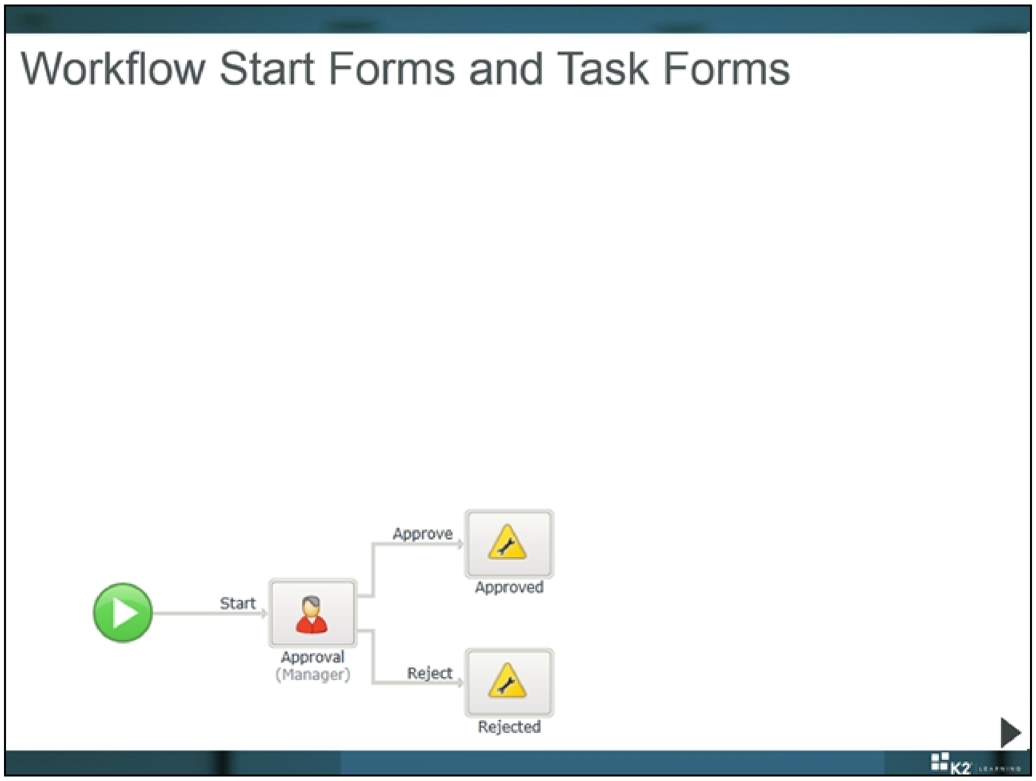
Outcomes are the "lines" in a Workflow



Summary

- The design canvas is where you “lay out” your Workflow
- The web-based Workflow designer does not offer free-form placement of steps (you cannot drag-and-drop steps around in the Workflow)
- The step wizards are drag-and-drop “events” or “steps” that you can drop into the Workflow and then configure according to the desired functionality
- Click on the category selector to pick a different set of wizards
- A process (also known as a Workflow) refers to the entire process and will have at least a Start step and one User Task or Server Step, but usually contains multiple steps.
- Steps are joined with Lines (also called Outcomes) which define the "flow" of the Workflow
- Outcomes may or may not be bound to user actions

Workflow Start Forms and Task Forms



Video

The integration between Forms and Workflows boils down to using Rules and Actions to start Workflows and open/-complete workflow tasks. When using SmartForms with Workflows, it is helpful to understand the relationship between the Workflow and the Form being used to interact with the Workflow. For example, you might use one Form to start a Workflow, but a completely different Form to open and complete a workflow task. Or you may want to use the same Form to both start the Workflow and complete a task in the Workflow, but use different States in the Form to determine whether the Form is starting the Workflow or completing a task.

There are separate Rule Actions to interact with Workflows, depending on what you want to do with the Rule. It is easiest to think in terms of Start Forms and Task Forms. Start Forms use a single action to start a Workflow, while Task Forms use two actions: one to open the worklist item based on a Serial Number when the Form is opened, and another action to complete the current task when something happens on the Form.

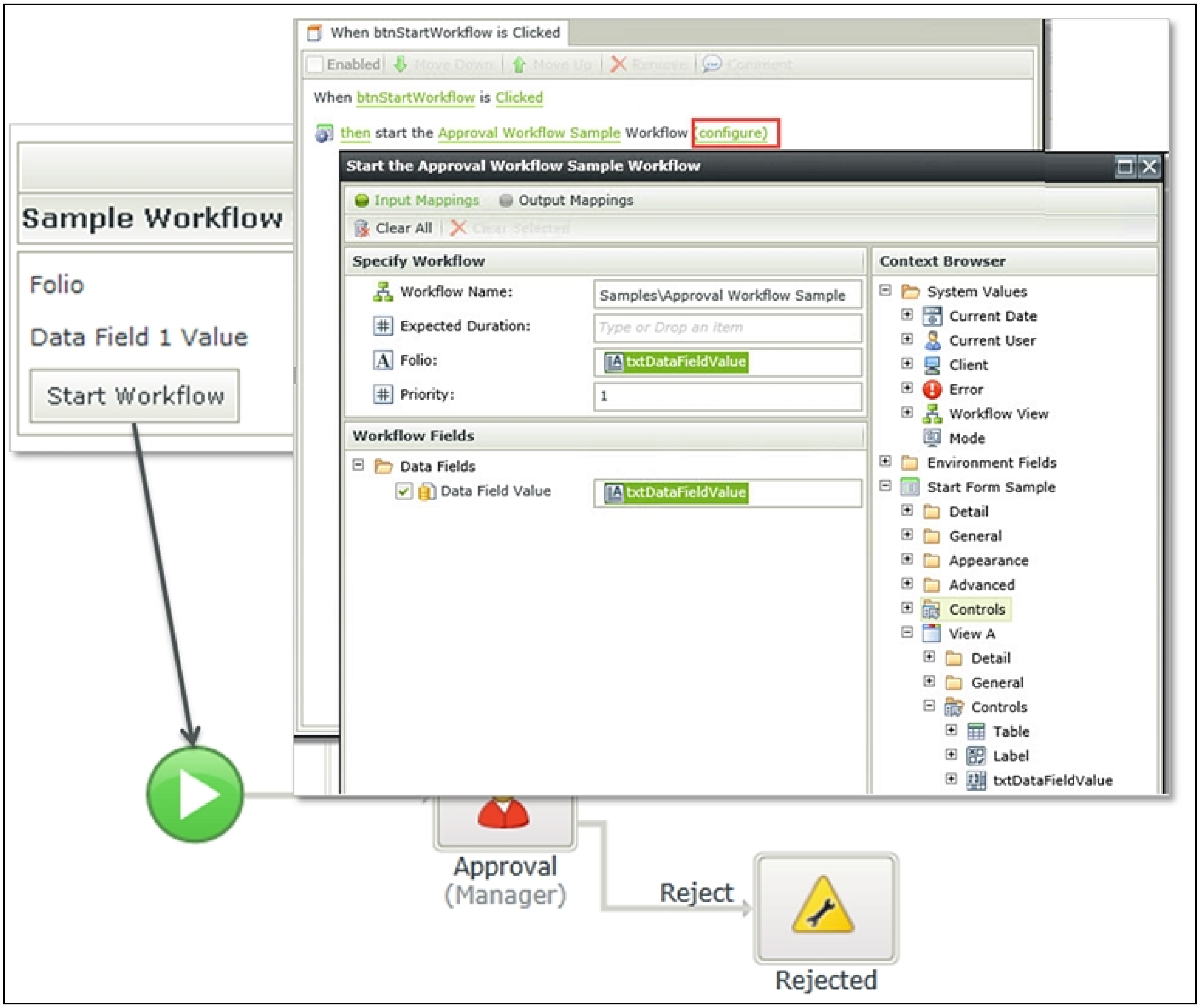
Start Forms

Start Forms use the Start a Workflow action to start a Workflow. This action is usually contained in a Rule for a Button Click event or some other user-driven event. In the example below, the Start Workflow button’s Rule calls the Start Workflow action to start the sample Workflow.

The Start Form is typically opened and completed by a user before the Workflow even exists: the Workflow is only started when the user takes some action on the Form. These Start Forms are usually “capture Forms” or “user input Forms” where the user will complete the Form with information before submitting the Form. In the majority of cases, the Submit action will save the data to some data store by calling a SmartObject method, and then start a Workflow in the same Rule. While these input Forms are normally purpose-built for Workflows, remember that you can re-use Views on the Form to save development time; you can also use Rule conditions to start Workflows based on some condition.

The Start Form would use a Rule calling the "Start a Workflow action" to initiate the Workflow. As part of the configuration of this action, you can set the Folio of the process that will start, set any data fields in the process, set the priority of the process instance, and the expected duration.

A Form Rule used to start a Workflow



When btnStartWorkflow is Clicked

then start the Approval Workflow Sample Workflow (configure)

Start the Approval Workflow Sample Workflow

Specify Workflow

Workflow Name:	Samples\Approval Workflow Sample
Expected Duration:	Type or Drop an item
Folio:	txtDataFieldValue
Priority:	1

Workflow Fields

Data Fields

Data Field Value	txtDataFieldValue

Task Forms

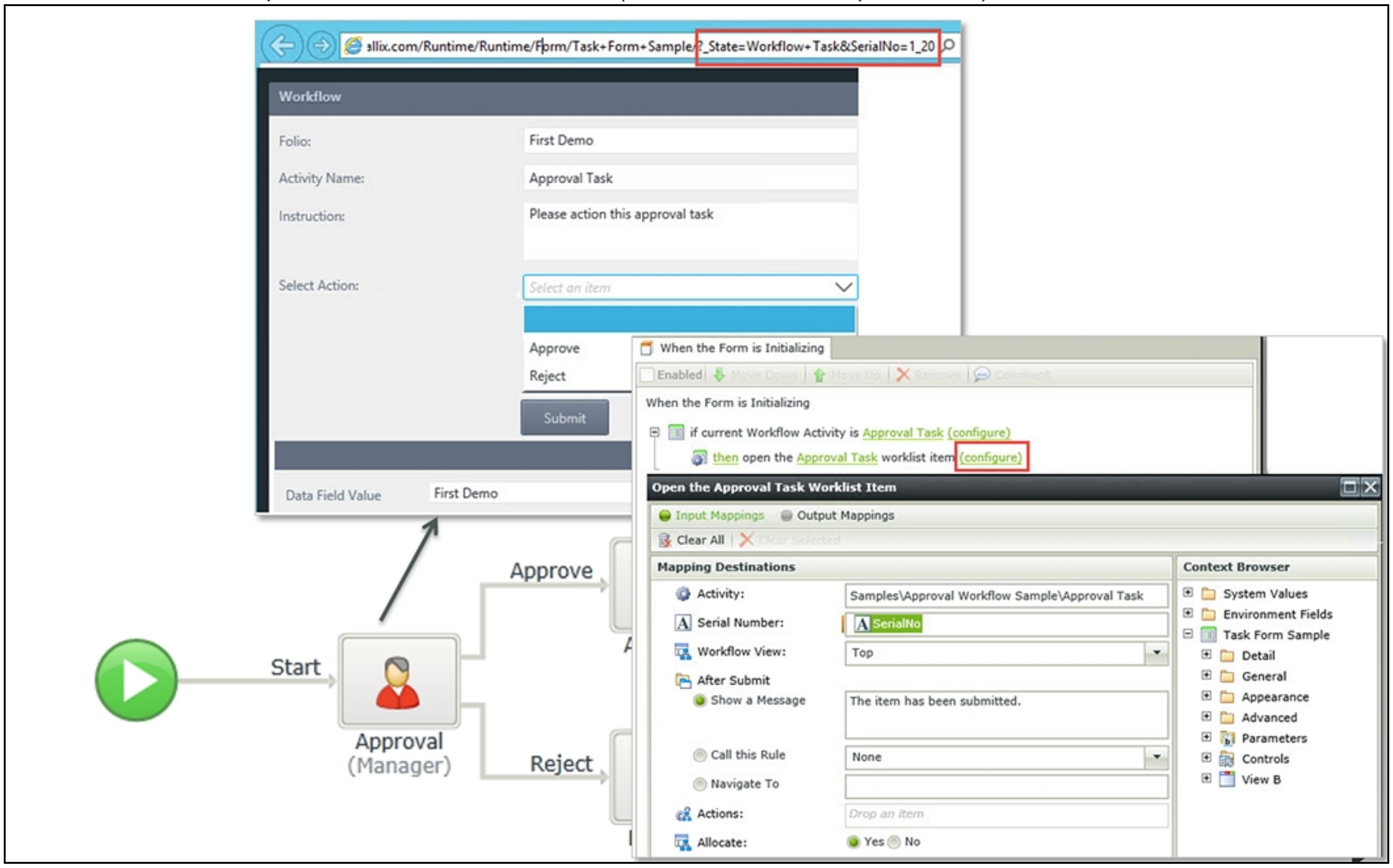
A Form used to complete a workflow task (also known as a worklist item) usually contains two workflow-specific actions: one action opens the workflow task when the Form is initialized, and another action completes the workflow task when the user takes an action on the Form. (Remember that actions are contained in Rules, and these Rules would be configured to fire at the appropriate times.)

The first step is to open the associated task with the "Open a worklist item" action, using the Serial Number to locate and open the workflow task. This rule is usually executed when the Form is initialized, and would usually also read the available actions for the task so that these can be presented to the user in some way, normally a drop-down menu. You can also read data from the Workflow instance, like the Folio or data field values.

This next image shows that when the Form associated with the Approval step is opened, then open the worklist item for this task. When configuring this Rule, you will have to provide the Serial Number for the task: this Serial Number is a unique value that is usually passed as a Form parameter. We have highlighted the Serial Number in the screenshot below to show where is it usually found.

Note
By default, opening a worklist item allocates the item to this user and they either need to complete the task or release it.

A Form Rule used to open a User Task worklist item (notice the SerialNo parameter)


allix.com/Runtime/Runtime/Form/Task+Form+Sample/?_State=Workflow+Task&SerialNo=1_20

Workflow

Folio:	First Demo
Activity Name:	Approval Task
Instruction:	Please action this approval task
Select Action:	Select an item
Data Field Value	First Demo

When the Form is Initializing

When the Form is Initializing
if current Workflow Activity is Approval Task (configure)
then open the Approval Task worklist item (configure)

Open the Approval Task Worklist Item

Mapping Destinations

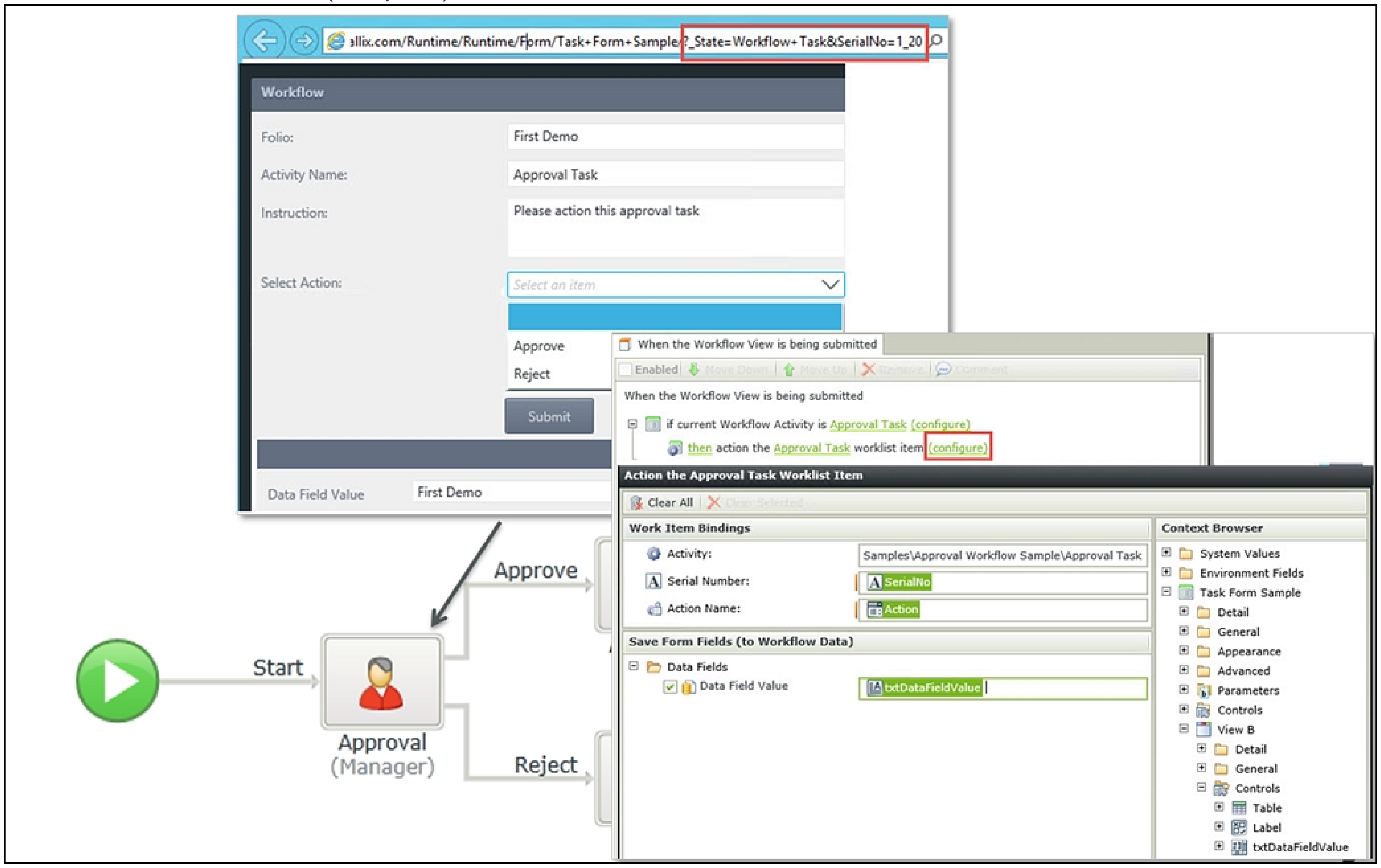
Activity:	Samples\Approval Workflow Sample\Approval Task
Serial Number:	SerialNo
Workflow View:	Top
After Submit	
Show a Message	The item has been submitted.
Call this Rule	None
Navigate To	
Actions:	Drop an item
Allocate:	[✓ Yes] No

Context Browser

The second part is executed when the user wants to complete the task, normally with a button click event. This part would use the "Action a Worklist Item" action to complete the workflow task. You would normally pass through the action that the user selected as part of the action configuration so that the Workflow knows which way to go next. As part of the action, you can also update the process Folio and process data field values.

Caution
The Action name used in the configuration for this step must exactly match the Action name defined in the Work-flow, otherwise the Action will not complete successfully.

A Form Rule used to action (complete) a User Task worklist item


allix.com/Runtime/Runtime/Form/Task+Form+Sample/?_State=Workflow+Task&SerialNo=1_20

Workflow

Folio: First Demo

Activity Name: Approval Task

Instruction: Please action this approval task

Select Action: Select an item

Approve

Reject

Submit

Data Field Value First Demo

When the Workflow View is being submitted

Enabled | Move Down | Move Up | Remove | Comment

When the Workflow View is being submitted

if current Workflow Activity is Approval Task (configure)

then action the Approval Task worklist item (configure)

Action the Approval Task Worklist Item

Clear All | Clear Selected

Work Item Bindings

Activity: Samples\Approval Workflow Sample\Approval Task

Serial Number: SerialNo

Action Name: Action

Save Form Fields (to Workflow Data)

Data Fields

Data Field Value: txtDataFieldValue

Context Browser

System Values

Environment Fields

Task Form Sample

Detail

General

Appearance

Advanced

Parameters

Controls

View B

Detail

General

Controls

Table

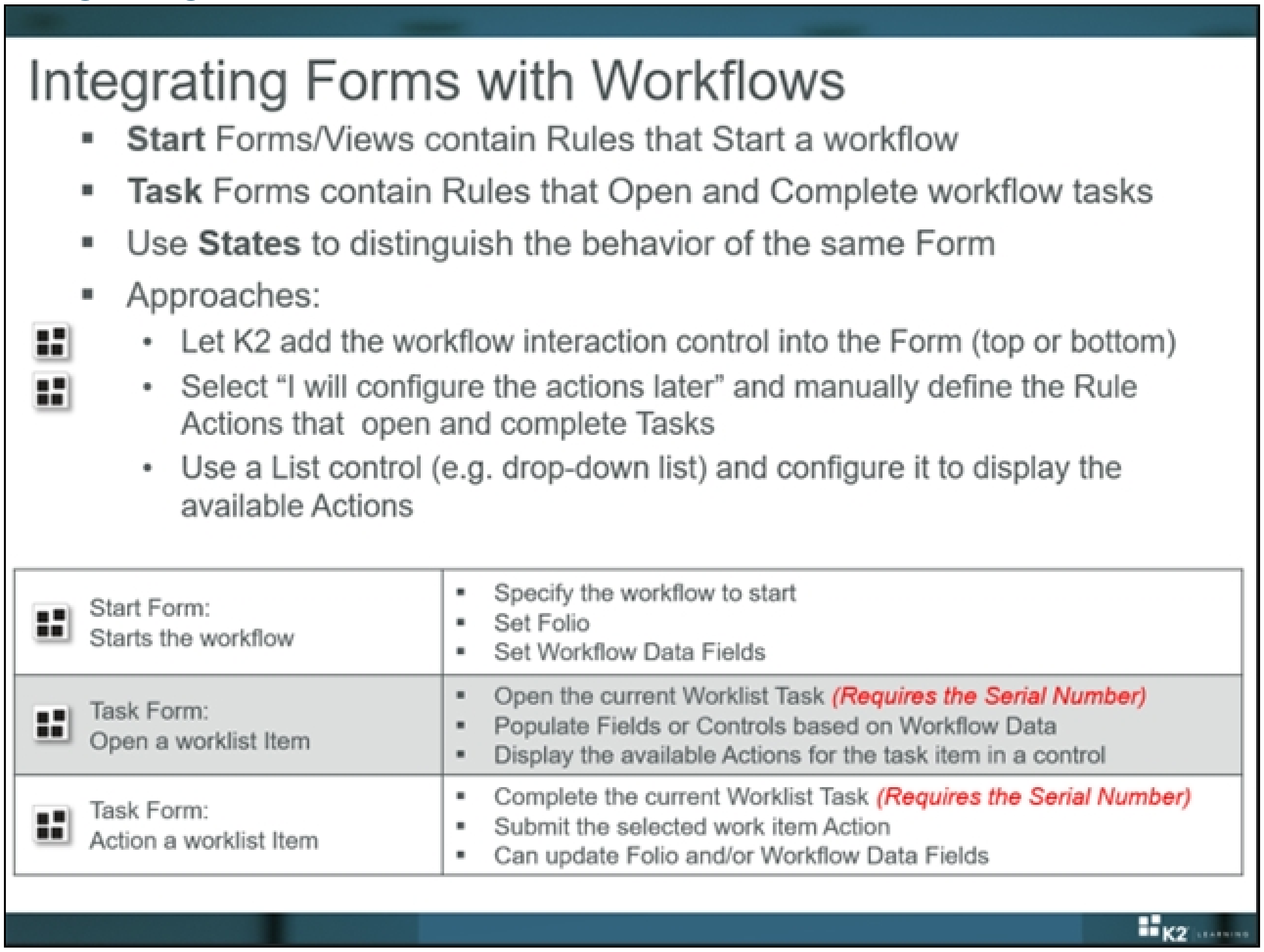
Label

txtDataFieldValue

Summary

- There are different Rule Actions to interact with Workflows:
    - Actions to start Workflows
    - Actions to open a worklist item (requires the task Serial Number, usually a parameter in the query string)
    - Actions to complete a worklist item (requires the Serial Number)
- You can use the same Form to both start a Workflow and complete a user task later on in that Workflow
    - Usually this is achieved with States and changing the behavior (commonly through Rules) of the Form based on the State

Integrating Forms with Workflows



Video

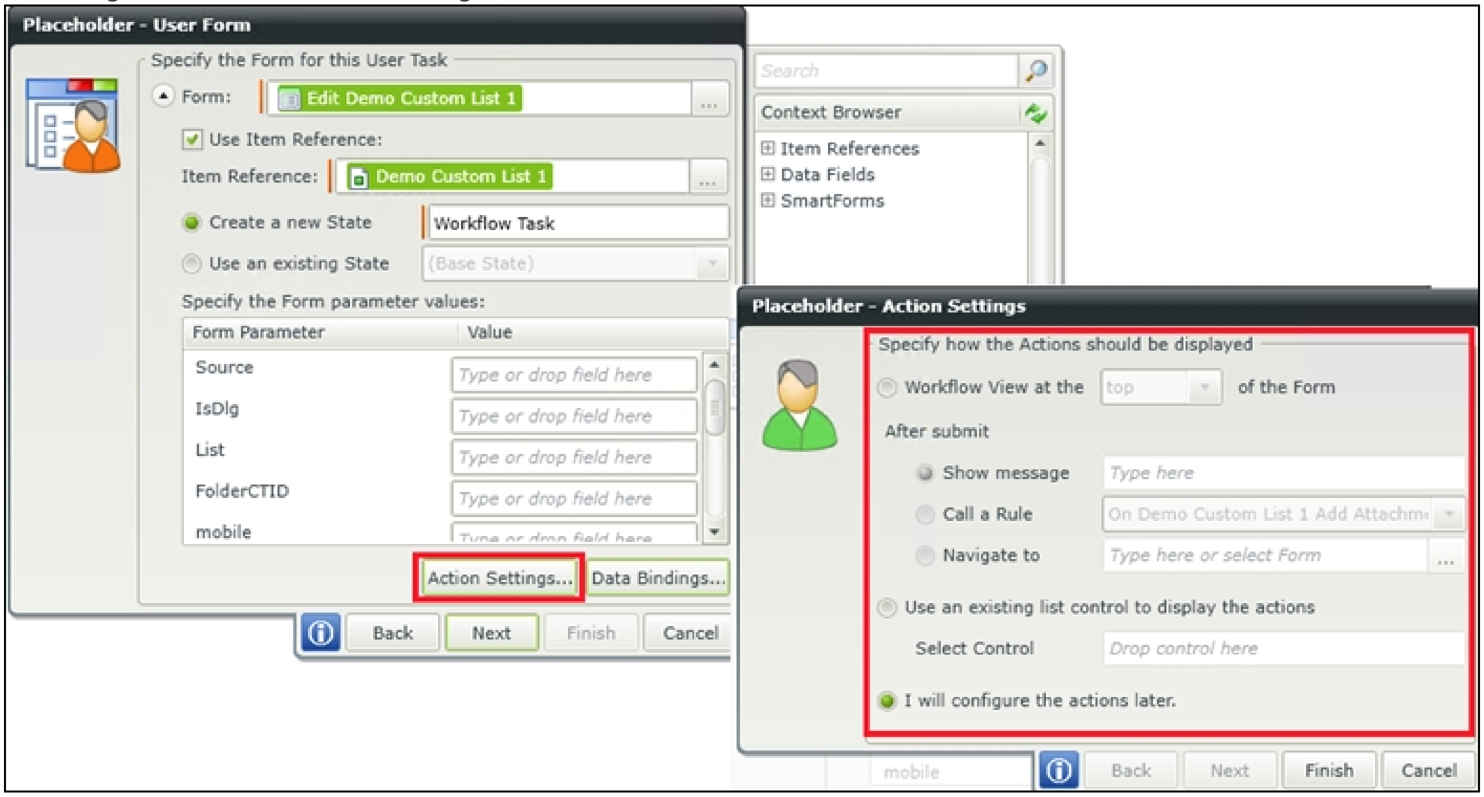
Integrating Forms with Workflows

- **Start** Forms/Views contain Rules that Start a workflow
- **Task** Forms contain Rules that Open and Complete workflow tasks
- Use **States** to distinguish the behavior of the same Form
- Approaches:
  - Let K2 add the workflow interaction control into the Form (top or bottom)
  - Select "I will configure the actions later" and manually define the Rule Actions that open and complete Tasks
  - Use a List control (e.g. drop-down list) and configure it to display the available Actions

Start Form: Starts the workflow	▪ Specify the workflow to start ▪ Set Folio ▪ Set Workflow Data Fields
Task Form: Open a worklist Item	▪ Open the current Worklist Task *(Requires the Serial Number)* ▪ Populate Fields or Controls based on Workflow Data ▪ Display the available Actions for the task item in a control
Task Form: Action a worklist Item	▪ Complete the current Worklist Task *(Requires the Serial Number)* ▪ Submit the selected work item Action ▪ Can update Folio and/or Workflow Data Fields

In K2, there several ways of configuring the relationship and interaction between SmartForms and workflows. From the previous topic you should recall that you can think of workflow-enabled Forms in terms of Start Forms or Task Forms. It is quite common that the Rules that contain the workflow integration are specific in different states of the Form, especially when you want to use the same Form to start the workflow and in user tasks, or when using the same Form for different user tasks in the workflow.

Selecting how the workflow task integration should be added to the Form



Placeholder - User Form

Specify the Form for this User Task

Form: Edit Demo Custom List 1

[✓] Use Item Reference:

Item Reference: Demo Custom List 1

(●) Create a new State: Workflow Task

( ) Use an existing State: (Base State)

Specify the Form parameter values:

Form Parameter	Value
Source	Type or drop field here
IsDlg	Type or drop field here
List	Type or drop field here
FolderCTID	Type or drop field here
mobile	Type or drop field here

Placeholder - Action Settings

Specify how the Actions should be displayed

( ) Workflow View at the top of the Form

After submit

( ) Show message: Type here

( ) Call a Rule: On Demo Custom List 1 Add Attachm

( ) Navigate to: Type here or select Form

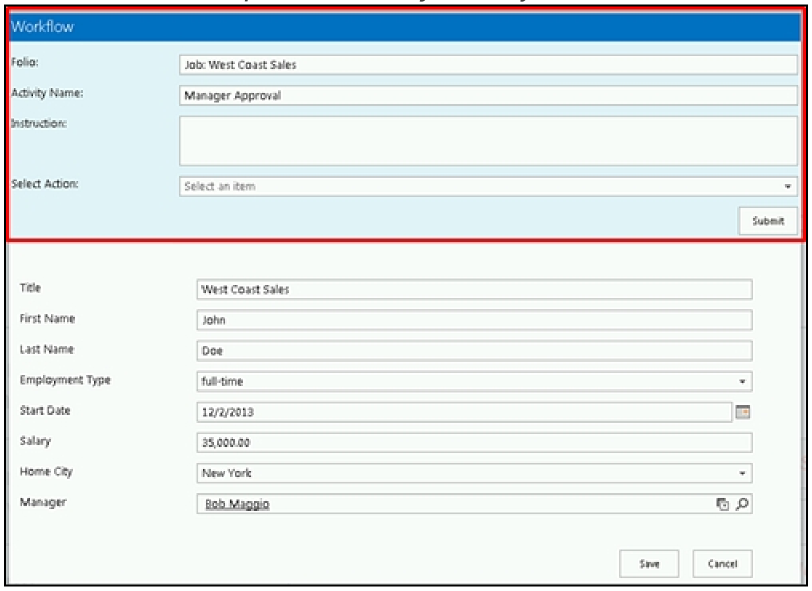
( ) Use an existing list control to display the actions

Select Control: Drop control here

(●) I will configure the actions later.

There are three basic approaches to configuring the Start and Task Forms for your workflow:

1. Let K2 add all the workflow integration for you. This is the easiest option but you do not have much control over the integration. If you select this option, K2 will "inject" the Workflow View panel into the Form.
   The Workflow task panel that is injected by K2

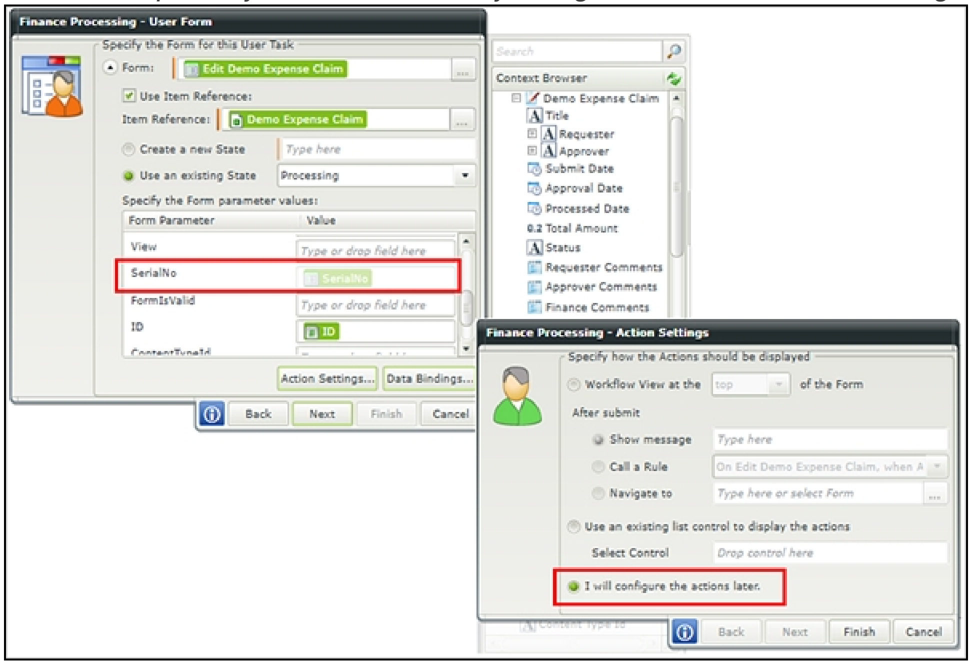
   Workflow

Folio:	Job: West Coast Sales
Activity Name:	Manager Approval
Instruction:	
Select Action:	Select an item

Title	West Coast Sales
First Name	John
Last Name	Doe
Employment Type	full-time
Start Date	12/2/2013
Salary	35,000.00
Home City	New York
Manager	Bob Maggio

2. Let K2 add the integration, but use a custom control to show the available actions. This is also an easy option, and you can decide where the actions should be shown to the user (usually, a drop-down list control) and which Rule the Workflow Actions should be added to. In this case, you will need to add a list control like a drop-down list onto the Form and then configure the workflow to populate the drop-down list with the available actions for the task.
3. Configure the workflow integration yourself by manually editing the Rules and Actions on the Form. This is the most flexible option and allows you to have complete control over the workflow integration. Once you are more comfortable with Rules and Actions and Forms, you may prefer to use this option.

Select this option if you want to manually configure the Form-Workflow integration


Finance Processing - User Form

Specify the Form for this User Task

Form: Edit Demo Expense Claim

Use Item Reference: ✓

Item Reference: Demo Expense Claim

Create a new State: Type here

Use an existing State: Processing

Specify the Form parameter values:

Form Parameter	Value
View	Type or drop field here
SerialNo	SerialNo
FormIsValid	Type or drop field here
ID	ID

Context Browser

Demo Expense Claim: Title, Requester, Approver, Submit Date, Approval Date, Processed Date, Total Amount, Status, Requester Comments, Approver Comments, Finance Comments

Finance Processing - Action Settings

Specify how the Actions should be displayed

Workflow View at the top of the Form

After submit

Show message: Type here

Call a Rule: On Edit Demo Expense Claim, when A

Navigate to: Type here or select Form

Use an existing list control to display the actions

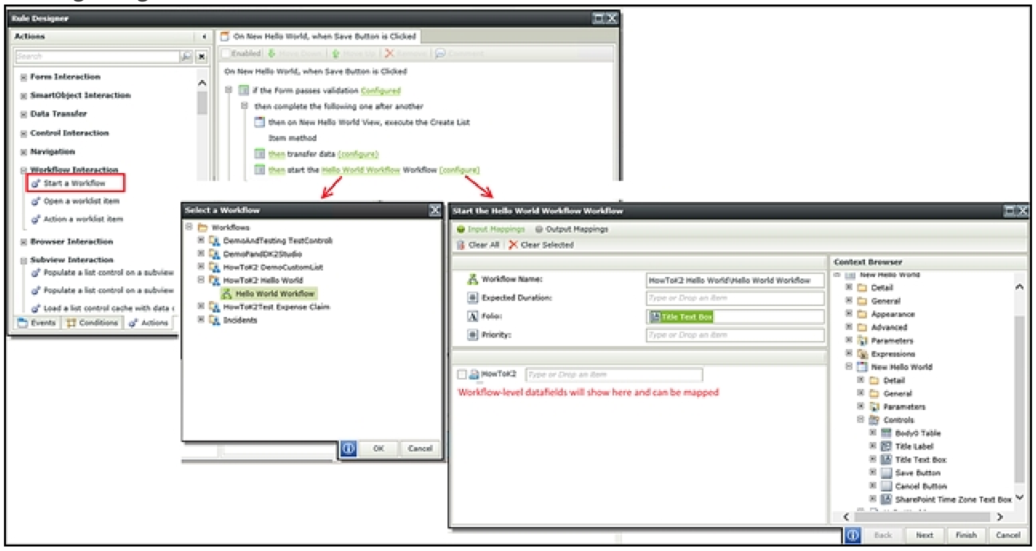
Select Control: Drop control here

[✓ I will configure the actions later.]

When you select one of the first two options, K2 will "inject" the Rules into the Forms when the workflow is deployed. When using the third option, you will need to define the Rule integration yourself.

If you want to configure the Workflow-Form integration manually, you will need to use appropriate Rule Actions in Rules, as shown below.

Configuring the Action that Starts the workflow.


Rule Designer

Actions: Form Interaction; SmartObject Interaction; Data Transfer; Control Interaction; Navigation; Workflow Interaction (Start a Workflow, Open a worklist item, Action a worklist item); Browser Interaction; Subview Interaction (Populate a list control on a subview, Populate a list control on a subview, Load a list control cache with data)

On New Hello World, when Save Button is Clicked

if the Form passes validation Configured

then complete the following one after another

then on New Hello World View, execute the Create List Item method

then transfer data (configure)

then start the Hello World Workflow Workflow (configure)

Select a Workflow

Workflows: DemoAndTesting TestControls; DemoFandDK2Studio; HowToK2 DemoCustomList; HowToK2 Hello World (Hello World Workflow); HowToK2Test Expense Claim; Incidents

Start the Hello World Workflow Workflow

Input Mappings | Output Mappings

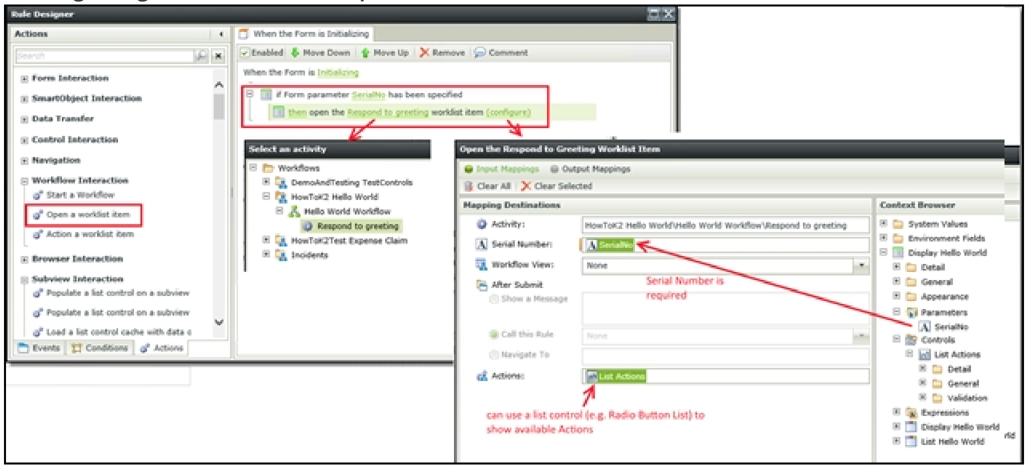
Workflow Name:	HowToK2 Hello World\Hello World Workflow
Expected Duration:	Type or Drop an item
Folio:	Title Text Box
Priority:	Type or Drop an item

HowToK2: Type or Drop an item

Workflow-level datafields will show here and can be mapped

Context Browser: New Hello World (Detail, General, Appearance, Advanced, Parameters, Expressions, New Hello World (Detail, General, Parameters, Controls (Body0 Table, Title Label, Title Text Box, Save Button, Cancel Button, SharePoint Time Zone Text Box)))

Configuring the Action that opens the worklist item. Notice that the Serial Number is required.



When the Form is Initializing

if Form parameter SerialNo has been specified

then open the Respond to greeting worklist item (configure)

Open the Respond to Greeting Worklist Item

Mapping Destinations

Activity: HowToK2 Hello World\Hello World Workflow\Respond to greeting

Serial Number: SerialNo

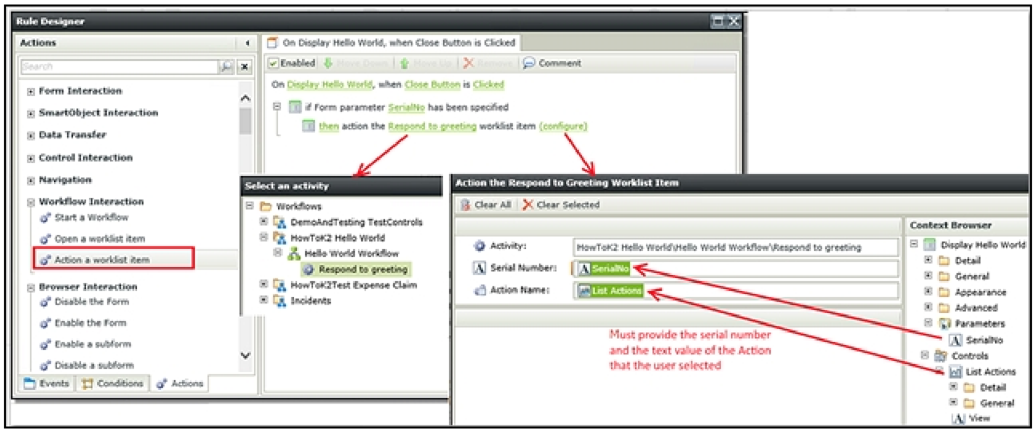
Workflow View: None

Serial Number is required

Actions: List Actions

can use a list control (e.g. Radio Button List) to show available Actions

Configuring the Action that completes the task. Notice that the Serial Number is required, as is the name of the Action to execute



On Display Hello World, when Close Button is Clicked

if Form parameter SerialNo has been specified

then action the Respond to greeting worklist item (configure)

Action the Respond to Greeting Worklist Item

Activity: HowToK2 Hello World\Hello World Workflow\Respond to greeting

Serial Number: SerialNo

Action Name: List Actions

Must provide the serial number and the text value of the Action that the user selected

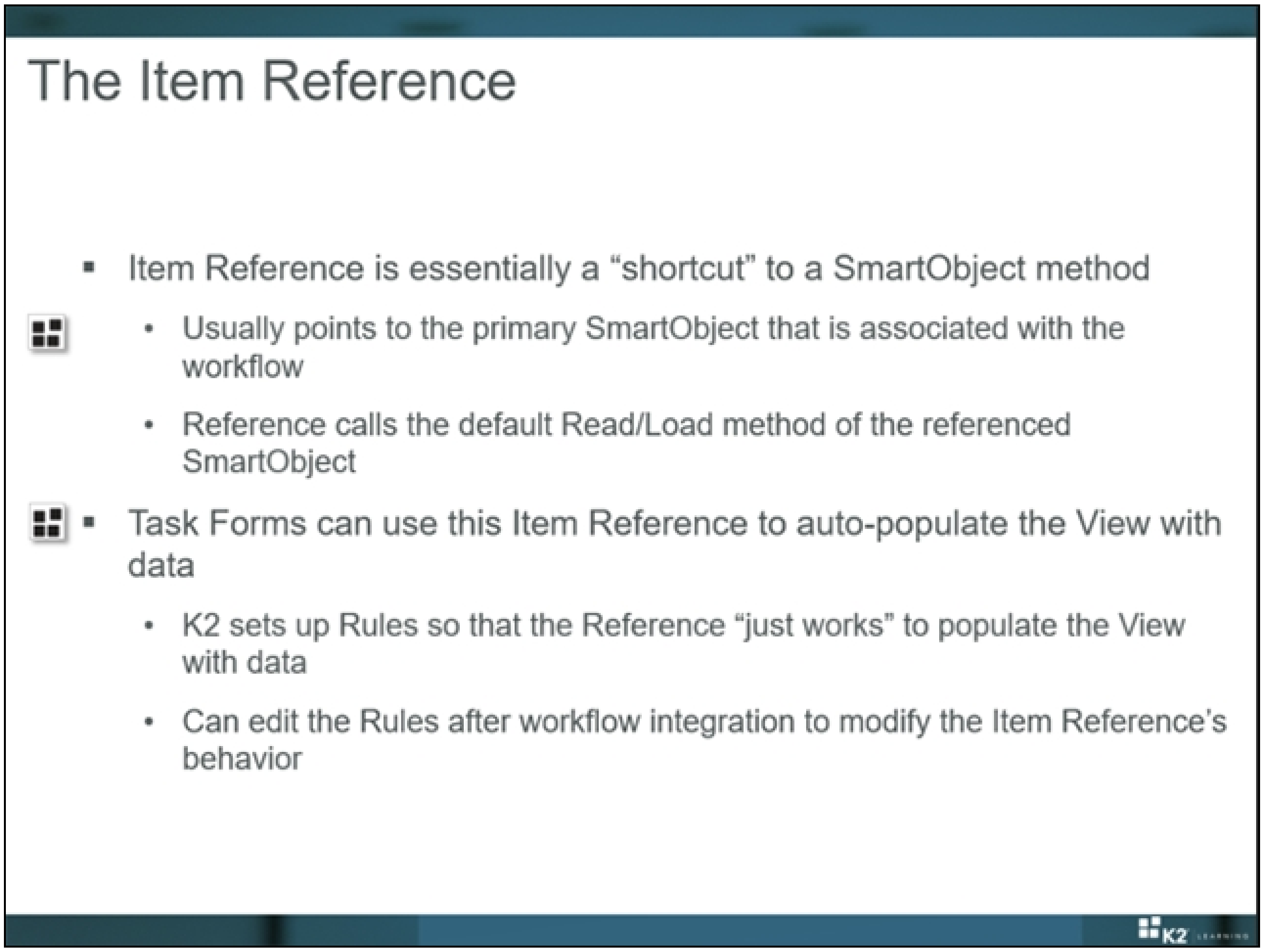
Caution
When configuring the Action that completes/actions a user task, note that the value of the **Action Name** must exactly match the Actions as they are defined in the workflow, otherwise the workflow step will not complete as expected.

Tip
Another approach for using Forms for User tasks is to use the Default User Task wizard, configure the step to use the Form URL and then set up the Workflow integration Rules manually. We will explore this option in more detail in the next module.

Summary

- The Start Workflow Action will allow you select a Workflow to start, set the Folio, then the Workflow Data Fields as part of the wizard.
- The Open Workflow Task Action wizard will allow you to select a Form and then optionally pass data between the Workflow and the Form.
- There are 3 standard approaches to integrate Task Forms with a Workflow:
  - Use the standard K2 Workflow task pane (easiest since K2 does all the integration for you).
  - Use a list Control like a drop-down list on the Form (you must configure Rules on the Form to populate the list control with the task actions).
  - Configure the Rules later (this is the most complex approach but allows for most control and flexibility).
- Where required, K2 will “inject” Rules into the Forms when the Workflow is deployed to the K2 server.

The Item Reference



The Item Reference

- Item Reference is essentially a "shortcut" to a SmartObject method
  - Usually points to the primary SmartObject that is associated with the workflow
  - Reference calls the default Read/Load method of the referenced SmartObject
- Task Forms can use this Item Reference to auto-populate the View with data
  - K2 sets up Rules so that the Reference "just works" to populate the View with data
  - Can edit the Rules after workflow integration to modify the Item Reference's behavior

Video

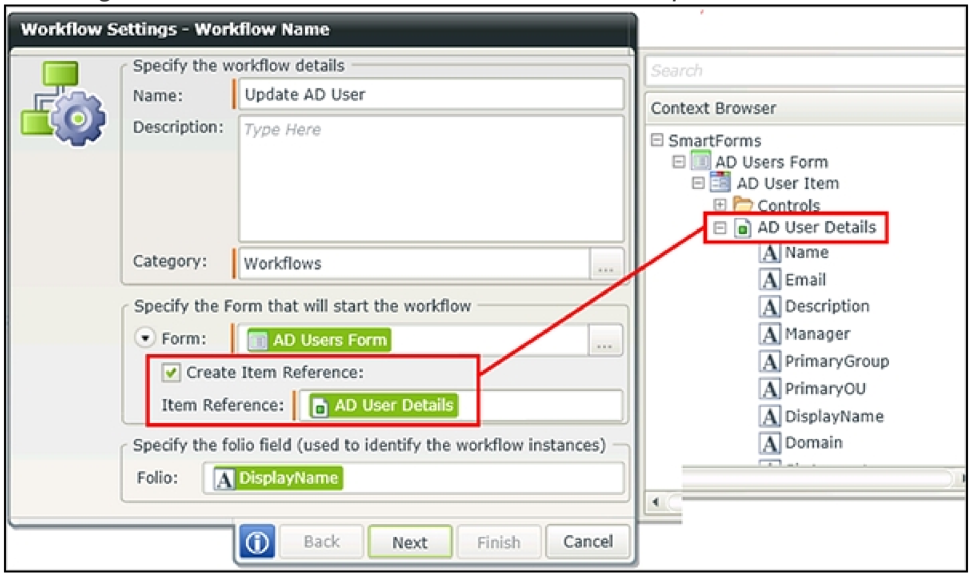
Remember that SmartForms can integrate with almost any data store through SmartObjects. This means you do not have to define workflow datafields to store the data entered by the user - instead, the data can be stored in some other system and then accessed through a SmartObject. Workflows can also use SmartObjects to interact with external systems, so therefore it should be easy to set up a Form-Workflow so that they both can interact with the "primary SmartObject" for that application. This is exactly what Item References can help with.

You can think of Item References as a "shortcut" to a SmartObject method that can load the details of the SmartObject associated with the application. For example, in a customer on-boarding application, perhaps the Customer.Read method is used to retrieve the details of the customer that is being on-boarded. Think of the customer record as the Primary SmartObject that is associated with the Forms and Workflow in the application. You can make use of Item References so that K2 will do all of the necessary configuration behind the scenes to load that customer record in the Workflow and the Forms used in the Workflow.

Note that you don't have to use Item References, because you can still define your own Rules for transferring data between Forms and the Workflow. Item References just make your life a little easier, because they can automatically set up Rules for Task Forms so that the Task Form is populated with data without you having to do any additional wiring-up, saving a lot of time. Also note that Item References are available in all the Workflow design tools provided by K2.

To make use of an Item Reference, select the **Create Item Reference** option in the Workflow Start wizard.

Creating an Item Reference in the Workflow Start Step



Workflow Settings - Workflow Name

Specify the workflow details

Name: Update AD User

Description: Type Here

Category: Workflows

Specify the Form that will start the workflow

Form: AD Users Form

☑ Create Item Reference:

Item Reference: AD User Details

Specify the folio field (used to identify the workflow instances)

Folio: DisplayName

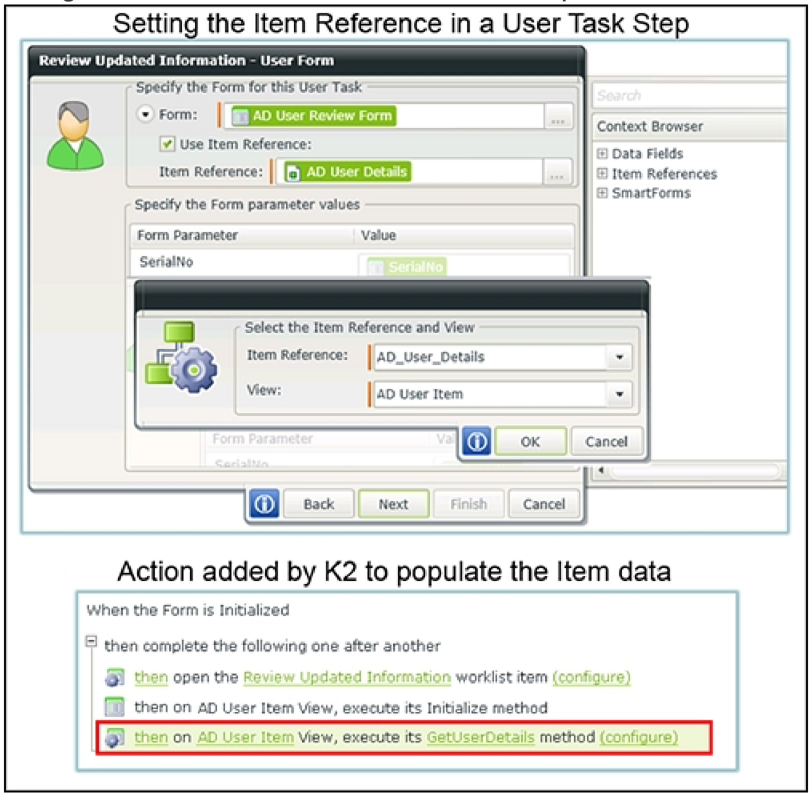
Search

Context Browser

- SmartForms
  - AD Users Form
    - AD User Item
      - Controls
      - AD User Details
        - Name
        - Email
        - Description
        - Manager
        - PrimaryGroup
        - PrimaryOU
        - DisplayName
        - Domain

Now that once the Item Reference exists, you can use it in the Forms for subsequent User Tasks. Behind the scenes, K2 will add the necessary Rule Actions to populate the Form with the data of the primary SmartObject record for this instance of the Workflow.

Using the Item Reference in a User Task Step



Setting the Item Reference in a User Task Step

Review Updated Information - User Form

Specify the Form for this User Task

Form: AD User Review Form

☑ Use Item Reference:

Item Reference: AD User Details

Specify the Form parameter values

Form Parameter	Value
SerialNo	SerialNo

Search

Context Browser

- Data Fields
- Item References
- SmartForms

Select the Item Reference and View

Item Reference: AD_User_Details

View: AD User Item

Action added by K2 to populate the Item data

When the Form is Initialized

then complete the following one after another

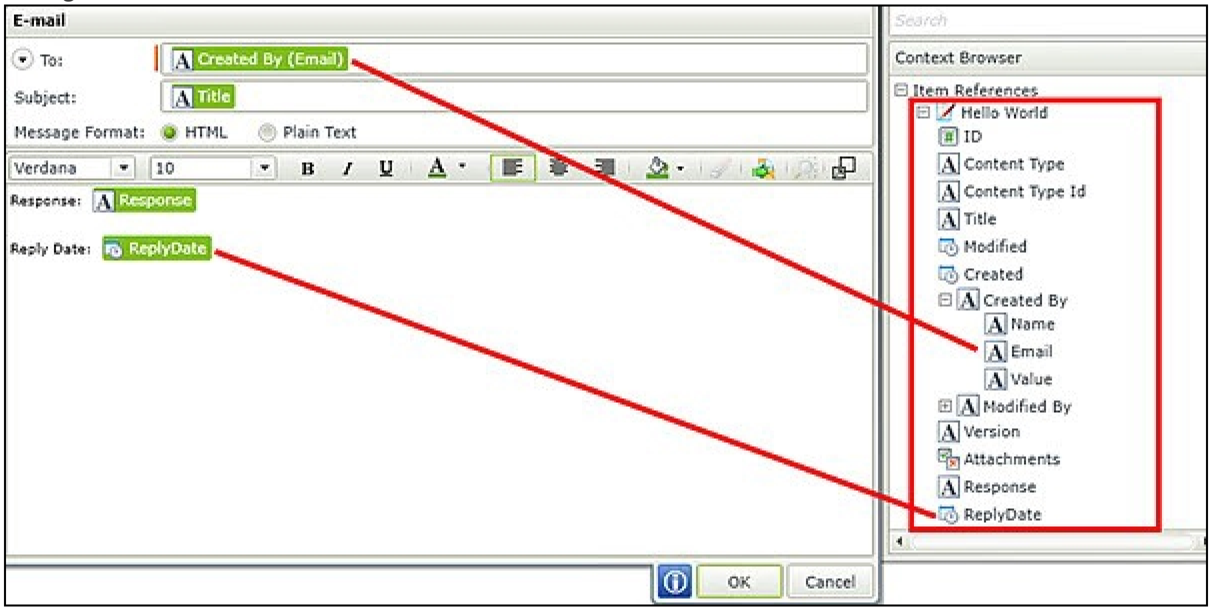
then open the Review Updated Information worklist item (configure)

then on AD User Item View, execute its Initialize method

then on AD User Item View, execute its GetUserDetails method (configure)

You can also easily use the Item Reference in other Workflow steps as well, for example when sending an email to the new on-boarded customer.

Using an Item Reference in a workflow e-mail wizard



E-mail

To: Created By (Email)

Subject: Title

Message Format: [✓ HTML] Plain Text

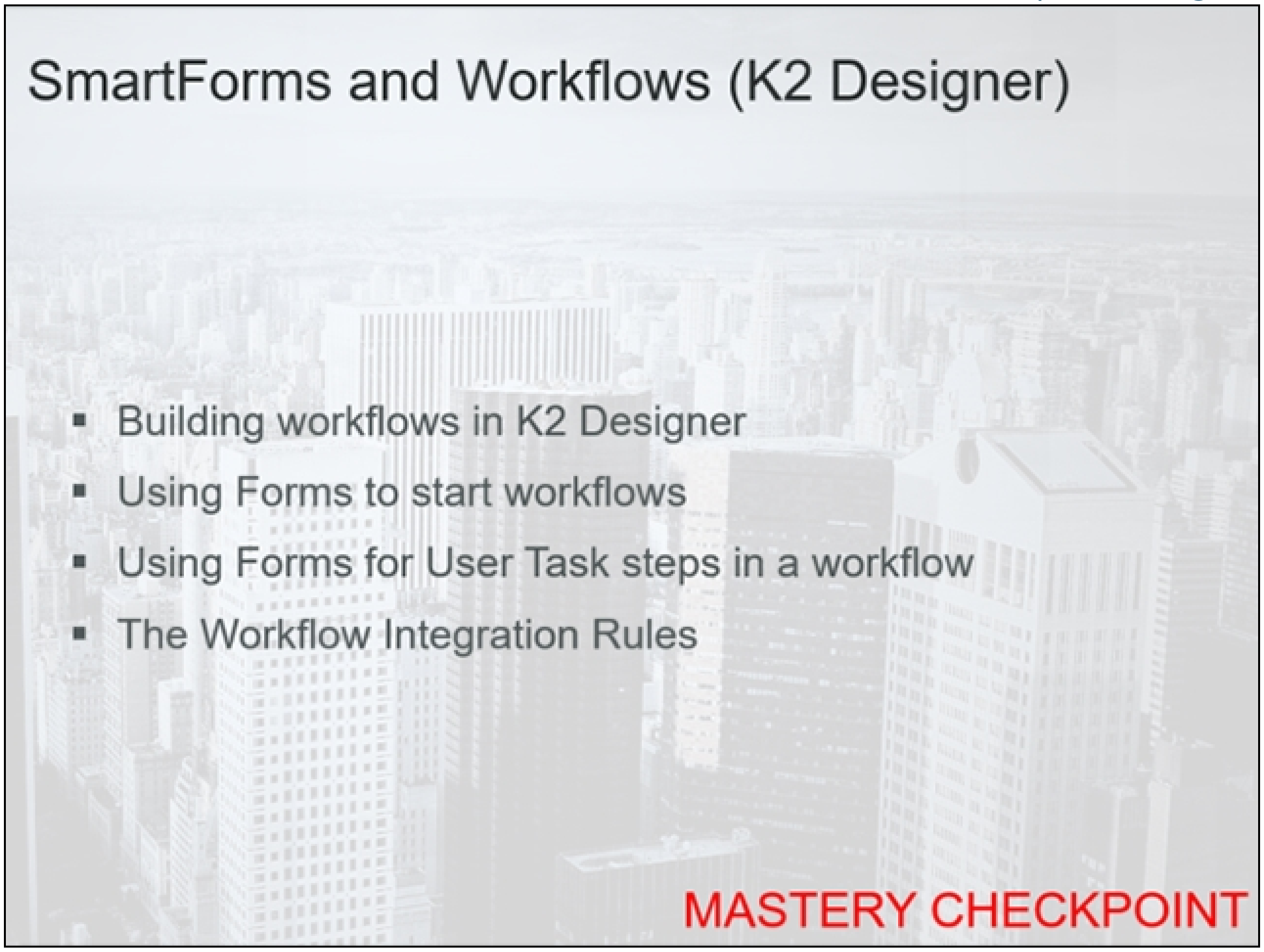
Response: Response

Reply Date: ReplyDate

Discussion Points:

- An Item Reference is a shortcut to the primary SmartObject record associated with a Workflow-and-Forms application.
- In the Workflow-Forms integration wizards, you can tell K2 what SmartObject and what SmartObject method it should use as the Item Reference for the application.
- The reference is set up to call the default Read (“Load”) method of the SmartObject using the unique ID for the record in question.
- Workflows and Forms can then use this Item Reference easily.

MASTERY CHECKPOINT: SmartForms and Workflows (K2 Designer)



SmartForms and Workflows (K2 Designer)

- Building workflows in K2 Designer
- Using Forms to start workflows
- Using Forms for User Task steps in a workflow
- The Workflow Integration Rules

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 4 of this module. If you are attending instructor-led training, use this slide as an opportunity to answer any questions around using SmartForms in K2 Designer-built Workflows.

By now you should know:

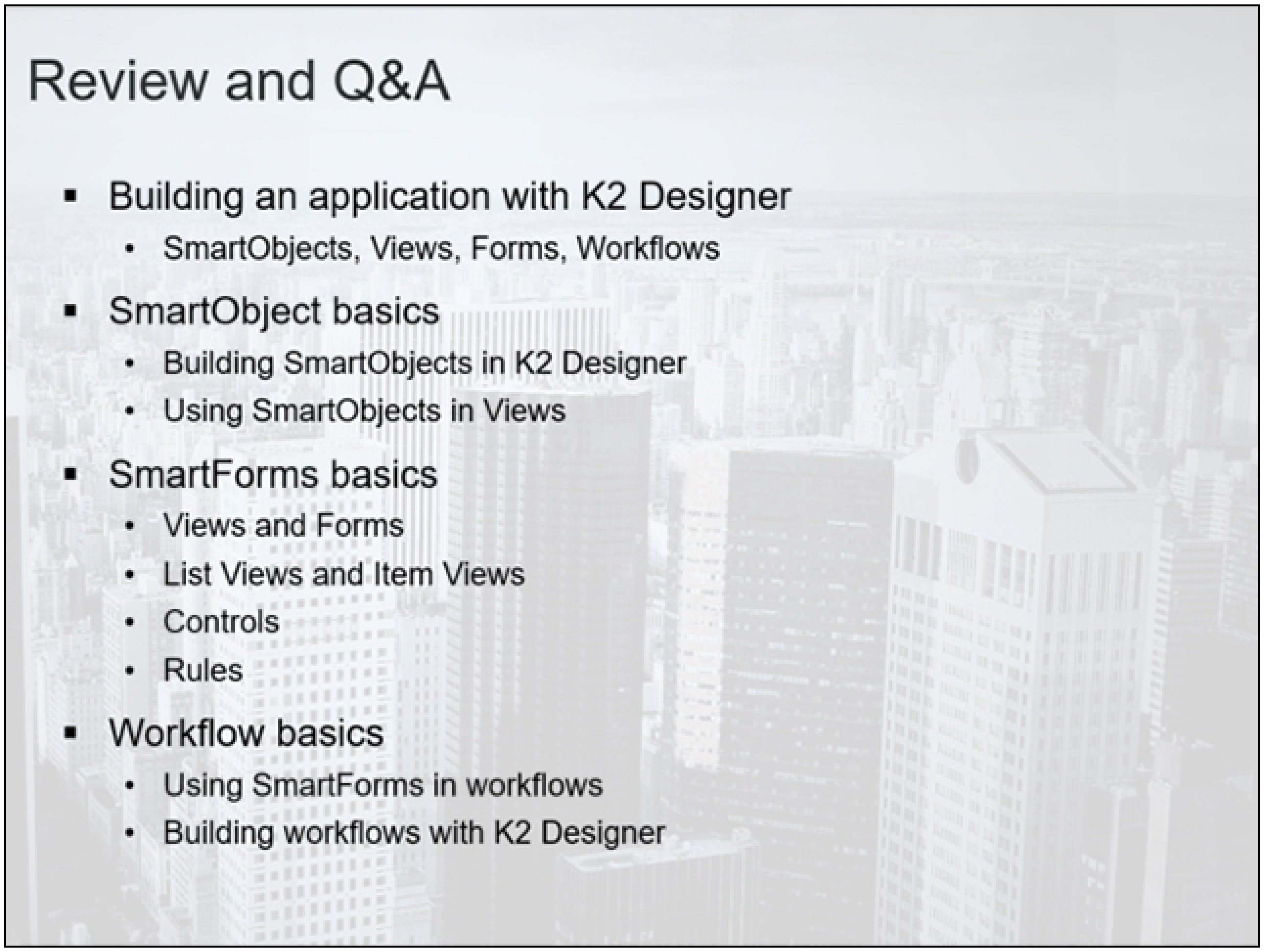
- How to create Workflows in K2 Designer
- How to use Forms to start Workflows
- How to use Forms as the User Interfaces for user steps in a Workflow
- The Rule Actions that make Forms-Workflow integration possible

Knowledge-check questions

Q: True or False: K2 applications always require a Workflow component.
Reveal answer**A:** False, applications do not have to have Workflows.

Q: Could you use the same Form to both start a Workflow and for a user task step in a Workflow?
Reveal answer**A:** Yes, this is typically achieved with States and Rules.

Review and Q&A



Review and Q&A

- Building an application with K2 Designer
  - SmartObjects, Views, Forms, Workflows
- SmartObject basics
  - Building SmartObjects in K2 Designer
  - Using SmartObjects in Views
- SmartForms basics
  - Views and Forms
  - List Views and Item Views
  - Controls
  - Rules
- Workflow basics
  - Using SmartForms in workflows
  - Building workflows with K2 Designer

Video

After completing this learning module, you should understand the following:

- How to build applications with K2 Designer using SmartObjects, Views, Forms and Workflows
- How to build SmartObjects in K2 Designer
- How to use SmartObjects in Views
- The basics of SmartForms
  - Views and Forms
  - List Views and Item Views
  - Controls
  - Rules
  - Form-Workflow integration

Knowledge-check and review questions

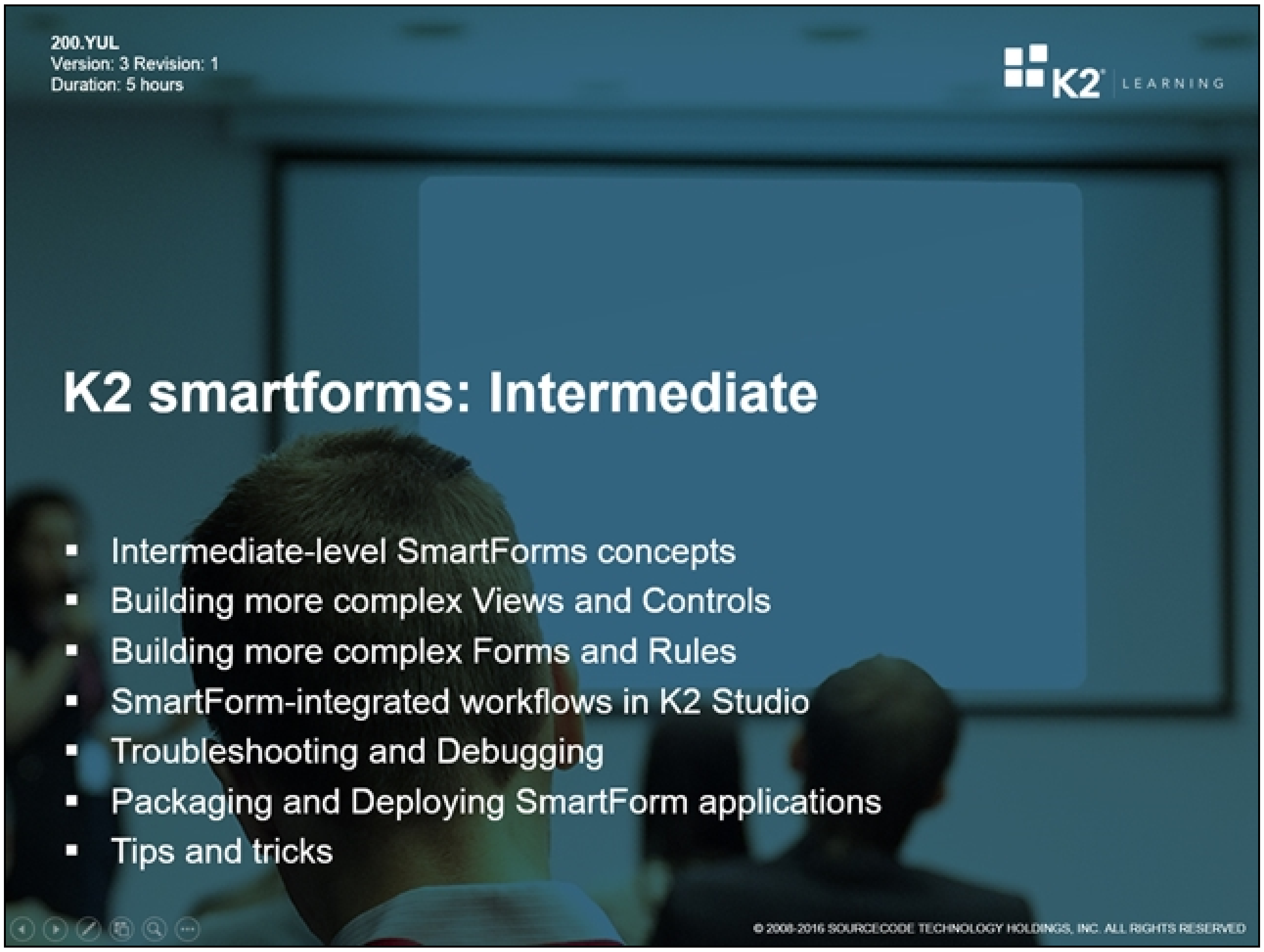
Q: Did your application work as expected when you tested it? If not, could you determine why?
A: (discussion question)

Q: Could you use a similar application in your organization to allow users to maintain their own contact information in Active Directory?
A: (discussion question)

Q: How would the sample application look in your organization? What would be different in the Workflow? Or the Forms?
A: (discussion question)

Q: Do you understand the difference between SmartObjects, Views, Forms and Workflows in a K2 application?
A: (discussion question)

200.YUL: K2 smartforms Intermediate



200.YUL
Version: 3 Revision: 1
Duration: 5 hours

K2 smartforms: Intermediate

- Intermediate-level SmartForms concepts
- Building more complex Views and Controls
- Building more complex Forms and Rules
- SmartForm-integrated workflows in K2 Studio
- Troubleshooting and Debugging
- Packaging and Deploying SmartForm applications
- Tips and tricks

© 2008-2016 SOURCECODE TECHNOLOGY HOLDINGS, INC. ALL RIGHTS RESERVED

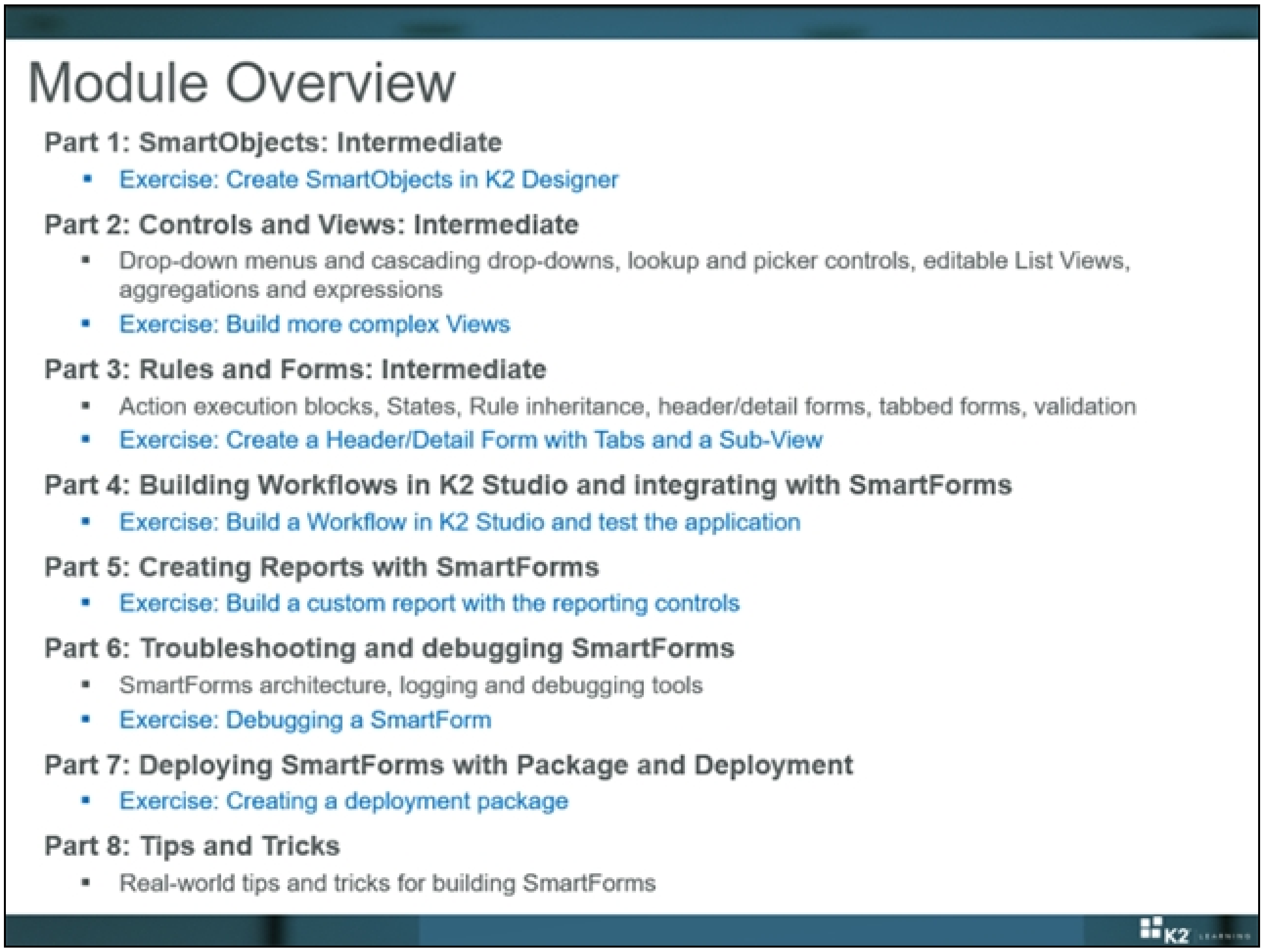
Video

The *200.YUL- K2 smartforms: Intermediate* training module explains how to build more complex SmartForms and SmartForm-centric applications with Data, Forms and Workflows. Concepts covered in this training course include:

- Building advanced-mode SmartObjects in K2 Designer
- Working with Controls like drop-down lists, lookups and picker controls, and populating list controls with data from SmartObjects
- Working with more complex Rules and Rule Execution Blocks
- Working with editable List Views and other more complex View topics such as Expressions, Aggregations and Validations
- Building more advanced Forms like Header-Detail Forms
- Integrating Forms with Workflows built in K2 Studio
- Architecture of SmartForms and troubleshooting SmartForms
- Packaging and Deploying SmartForm applications
- How to implement SmartForms effectively with real-world tips and tricks

Note
You must have completed the *100.YYZ - SmartForms: Fundamentals* training module before starting this module, since this module will be building on your fundamental understanding of SmartForms.

Module Overview



Module Overview

Part 1: SmartObjects: Intermediate
- Exercise: Create SmartObjects in K2 Designer

Part 2: Controls and Views: Intermediate
- Drop-down menus and cascading drop-downs, lookup and picker controls, editable List Views, aggregations and expressions
- Exercise: Build more complex Views

Part 3: Rules and Forms: Intermediate
- Action execution blocks, States, Rule inheritance, header/detail forms, tabbed forms, validation
- Exercise: Create a Header/Detail Form with Tabs and a Sub-View

Part 4: Building Workflows in K2 Studio and integrating with SmartForms
- Exercise: Build a Workflow in K2 Studio and test the application

Part 5: Creating Reports with SmartForms
- Exercise: Build a custom report with the reporting controls

Part 6: Troubleshooting and debugging SmartForms
- SmartForms architecture, logging and debugging tools
- Exercise: Debugging a SmartForm

Part 7: Deploying SmartForms with Package and Deployment
- Exercise: Creating a deployment package

Part 8: Tips and Tricks
- Real-world tips and tricks for building SmartForms

Video

This module consists of eight parts. The first four parts roughly align with the K2 smartforms Fundamentals course: SmartObjects, Views, Forms and Workflows. In K2 smartforms Intermediate, we will look into more advanced topics in each of those areas.

The last four parts will introduce more in-depth topics for SmartForms such as creating reports with Forms, SmartForms architecture, troubleshooting and debugging and using the K2 Package and Deployment tool. We will finish the course by discussing some real-world tips and tricks for building SmartForms.

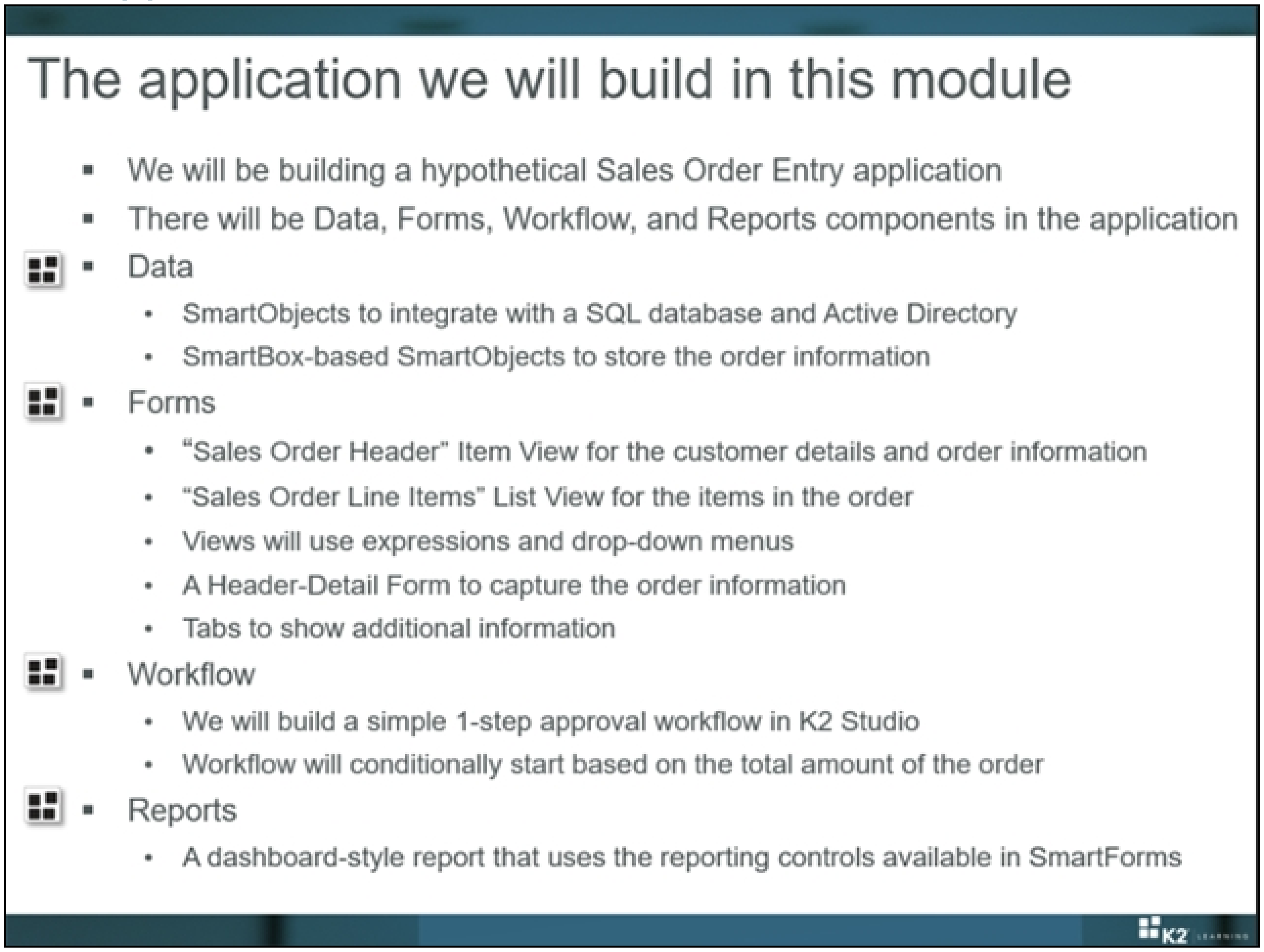
With the exception of Part 8, each part will end with a hands-on exercise where you will have the opportunity to put into practice the information covered in that section.

- Part 1 discusses more advanced SmartObject concepts:
  - SmartObject associations and why they are used
  - Creating advanced SmartObjects in K2 Designer
  - Exercise: build the SmartObjects that we will use in the Sales Orders application. This includes SmartObjects that will integrate with external data sources like Active Directory and Microsoft SQL databases.
- Part 2 covers more complex topics around Controls and Views, including:
  - Using drop-down menus and cascading drop-downs, Lookup and Picker Controls
  - Building Edit Mode List Views
  - Using aggregations and expressions
  - Exercise: build the more complex Views for the Sales Orders application, including an editable List View for the order line items
- Part 3 covers more complex topics around Rules and Forms, including:
  - Rule concepts like Action Execution Blocks, Rules Inheritance, States and how to implement advanced validation
  - How to build header/detail Forms

    - Creating Tabbed Forms and using Sub-views
    - Exercise: build the header/detail Form for the Sales Orders application, including tabbed Views and Sub-views
- Part 4 looks at building Workflows in K2 Studio and integrating them with SmartForms
    - How SmartForms integrate with Workflows
    - Options for integrating Forms with Workflows
    - Exercise: build a simple Workflow in K2 Studio and integrate it with the Forms used in the Sales Orders application
- Part 5 explains how to build SmartForms-based reports with the reporting Controls
    - How to use the available reporting controls in SmartForms to build dashboard-style reports
    - Exercise: build a simple dashboard-style report with a SmartForm using reporting Controls
- Part 6 is a more technical section and includes:
    - SmartForms architecture and authentication
    - Logging tools
    - Troubleshooting and debugging SmartForms
    - Exercise: learn how to debug a SmartForm
- Part 7 explains how to deploy SmartForm-based applications and includes:
    - Using the K2 Package and Deployment tool to move SmartForms-based applications between environments
    - Considerations when packaging and deploying applications
    - Exercise: creating a deployment package
- Part 8 covers some tips and tricks for working with SmartForms, including:
    - Tips and tricks for developing SmartForms
    - Tips to improve performance
    - Some interesting things that you can do with SmartForms

As you can tell, this is quite an involved module and we will be covering a lot of information, much more than the 100.YYZ SmartForms Fundamentals module. However, at the end of this module you should be able to implement fairly complex solutions with SmartForms.

The application we will build in this module



The application we will build in this module

- We will be building a hypothetical Sales Order Entry application
- There will be Data, Forms, Workflow, and Reports components in the application
- Data
  - SmartObjects to integrate with a SQL database and Active Directory
  - SmartBox-based SmartObjects to store the order information
- Forms
  - "Sales Order Header" Item View for the customer details and order information
  - "Sales Order Line Items" List View for the items in the order
  - Views will use expressions and drop-down menus
  - A Header-Detail Form to capture the order information
  - Tabs to show additional information
- Workflow
  - We will build a simple 1-step approval workflow in K2 Studio
  - Workflow will conditionally start based on the total amount of the order
- Reports
  - A dashboard-style report that uses the reporting controls available in SmartForms

Video

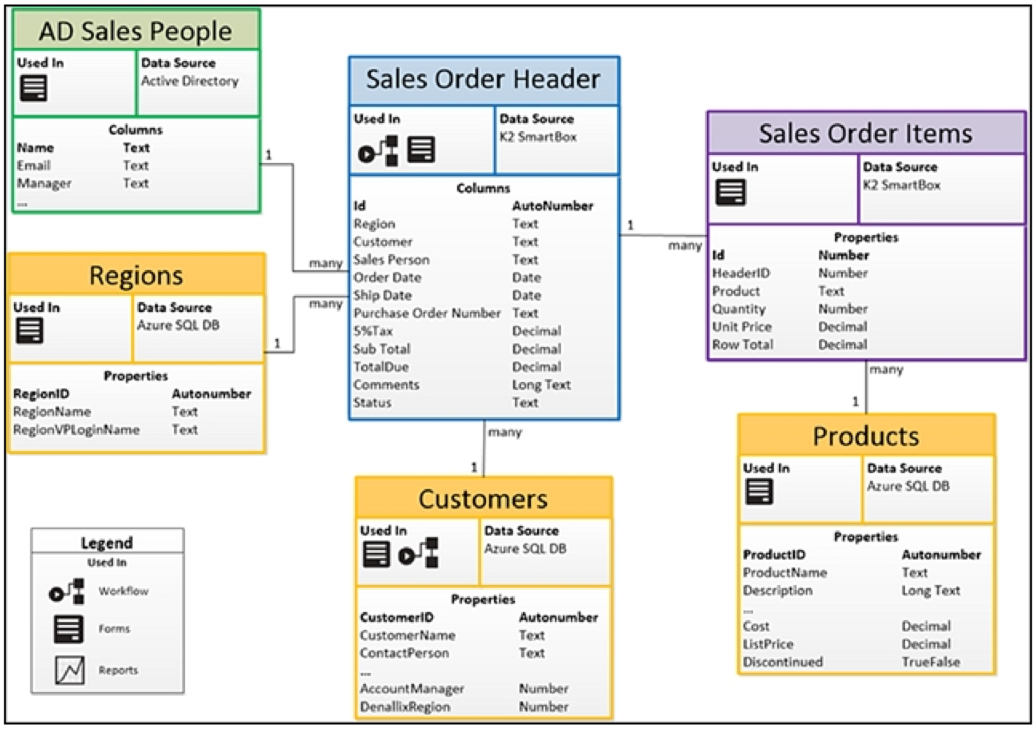
As with other learning modules, we will be building an application to apply the concepts covered. In this module, we will create a hypothetical Sales Orders application for our fictional Denallix organization. The primary focus for this application is on the Data elements (where we will be using several different data sources) and the Forms elements (where we will create more advanced SmartForms, including master/detail Forms, Tabbed Forms and more advanced Rules). While this is not a generic application you may find useful in your organization, you can still rebuild the application in another K2 environment if you wish to practice.

Our Sales Orders application will have Data (SmartObjects), Forms (SmartForms-based Views and Forms), Workflow (built in K2 Studio) and a custom Reports component.

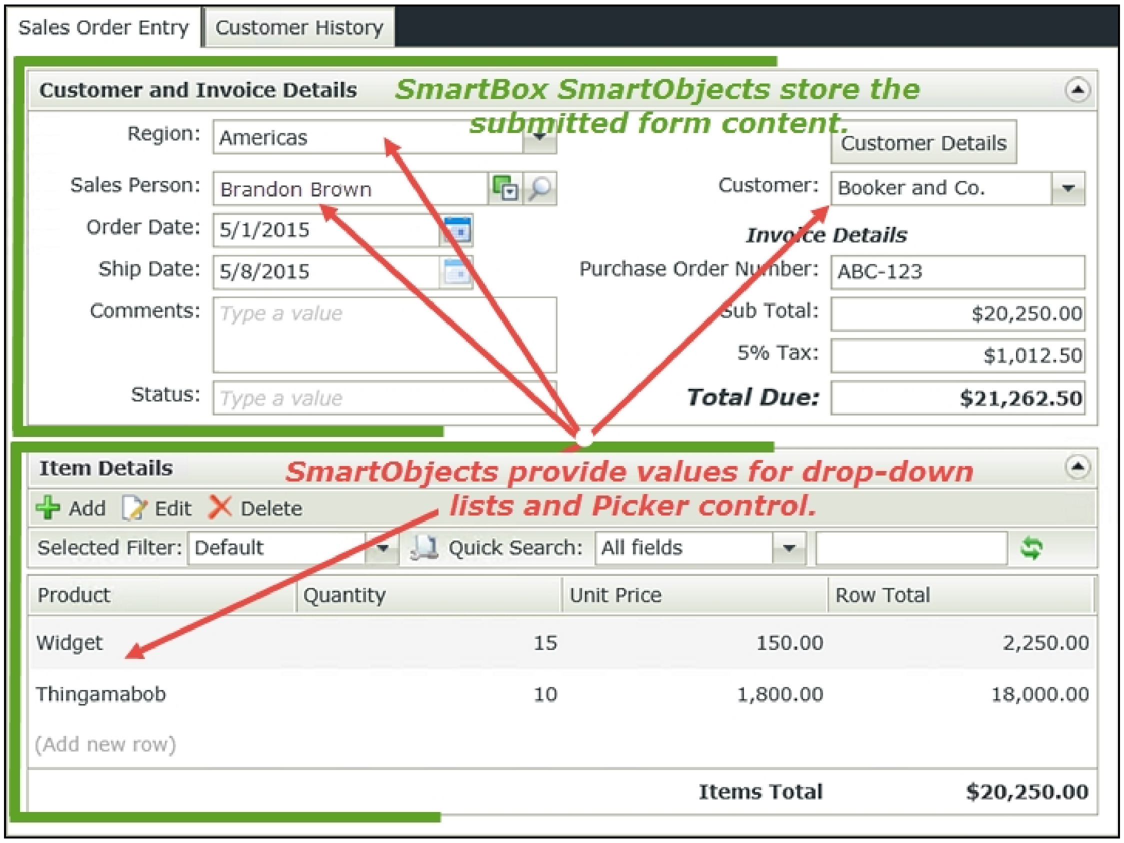
Data

This application requires six SmartObjects for its Data components. The first SmartObject (AD Sales People) will be connected to Active Directory and will provide the values for Sales Person Picker Control. The next three SmartObjects (Regions, Customers and Products) are lookup SmartObjects that will be connected to an external Azure SQL database, and will provide the Form field values for the Regions, Customers and Products drop-down lists. The final two SmartObjects (Sales Order Header and Sales Order Items) will be SmartBox SmartObjects. The data entered when capturing a new sales order will be saved into these SmartBox SmartObjects; all the other data is essentially "lookup" data that is used to populate Controls with values.

The diagram below illustrates the Data objects used in this application, their respective Data Sources and the relationships between the SmartObjects.



The screenshot below illustrates how the SmartObjects will present their data on the Sales Orders Form.


Sales Order Entry
Customer History
SmartBox SmartObjects store the submitted form content.
SmartObjects provide values for drop-down lists and Picker control.
Customer Details
Add
Edit
Delete

Customer and Invoice Details

Region:	Americas		
Sales Person:	Brandon Brown	Customer:	Booker and Co.
Order Date:	5/1/2015		
Ship Date:	5/8/2015	Purchase Order Number:	ABC-123
Comments:	Type a value	Sub Total:	$20,250.00
		5% Tax:	$1,012.50
Status:	Type a value	**Total Due:**	**$21,262.50**

Invoice Details

Item Details

Selected Filter: Default Quick Search: All fields

Product	Quantity	Unit Price	Row Total
Widget	15	150.00	2,250.00
Thingamabob	10	1,800.00	18,000.00
(Add new row)			
		Items Total	**$20,250.00**

Forms

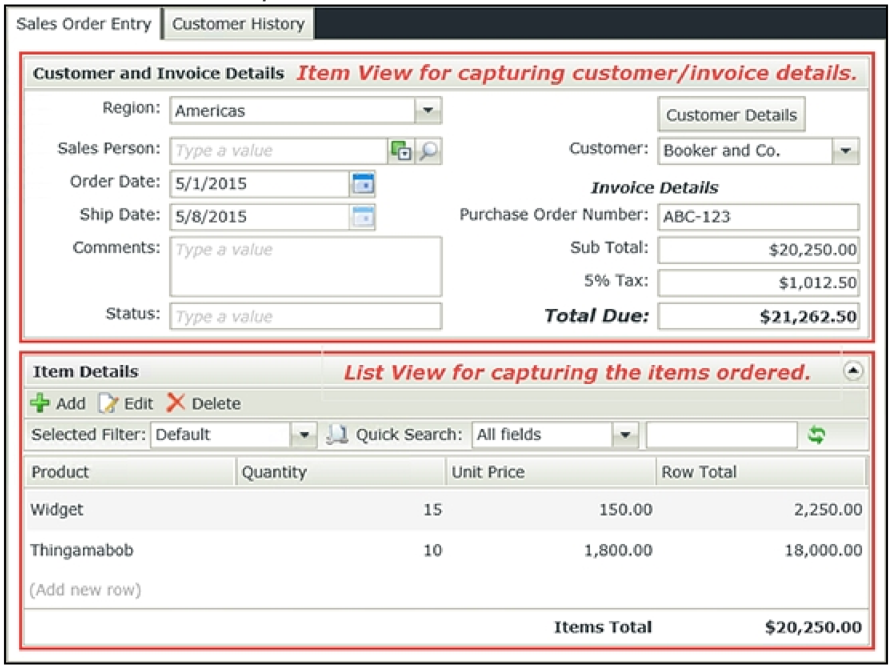
This application will consist of two Forms that will be used for the order entry and approval steps. A third Form (not part of the Workflow process) will display custom reporting Controls in a dashboard format.

The Sales Orders Form will contain two Views. The first View, an Item View, will be used to capture the customer and invoice details. The second View, a List View, will contain editable rows to capture the items that are being ordered. The

Customer drop-down list will be configured so that it only displays customers associated with the Region that has been selected. There are a number of expressions and Rules that we will apply to perform calculations and populate fields. For example, when a product is selected from the drop-down list, the price will automatically be called to the Unit Price field. As items are added, the Sub-Total, 5% Tax and Total Due fields will be automatically calculated.

This Form will also contain a tabbed Form displaying the Customer History and a Sub-view which will display the details for the customer selected.

The Form used to capture the Sales Order entries


Sales Order Entry
Customer History
Customer and Invoice Details
Item View for capturing customer/invoice details.
Customer Details
Invoice Details
Item Details
List View for capturing the items ordered.
Add
Edit
Delete

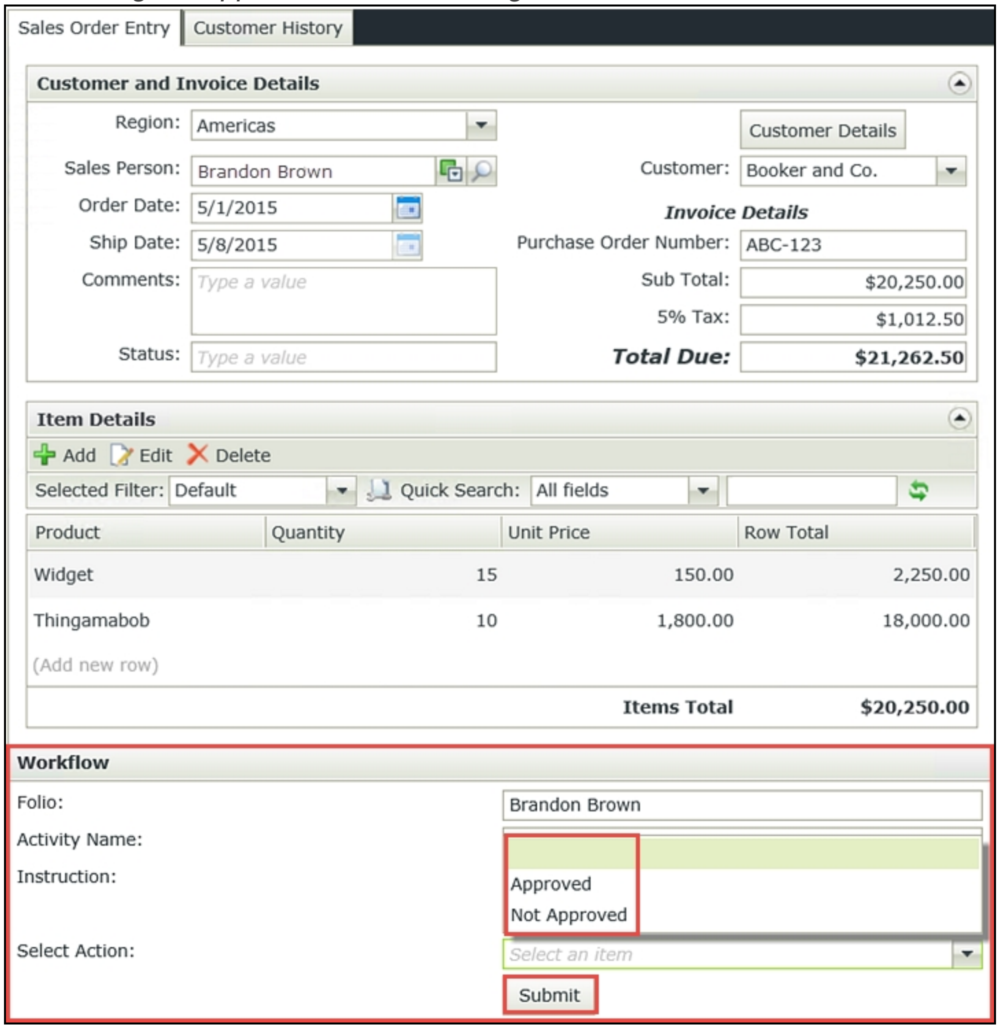
Region:	Americas
Sales Person:	Type a value
Order Date:	5/1/2015
Ship Date:	5/8/2015
Comments:	Type a value
Status:	Type a value
Customer:	Booker and Co.
Purchase Order Number:	ABC-123
Sub Total:	$20,250.00
5% Tax:	$1,012.50
Total Due:	**$21,262.50**

Selected Filter: Default Quick Search: All fields

Product	Quantity	Unit Price	Row Total
Widget	15	150.00	2,250.00
Thingamabob	10	1,800.00	18,000.00
(Add new row)			
		Items Total	**$20,250.00**

The second Form is used by the approving manager if the sales order requires approval. Along with the original Form content, K2 has added a third View (Workflow) containing the action options (Approved or Not Approved) along with a submit button. The Workflow View is at the bottom of the Sales Orders Form, which is a configuration we will set during the Workflow build.

The Manager's Approval Form containing the Workflow View



Sales Order Entry | Customer History

Customer and Invoice Details

Region:	Americas		Customer Details
Sales Person:	Brandon Brown	Customer:	Booker and Co.
Order Date:	5/1/2015		***Invoice Details***
Ship Date:	5/8/2015	Purchase Order Number:	ABC-123
Comments:	Type a value	Sub Total:	$20,250.00
		5% Tax:	$1,012.50
Status:	Type a value	***Total Due:***	**$21,262.50**

Item Details

Add Edit Delete

Selected Filter: Default Quick Search: All fields

Product	Quantity	Unit Price	Row Total
Widget	15	150.00	2,250.00
Thingamabob	10	1,800.00	18,000.00
(Add new row)			
		Items Total	**$20,250.00**

Workflow

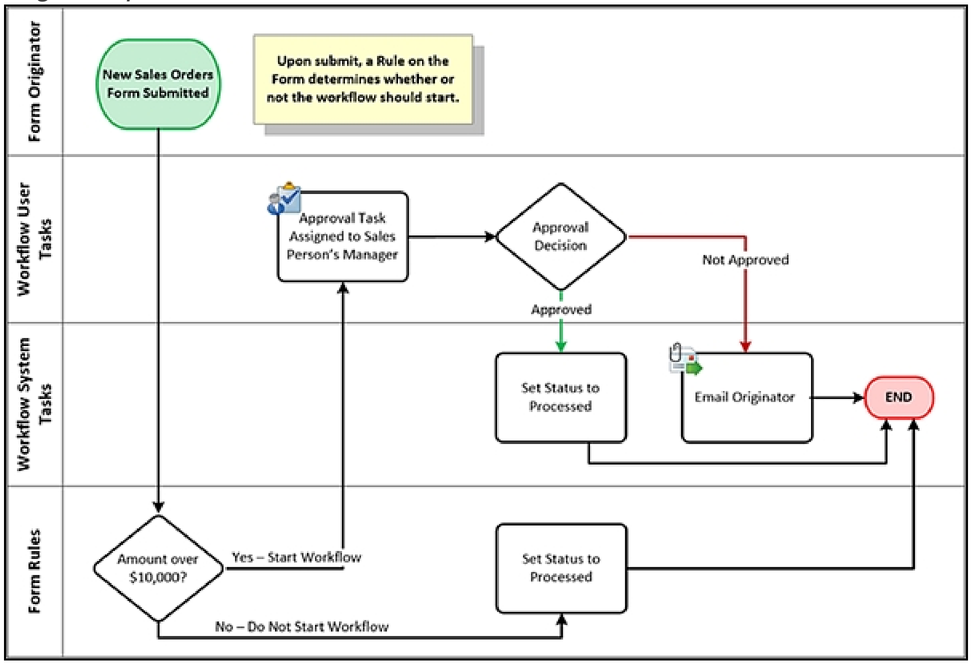
Folio:	Brandon Brown
Activity Name:	
Instruction:	
Select Action:	Select an item

Workflow

The diagram below represents the logical flow of the process in our application. In this scenario, we will add a conditional Rule to the Form so that the Workflow is only started if a certain set of criteria is met, otherwise the sales order will just be saved. Of course, real-world Workflows are usually much more involved than the one we are building here, but we are mostly interested in the integration between the Workflow and SmartForms, rather than building a very complex Workflow.

There are two swim-lanes for User Tasks. The first is the Form Originator, or the person submitting the Form for the first time. The second swim-lane represents the User Tasks, in our case this will be the Form Originator’s Manager who must approve the sales order if the amount is greater than $10,000. The third lane represents System Tasks, or tasks that K2 will manage behind-the-scenes. The fourth lane represents Form Rules which are performed outside of the K2 Studio-generated Workflow.

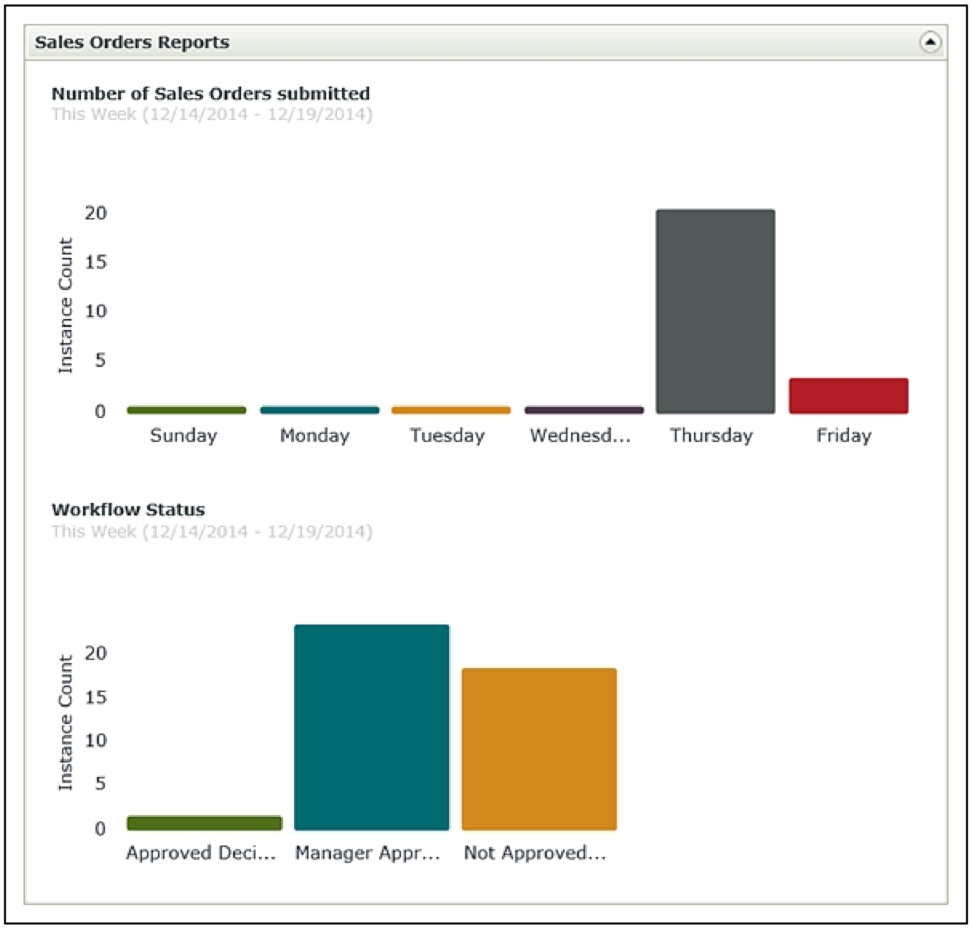
Logical representation of the Sales Order Workflow



Reports

In this application, we will use the reporting Controls found in K2 Designer to create a dashboard-style report. We will add two charting Controls to a new View and then a new Form. The first chart will display the number of sales orders submitted, and the second chart will display the status of submitted Workflows.

The screenshot below shows the eventual custom report SmartForm that we will create for this Sales Orders application.



PART 1: SmartObjects: Intermediate

PART 1

SMARTOBJECTS: INTERMEDIATE

- ✓ Registering Service Instances
- ✓ Building advanced SmartObjects that integrate with SQL Databases and Active Directory
- ✓ Creating Associations between SmartObjects

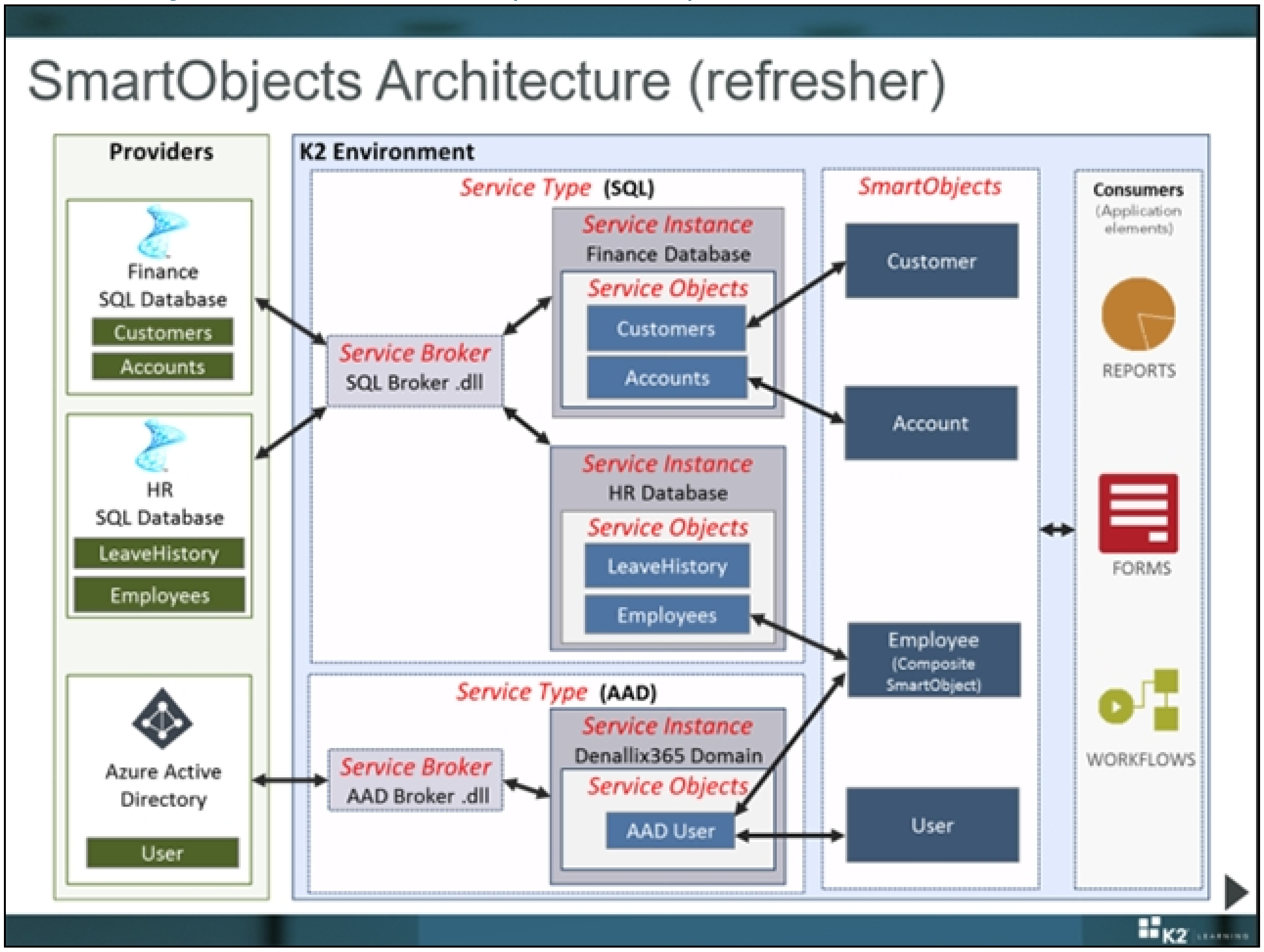
Video

In Part 1 we will look at intermediate-level SmartObject concepts including:

- Registering Service Instances
- Manually creating advanced-mode SmartObjects in K2 Designer that integrate with back-end systems like SQL databases and Active Directory
- Creating associations between SmartObjects and why you would want to do so

At the end of Part 1, we will do an exercise to create all of the SmartObjects that are required for our Sales Orders application.

SmartObjects Architecture (refresher)



Video

Note
This is a refresher of the SmartObjects architecture we discussed in the K2 Core training course. If you recently completed that course, you can just skim over this topic.

Note
This topic contains fairly technical content. It is not critical to understand everything in the slide, as long as you are familiar with the basic architecture of SmartObjects.

The terms are important to understand if you want to know how to integrate specific back-end systems with SmartObjects and will be creating SmartObjects to interact with other systems.

Service Objects, Service Types and SmartObjects are all components of K2 SmartObjects that allow K2 to connect to and interact with Data Providers and Data Consumers. Service Types are specific Data Providers such as Active Directory, Oracle Databases or Microsoft SQL Databases.

Sometimes you may need to configure an instance of a Service Type (this is called a Service Instance) because you have to provide a connection string so that K2 knows how to connect to the provider you want to interact with.

Service Objects are just an internal representation of the items that K2 discovered in the Data Provider. Service Objects are the Properties of the SmartObject.

Consider the diagram in the slide above. Here we have a SQL database for Finance. We have an instance of the SQL service that is configured to point to the Finance SQL database. Registering the Service Instance will discover the available tables, views and stored procedures in the target database and generate Service Objects for those artifacts.

Then, we can either auto-generate SmartObjects for the Service Objects, or manually create SmartObjects using K2 design tools.

Next, we add another Service Instance of the SQL Service Type to expose an HR SQL database as SmartObjects. The same discovery procedure follows the entities in the target database, and are exposed as Service Objects. We could now generate or manually create SmartObjects for those Service Objects if we wanted to, but we don't have to.

Finally, we use an instance of the Azure Active Directory (AAD) Service Type to expose user data that lives in our Azure Active Directory domain for Denallix. Again, this creates Service Objects for the entities in the AAD system.

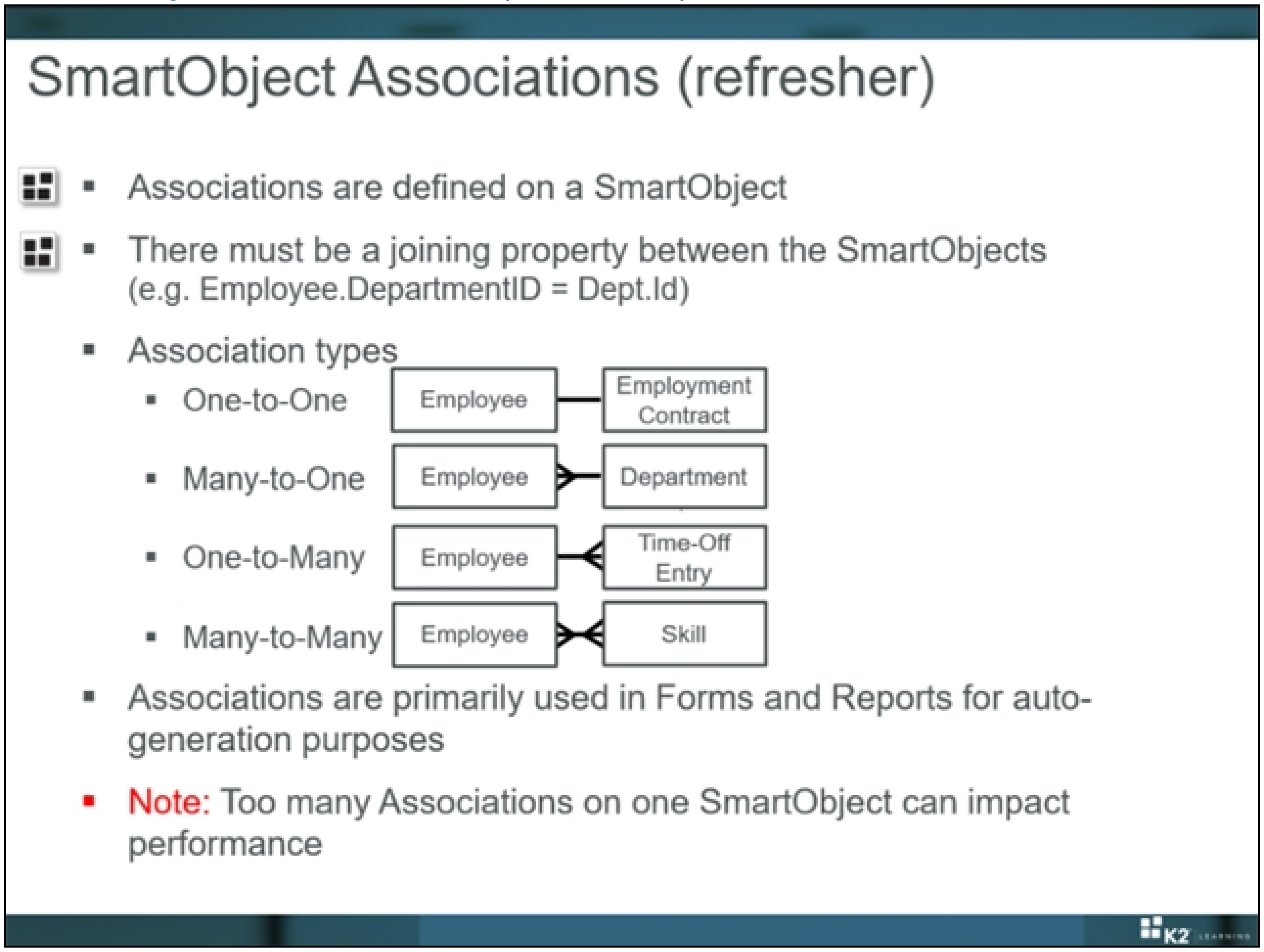
We can combine Service Objects to build up Composite SmartObjects: in this case, we are combining data from the Azure AD store and the Employee database to create a logical "Employee" SmartObject that combines data from both systems (and represents it as a logical business object to the consuming Forms, Workflows and Reports). (If you are familiar with SQL, Composite SmartObjects are similar to JOIN statements for SmartObjects except that the data could come from different systems).

What is important is that Data Consumers don't know anything about the underlying systems. As far as they are concerned, they "talk to" an abstracted yet consistent set of business entities where K2 takes care of the integration behind the scenes.

Summary

- Service Instances are used to point a Service Type to a specific data source, for example registering an instance of the SQL Server Service Type to point it to the Finance database.
- Registering a Service Instance will discover the available artifacts in the target system, such as SQL database tables and stored procedures.
- You can then manually create or generate SmartObjects for the artifacts in the target system.
- You can have multiple instances of the same Service Type that each point to different target systems.
- You can combine data from multiple Service Objects to create Composite SmartObjects that combine data from different systems as well.

SmartObject Associations (refresher)


SmartObject Associations (refresher)

- Associations are defined on a SmartObject
- There must be a joining property between the SmartObjects (e.g. Employee.DepartmentID = Dept.Id)
- Association types
  - One-to-One
  - Many-to-One
  - One-to-Many
  - Many-to-Many
- Associations are primarily used in Forms and Reports for auto-generation purposes
- Note: Too many Associations on one SmartObject can impact performance

Video

Note
This topic is a refresher of content previously covered in the K2 Core training course. Take a moment to refresh your understanding of SmartObject Associations since we will be using them in the exercise for this part of the training module.

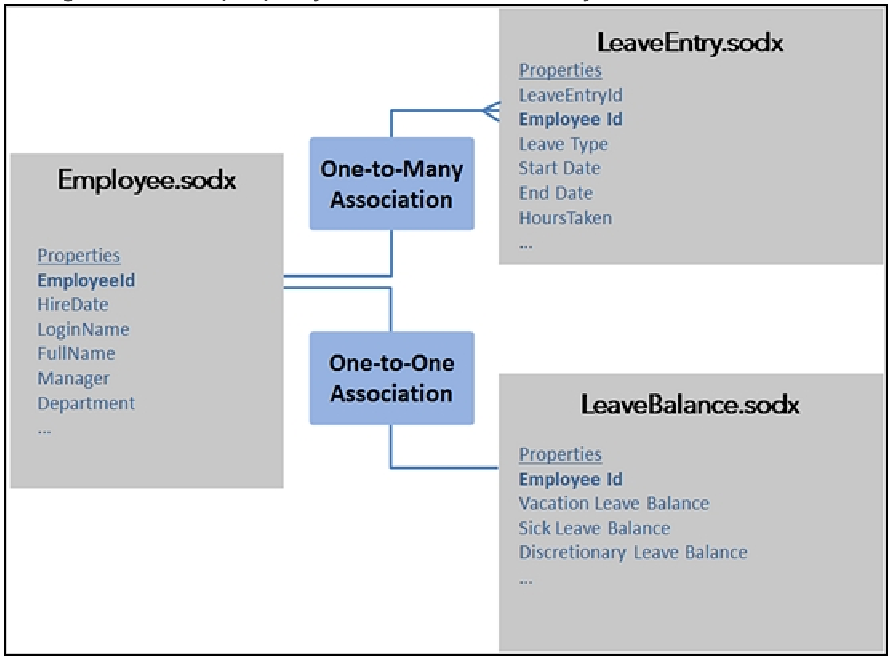
SmartObject Associations are used to create logical relationships between different SmartObjects. These associations support different kinds of relationships between SmartObjects and are typically used in user interfaces. For example, a SmartForm may use an association between the Employee SmartObject and the LeaveEntries SmartObject to display all leave taken by employees. In another example for Workflows, you can use SmartObject Associations to start a sub-process for each line item in the Product SmartObject that relates to a specific SalesOrder SmartObject.

Associations are always defined between two SmartObjects, but there is no restriction on the number of associations a SmartObject may have with other SmartObjects.

Note
Associations are not the same as referential integrity and do not implement referential integrity. Associations are merely additional information that describes how a particular SmartObject relates to other SmartObjects, and is mostly used when you are designing Forms and reports against that SmartObject.

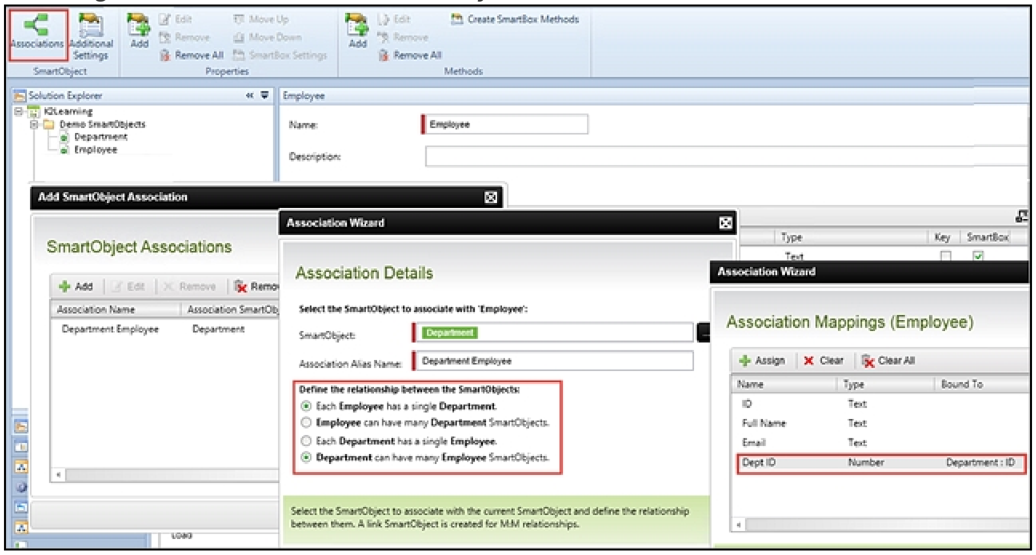
Consider the example association diagram shown below. The **Employee** SmartObject has associations with both the **LeaveBalance** SmartObject (One-to-One Association) and the **LeaveEntry** SmartObject (One-to-Many Association). There is a common property called Employee Id that can be used to "match" Leave Entries and Leave Balance with an Employee, based on the Employee Id. Using these associations, designers can create a Form which combines data from all three SmartObjects. For example, to create a report for an employee’s current leave balances and the leave taken by the employee.

Using a common property to define a SmartObject Association



The association is defined using a wizard (shown in the screenshot below).

Defining an association for a SmartObject



Employee

Name: Employee

Description:

Add SmartObject Association

SmartObject Associations

Association Name	Association SmartOb
Department Employee	Department

Association Wizard

Association Details

Select the SmartObject to associate with 'Employee':

SmartObject: Department

Association Alias Name: Department Employee

Define the relationship between the SmartObjects:

[✓ Each **Employee** has a single **Department**.]
Employee can have many **Department** SmartObjects.
Each **Department** has a single **Employee**.
[✓ **Department** can have many **Employee** SmartObjects.]

Select the SmartObject to associate with the current SmartObject and define the relationship between them. A link SmartObject is created for M:M relationships.

Association Wizard

Association Mappings (Employee)

Name	Type	Bound To
ID	Text	
Full Name	Text	
Email	Text	
Dept ID	Number	Department : ID

K2 provides the following types of SmartObject Associations, and you can create the associations in tools like K2 Designer and K2 Studio.

Association Name	Design environment settings	Example
One-to-One	Each SmartObject 1 has a single SmartObject 2 Each SmartObject 2 has a single SmartObject 1	Each employee has a single Employment contract, and each Employment contract has a single Employee.

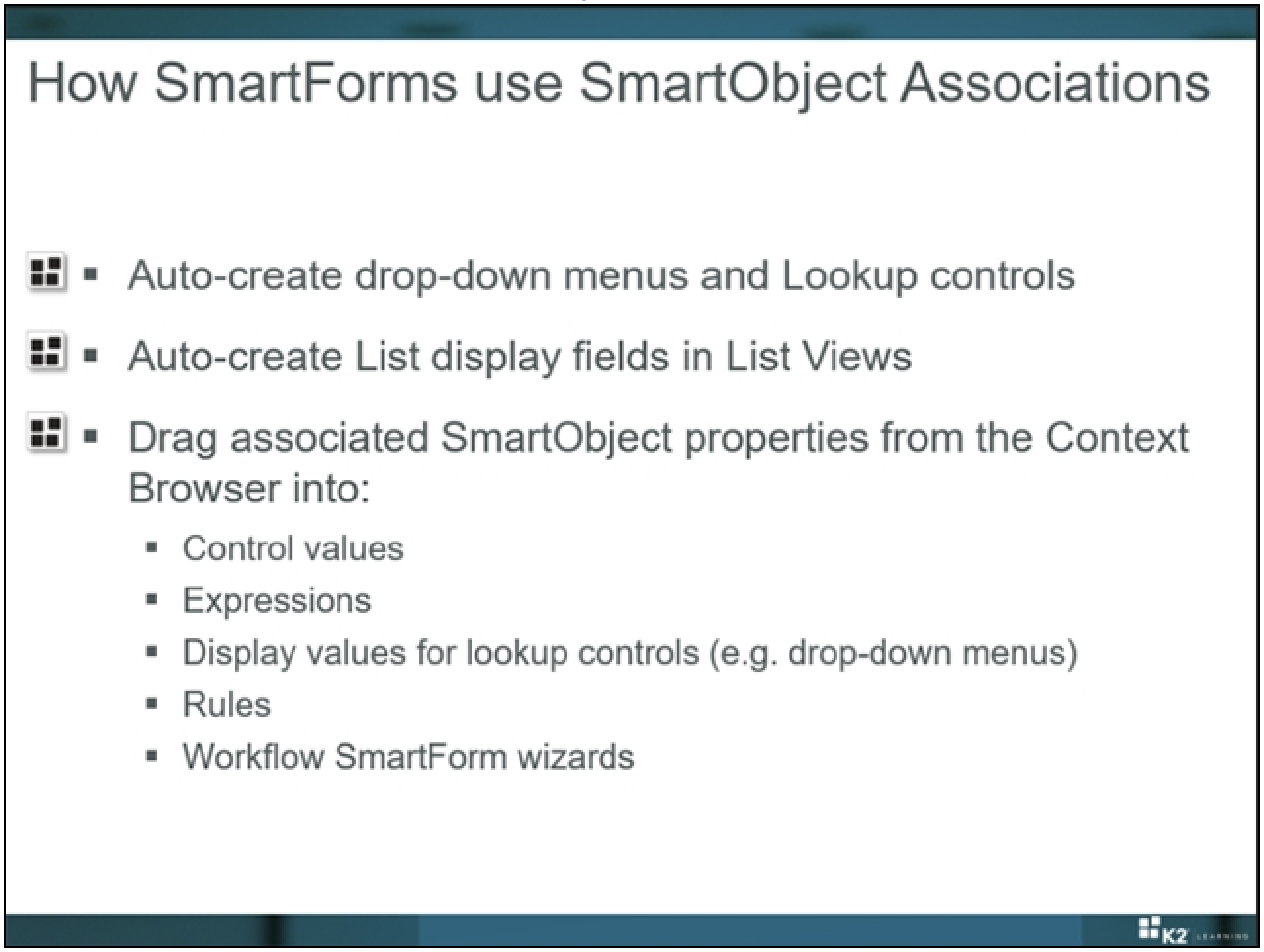
Many-to-One	Each SmartObject 1 has a single SmartObject 2 Each SmartObject 2 can have many SmartObject 1 SmartObjects	An employee can have only one department, while a department can have many employees.
One-to-Many	SmartObject 1 can have many SmartObject 2 SmartObjects Each SmartObject 2 has a single SmartObject 1	An employee can have many time-off entries. A time-off entry can have only one employee.
Many-to-Many	SmartObject 1 can have many SmartObject 2 SmartObjects SmartObject 2 can have many SmartObject 1 SmartObjects For Many-to-Many relationships, K2 will automatically create a linked SmartObject (sometimes called a “hamburger” table) to represent the many-to-many relationship.	An employee can have many skills, and a skill can belong to many employees.

Caution
Using too many associations on one SmartObject could impact performance, so it is not recommended to add associations just for the sake of adding associations. There are some cases where adding associations can be helpful when you are building SmartForms, which we will look at in the next topic.

Summary

- SmartObject Associations are essentially relationships but do not have any referential integrity.
- You don’t have to define relationships, but doing so can make your life a little easier.
- Associations are primarily used so that SmartForms and reports can auto-generate Controls based on the relationships.
- Too many associations could impact runtime performance.

How SmartForms use SmartObject Associations



How SmartForms use SmartObject Associations

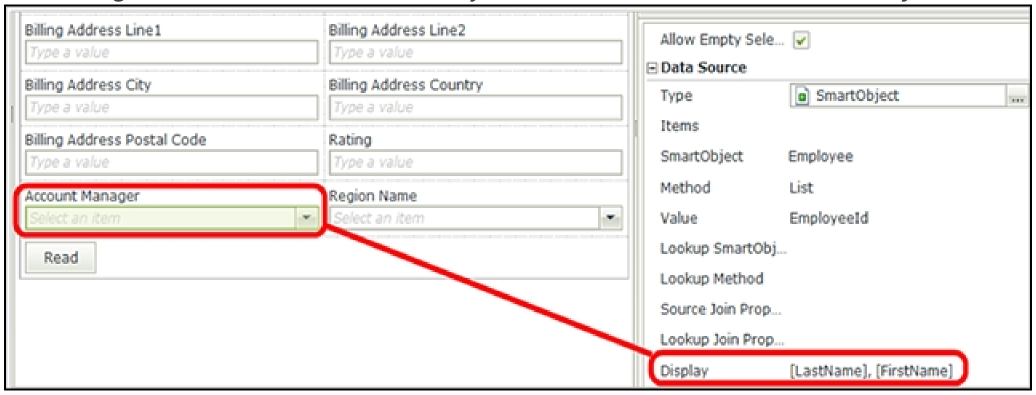
- Auto-create drop-down menus and Lookup controls
- Auto-create List display fields in List Views
- Drag associated SmartObject properties from the Context Browser into:
  - Control values
  - Expressions
  - Display values for lookup controls (e.g. drop-down menus)
  - Rules
  - Workflow SmartForm wizards

K2 smartforms can leverage the associations you have defined between SmartObjects in various ways to make your life as a Form designer easier.

Auto-create Drop-down menus and Lookup Properties

The most common approach is to use an association SmartObject as a "lookup" property, as we described in the previous example where the *Department* SmartObject becomes a Lookup-style drop-down menu because of the association between the *Employee* SmartObject and the *Department* SmartObject. Generated Views will automatically use a drop-down menu for Lookup-style SmartObjects, but of course, you can add your own drop-down menus or display the association in some other way.

An auto-generated View automatically includes the associated SmartObject as a drop-down menu

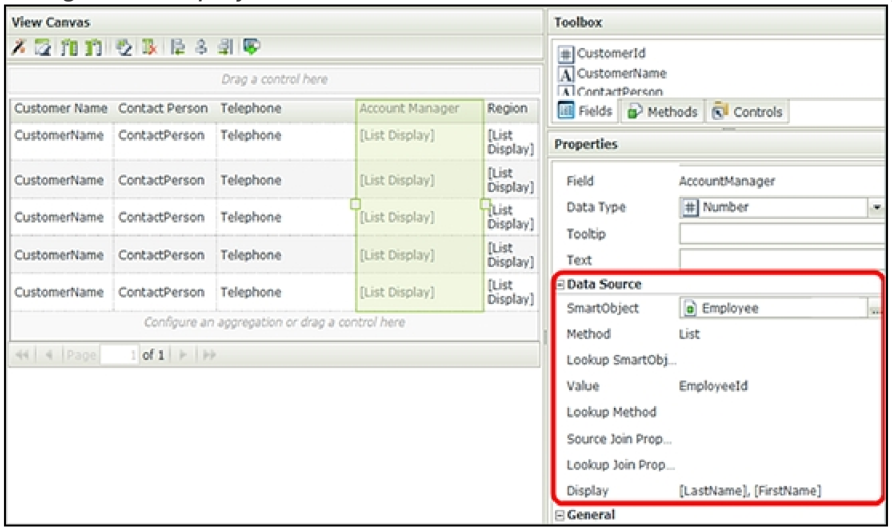


Allow Empty Sele...	✓
Data Source	
Type	SmartObject
Items	
SmartObject	Employee
Method	List
Value	EmployeeId
Lookup SmartObj...	
Lookup Method	
Source Join Prop...	
Lookup Join Prop...	
Display	[LastName], [FirstName]

Auto-create List Display fields in List Views

When you create a List View for a SmartObject with associations, the K2 Designer will automatically insert a List Display field for the associated SmartObject. You can then use the Data Source editing tool to decide which property (or combination of properties) to show in the List. In this example, we want to show the Account Manager's **LastName, Firstname** in the Customer List View. Because of the association that exists between the *Customer* SmartObject and the *Employee* SmartObject, we have the option of displaying another property from the *Employee* SmartObject, not just the ID of the Employee associated with that customer.

Using a List Display Field in a List View to show an associated SmartObject's Properties



View Canvas

Drag a control here

Customer Name	Contact Person	Telephone	Account Manager	Region
CustomerName	ContactPerson	Telephone	[List Display]	[List Display]
CustomerName	ContactPerson	Telephone	[List Display]	[List Display]
CustomerName	ContactPerson	Telephone	[List Display]	[List Display]
CustomerName	ContactPerson	Telephone	[List Display]	[List Display]
CustomerName	ContactPerson	Telephone	[List Display]	[List Display]

Configure an aggregation or drag a control here

Page 1 of 1

Toolbox

CustomerId
CustomerName
ContactPerson

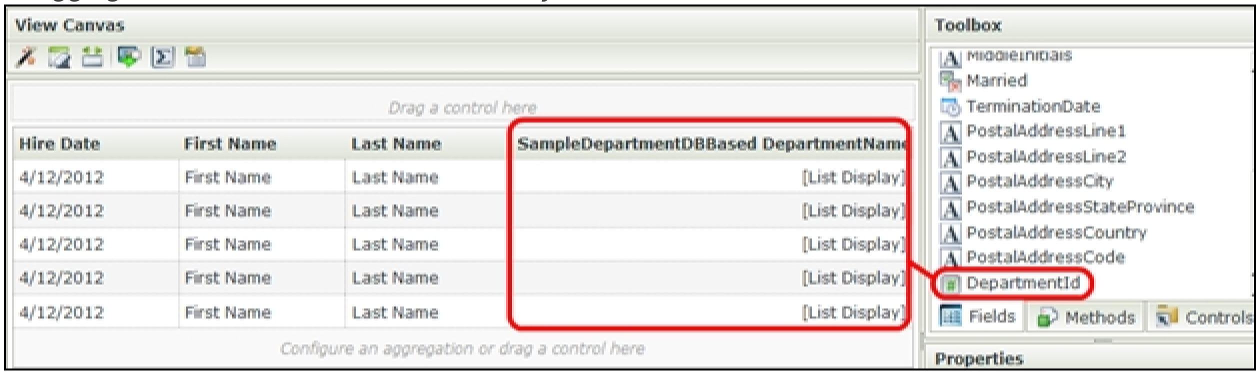
Fields | Methods | Controls

Properties

Field	AccountManager
Data Type	Number
Tooltip	
Text	
Data Source	
SmartObject	Employee
Method	List
Lookup SmartObj...	
Value	EmployeeId
Lookup Method	
Source Join Prop...	
Lookup Join Prop...	
Display	[LastName], [FirstName]
General	

To show values from an associated SmartObject on a List View, drag the ID field onto the List View, then select the **Body** tab in the **Properties** pane to edit the **Data Source** for the List Display field. Select which properties (or combination of properties) you want to display on the list.

Dragging the ID of the associated SmartObject onto the canvas



View Canvas

Drag a control here

Hire Date	First Name	Last Name	SampleDepartmentDBBased DepartmentName
4/12/2012	First Name	Last Name	[List Display]
4/12/2012	First Name	Last Name	[List Display]
4/12/2012	First Name	Last Name	[List Display]
4/12/2012	First Name	Last Name	[List Display]
4/12/2012	First Name	Last Name	[List Display]

Configure an aggregation or drag a control here

Toolbox

MiddleInitials
Married
TerminationDate
PostalAddressLine1
PostalAddressLine2
PostalAddressCity
PostalAddressStateProvince
PostalAddressCountry
PostalAddressCode
DepartmentId

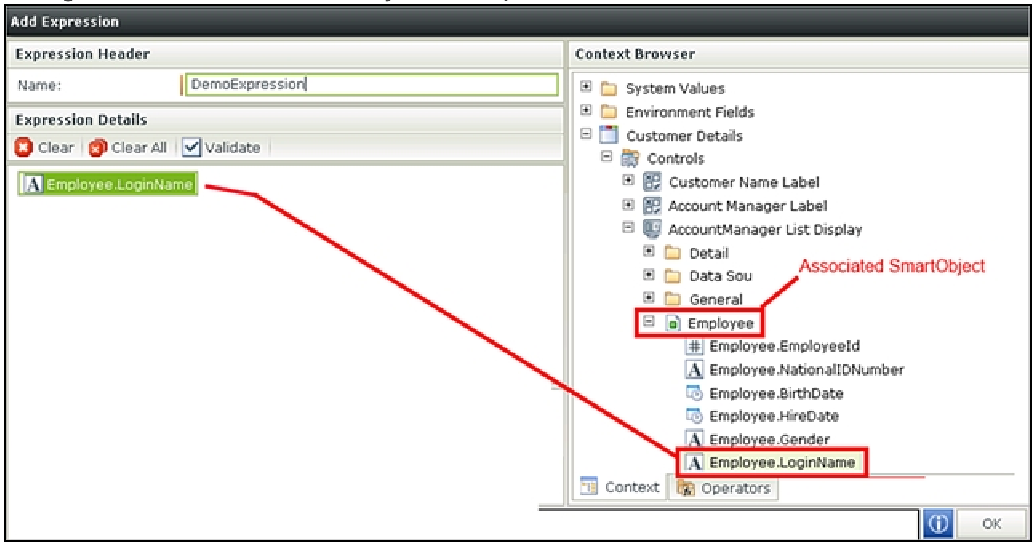
Fields | Methods | Controls

Properties

The Context Browser

When you are designing a View for a SmartObject with associations, the associated SmartObject's Properties and Methods can be available in the Context Browser. In the example below, we are working with a View for the Customer Details SmartObject. Since the Customer Details SmartObject has an association with the Employee SmartObject, we can use properties from the Employee SmartObject in Controls, Expressions and Rules.

Using the associated SmartObject's Properties from the Context Browser



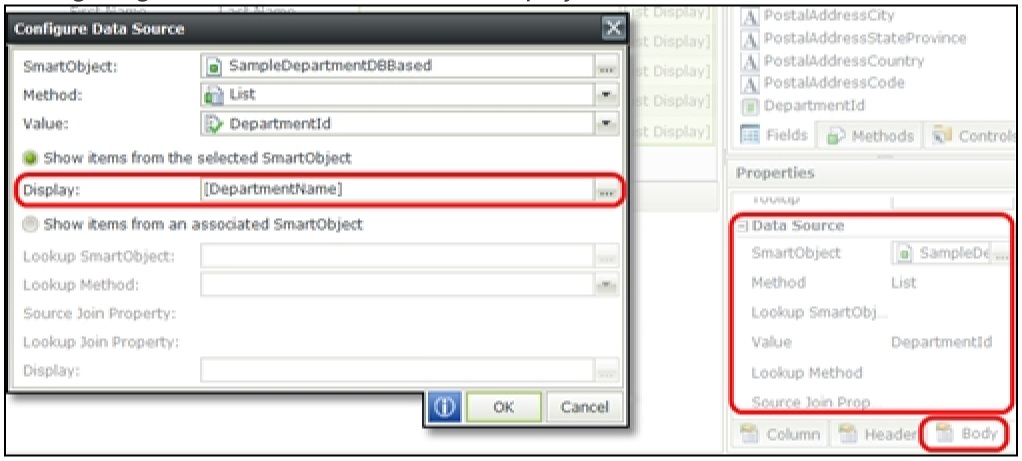
Note
You can view the associated SmartObject in the Context Browser as long as the associated SmartObject is used in at least one Control on the View. This is because of performance considerations: K2 will not automatically retrieve the data from an associated SmartObject if that SmartObject is not actually used anywhere on the View.

You do not have to define an association to show data from another SmartObject on a View. You could, for example, select data from any SmartObject List or Read method to show in a List View, as long as there is some property in the primary SmartObject that you can use to "join" to the secondary SmartObject. Defining associations is just easier because the View designer will discover the association and automatically configure the display value for the List Display Control.

Be careful of the association type, though. It doesn't make sense to try to show a one-to-many association on a List View. For example, it doesn't make sense to create a List View of employee data and then try to add a column to that list to show a list of the leave entries of each employee in the List View.

To add data from an external SmartObject, drag a List Display Control onto the View design canvas and then configure the Control's data source for the relevant SmartObject.

Configuring the data source of the List Display Control on the List View



Summary

- SmartForms can make use of SmartObject associations in the following ways:
  - Auto-create drop-down menus and Lookup Controls
  - Auto-create List Display fields in List Views

- The Context Browser
    - The associated SmartObject's Properties and Methods can be available in the Context Browser for use in mappings, display values, expressions, rules, etc.
- For performance reasons K2 will not automatically retrieve the data from an associated SmartObject if that SmartObject is not actually used anywhere on the View.

MASTERY CHECKPOINT: SmartObjects Intermediate

SmartObjects: Intermediate

- Registering Service Instances
- Creating advanced SmartObjects in K2 Designer
- Creating associations between SmartObjects

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 1 of this module. Use this topic as an opportunity to answer any questions around SmartObjects, especially advanced SmartObjects.

After completing Part 1 you should know:

- Why and how to register Service Instances
- How to design advanced-mode SmartObjects in K2 Designer
- How to define associations between SmartObjects
- The difference between storing and retrieving data from some external system vs. storing and retrieving data from a K2 SmartBox.

Knowledge-check questions

Q: Why do you have to register a Service Instance when you want to integrate with an external system?
Reveal answer**A:** The Service Instance tells K2 how to connect to that system, including what security credentials to use. Registering a Service Instance also discovers the artifacts in the external system and exposes them as Service Objects, so that you can create SmartObjects from those Service Objects.

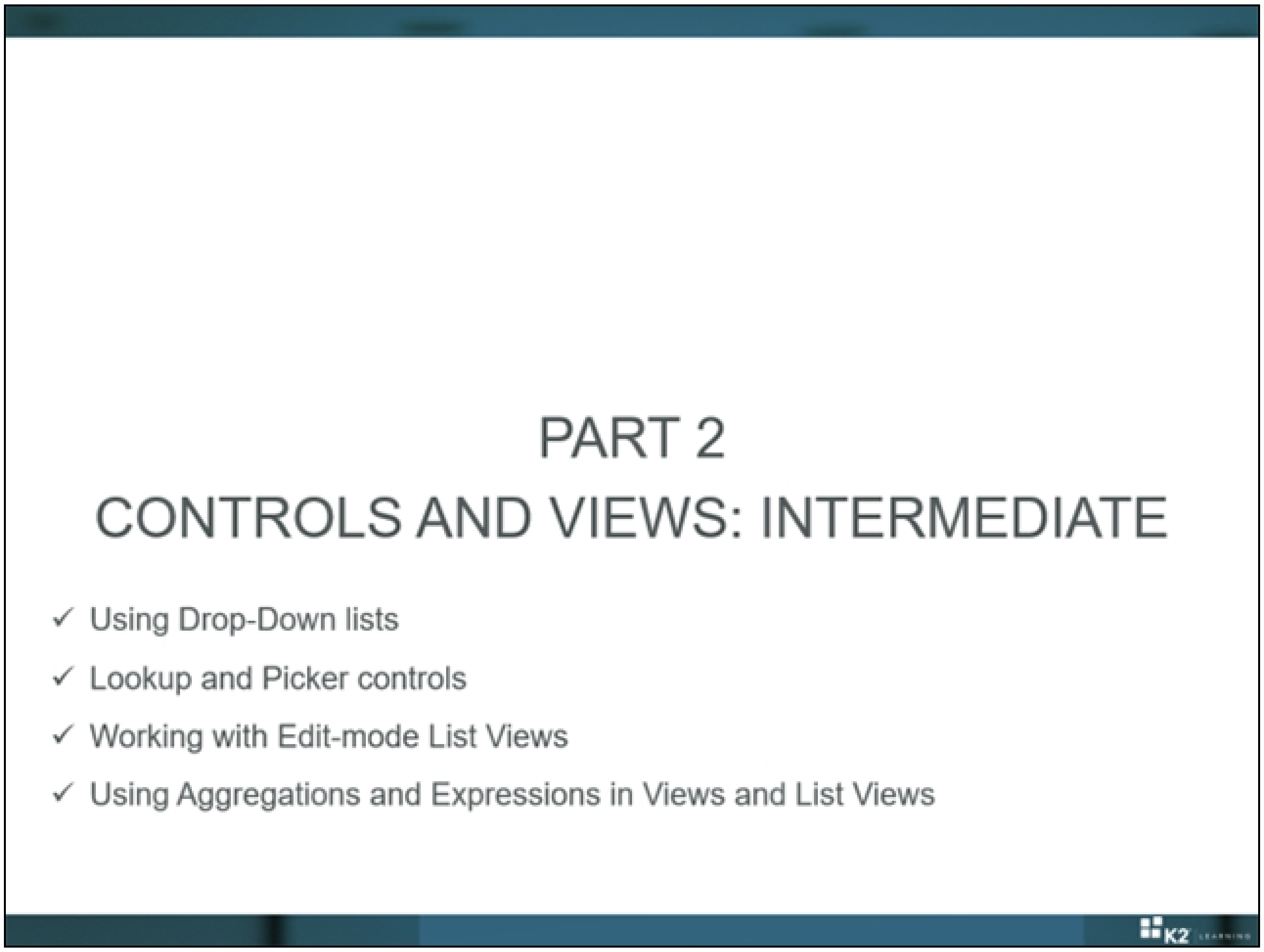
Q: What are some of the associations you can define between SmartObjects?
Reveal answer**A:** One-to-one, one-to-many, many-to-one, many-to-many.

Q: Why would you want to define associations between SmartObjects?
Reveal answer**A:** It makes life easier when working with the SmartObjects in things like reports, and makes it easier to design Views and work with the associated SmartObjects in things like Rules and expressions.

Q: How is a SmartBox SmartObject different to SmartObjects that integrate with other systems?
Reveal answer**A:** A SmartBox-based SmartObject stores its data in the K2 Database in a table that is created by K2 for

that SmartObject. SmartObjects that integrate with other systems do not store data in K2, instead, they retrieve their data from that external system.

PART 2: Controls and Views: Intermediate



PART 2

CONTROLS AND VIEWS: INTERMEDIATE

- ✓ Using Drop-Down lists
- ✓ Lookup and Picker controls
- ✓ Working with Edit-mode List Views
- ✓ Using Aggregations and Expressions in Views and List Views

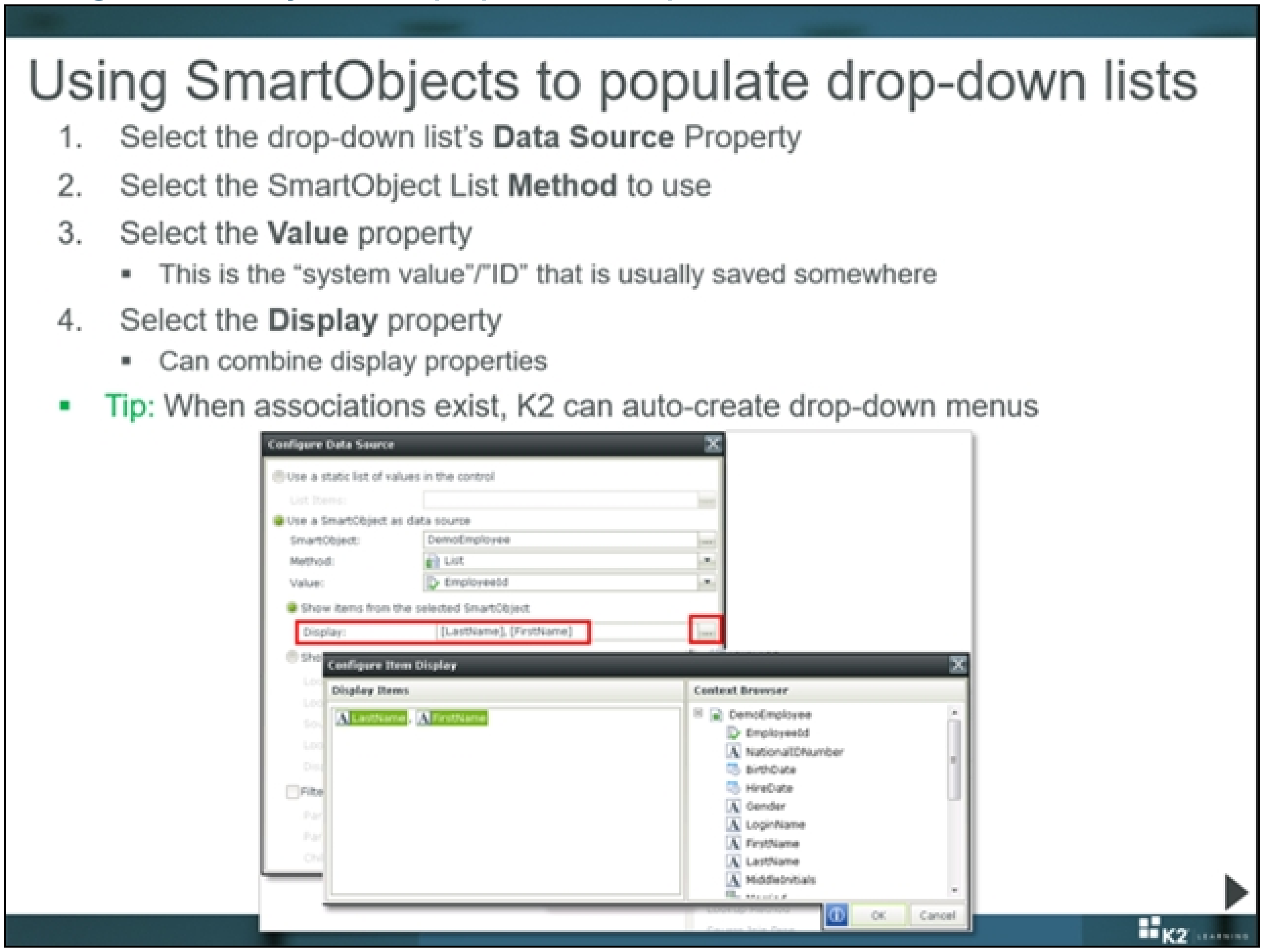
Video

In Part 2 we will look at intermediate-level concepts when working with Controls and Views, including:

- Using SmartObjects to populate the values in drop-down lists
- Configuring cascading drop-down lists
- Working with Lookup and Picker Controls and using SmartObjects to populate Lookup and Picker Controls
- Working with Edit-mode List Views, and defining the Edit Row vs. the Display Row
- Using aggregations and expressions in Item Views and List Views

At the end of Part 2, we will build two of the Views used in our application: the Sales Orders Header (item) View and the Sales Orders Items (list) View.

Using SmartObjects to populate drop-down lists



Using SmartObjects to populate drop-down lists

1. Select the drop-down list's **Data Source** Property
2. Select the SmartObject List **Method** to use
3. Select the **Value** property
   - This is the "system value"/"ID" that is usually saved somewhere
4. Select the **Display** property
   - Can combine display properties

- Tip: When associations exist, K2 can auto-create drop-down menus

Configure Data Source

Use a static list of values in the control

List Items:

Use a SmartObject as data source

SmartObject: DemoEmployee

Method: List

Value: EmployeeId

Show items from the selected SmartObject

Display: [LastName], [FirstName]

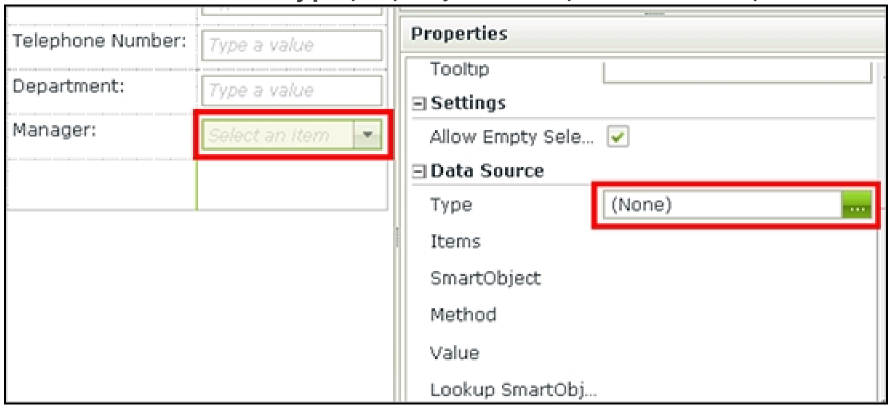
Configure Item Display

Display Items: LastName, FirstName

You can use any SmartObject List method to populate a drop-down menu with values. If there is a many-to-one relationship between SmartObjects, K2 can automatically generate a drop-down menu populated with the "lookup" SmartObject's data, but of course, you can also add drop-down lists to the View manually and configure the data source for the control quite easily.

Remember that when you have defined associations between SmartObjects, K2 Designer will generate drop-down lists based on those associations where appropriate. However, it is not required to define associations in order to populate drop-down lists. You can always set the data source and Rules for a drop-down list manually, as described in the sequence below.

1. Use the **Data Source Type** property for a drop-down list to open the data source picker window.

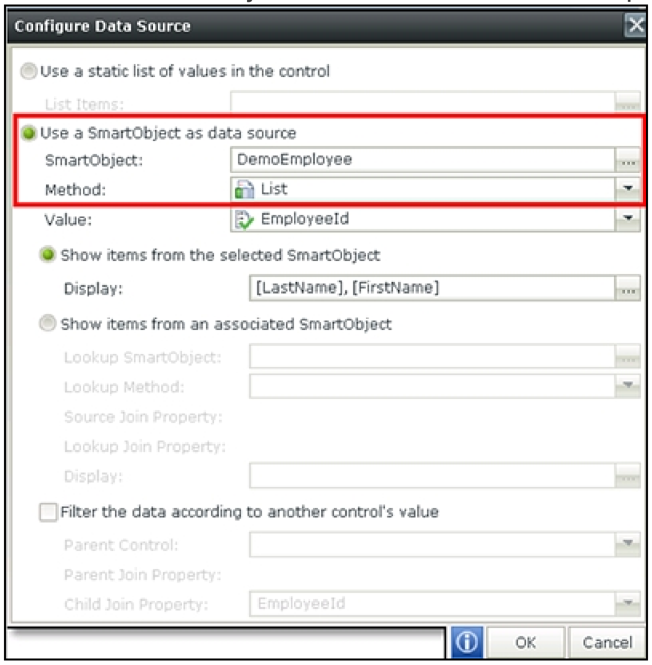


Telephone Number:	Type a value
Department:	Type a value
Manager:	Select an item

Properties

Tooltip	
⊟ Settings	
Allow Empty Sele...	✓
⊟ Data Source	
Type	(None)
Items	
SmartObject	
Method	
Value	
Lookup SmartObj...	

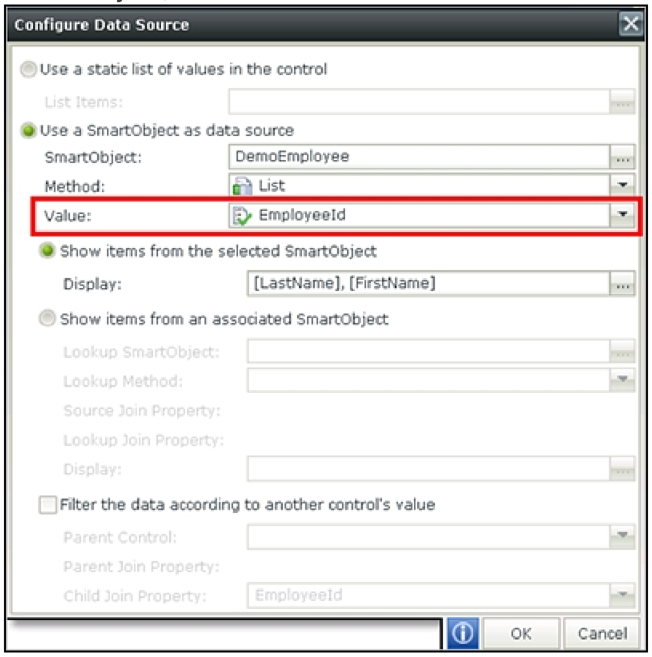
2. Select the SmartObjects and the Method to use to populate the drop-down list.



Configure Data Source

- [ ] Use a static list of values in the control
  - List Items:
- [✓] Use a SmartObject as data source
  - SmartObject: DemoEmployee
  - Method: List
  - Value: EmployeeId
  - [✓] Show items from the selected SmartObject
    - Display: [LastName], [FirstName]
  - [ ] Show items from an associated SmartObject
    - Lookup SmartObject:
    - Lookup Method:
    - Source Join Property:
    - Lookup Join Property:
    - Display:
  - [ ] Filter the data according to another control's value
    - Parent Control:
    - Parent Join Property:
    - Child Join Property: EmployeeId

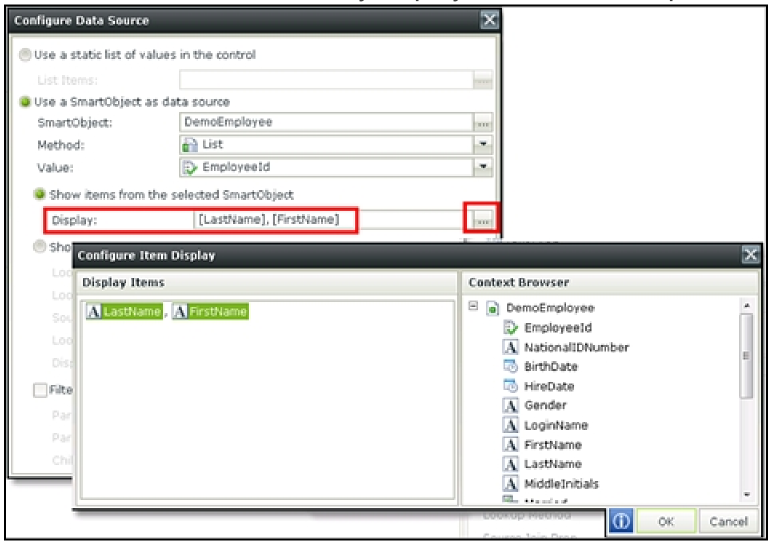
3. Select the Value property (this is usually something like an ID and is usually saved somewhere in the primary SmartObject, such as the CustomerID in the Sales Orders application).



Configure Data Source

- [ ] Use a static list of values in the control
  - List Items:
- [✓] Use a SmartObject as data source
  - SmartObject: DemoEmployee
  - Method: List
  - Value: EmployeeId
  - [✓] Show items from the selected SmartObject
    - Display: [LastName], [FirstName]
  - [ ] Show items from an associated SmartObject
    - Lookup SmartObject:
    - Lookup Method:
    - Source Join Property:
    - Lookup Join Property:
    - Display:
  - [ ] Filter the data according to another control's value
    - Parent Control:
    - Parent Join Property:
    - Child Join Property: EmployeeId

4. Select the Display property or properties (this is the value displayed to the user).You can combine multiple properties to make a more friendly display name, for example *LastName, FirstName*

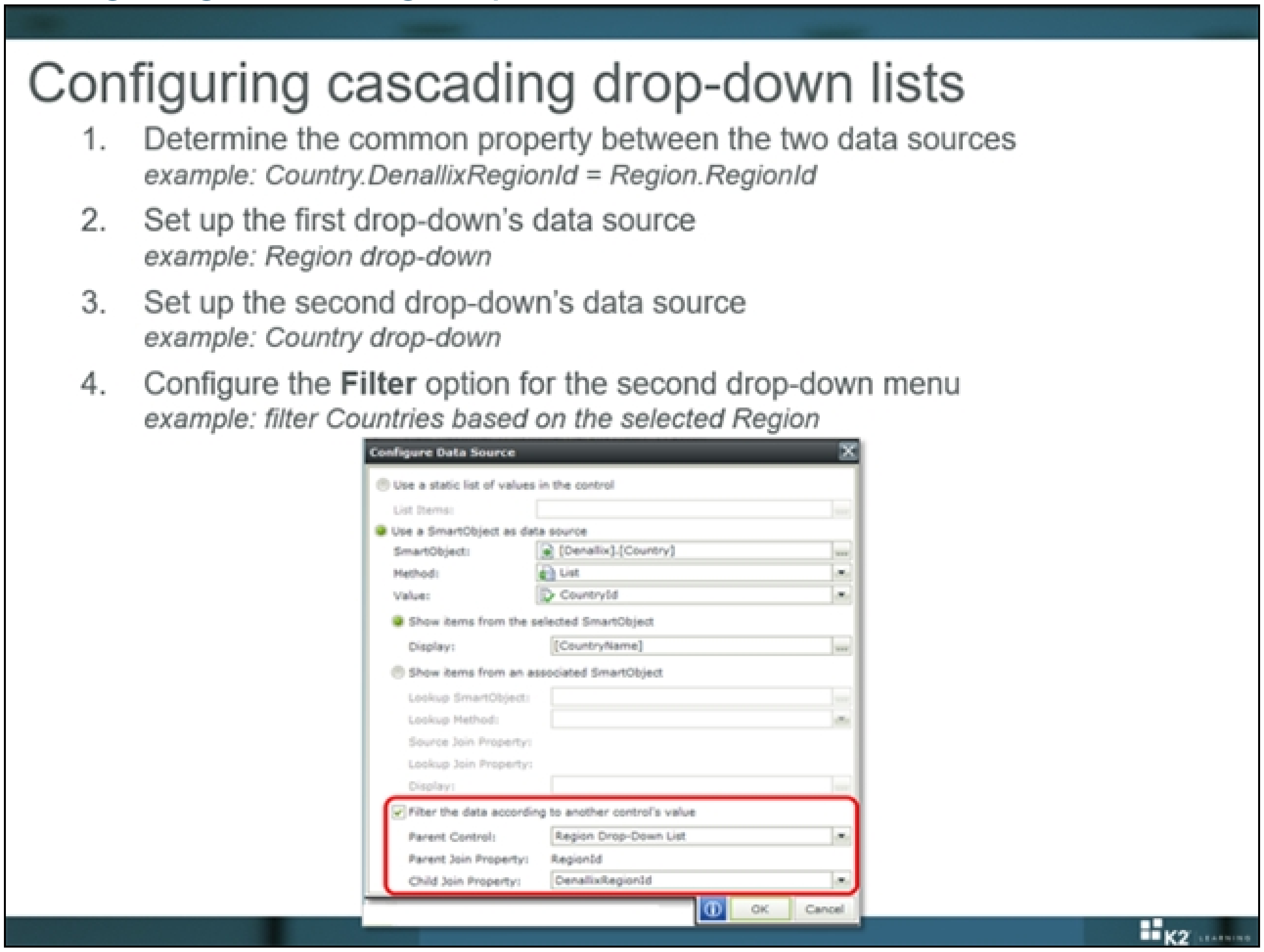


5. Behind the scenes, K2 will set up a Rule Action to populate the drop-down list with data when the View is initialized. You can edit that Action if you like.

Summary

1. Select the drop-down list's Data Source Property
2. Select the SmartObject List Method to use
3. Select the Value property (this is the “system value”/”ID” that is usually saved somewhere)
4. Select the Display property (you may combine several properties if you prefer)

Configuring cascading drop-down lists



Configuring cascading drop-down lists

1. Determine the common property between the two data sources
   example: Country.DenallixRegionId = Region.RegionId
2. Set up the first drop-down's data source
   example: Region drop-down
3. Set up the second drop-down's data source
   example: Country drop-down
4. Configure the **Filter** option for the second drop-down menu
   example: filter Countries based on the selected Region

Configure Data Source

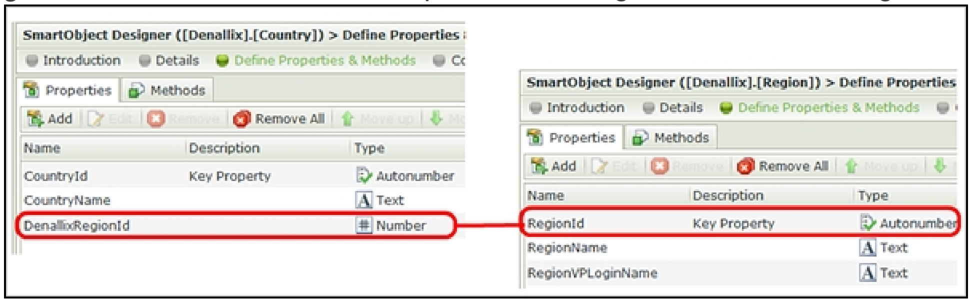
- Use a static list of values in the control
  - List Items:
- Use a SmartObject as data source
  - SmartObject: [Denallix].[Country]
  - Method: List
  - Value: CountryId
  - Show items from the selected SmartObject
    - Display: [CountryName]
  - Show items from an associated SmartObject
    - Lookup SmartObject:
    - Lookup Method:
    - Source Join Property:
    - Lookup Join Property:
    - Display:
- ✓ Filter the data according to another control's value
  - Parent Control: Region Drop-Down List
  - Parent Join Property: RegionId
  - Child Join Property: DenallixRegionId

It is a common requirement to implement "cascading drop-down lists" to make data entry easier for users. Here is an example: suppose there is a list of *Regions* (Americas, EMEA, APAC and so on) and a list of *Countries* (USA, Canada, UK, France, China, and Australia, etc.) defined for your organization. Perhaps the list of countries is large and we want to give users some mechanism to reduce the list of options, or maybe you want to ensure that when the user selects a region, they can only select a country from the list of countries assigned to that region. A cascading drop-down is a good approach to solve this requirement: let the user select a region first and then filter the list of countries based on the region that the user selected.

The "cascading drop-down lists" concept is easy to do with SmartForms by using the option to filter the data in the control according to another control's value. Let's see how to implement the requirement described above: we want to show a list of regions and then, depending on the region selected, show a list of countries in that region.

1. Ensure that there is a common property between the two data sources

Identify the common property that will allow you to filter the list of items in the second drop-down list based on the value selected in the first drop-down list. In this example, the common property is *Country.DenallixRegionId* and *Region.RegionId* - the list of countries will depend on the RegionID value of the region selected.



SmartObject Designer ([Denallix].[Country]) > Define Properties

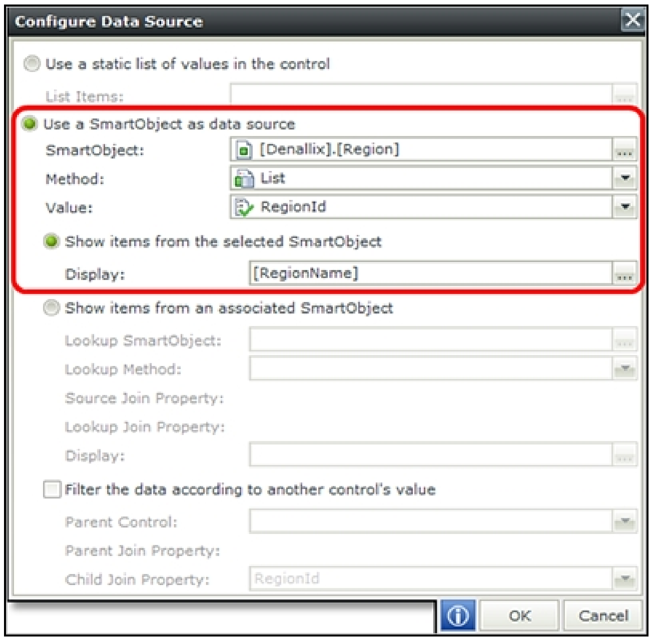
Name	Description	Type
CountryId	Key Property	Autonumber
CountryName		Text
DenallixRegionId		Number

SmartObject Designer ([Denallix].[Region]) > Define Properties

Name	Description	Type
RegionId	Key Property	Autonumber
RegionName		Text
RegionVPLoginName		Text

2. Set up the first drop-down's data source

Configure the first drop-down list's data source first (think of this as a "parent" drop-down). In this case, we are setting the Region drop-down data source. This is done in the same way as before: just select the drop-down menu and then configure the data source so that the Control is populated by the relevant List method.



Configure Data Source

Use a static list of values in the control

List Items:

[✓ Use a SmartObject as data source]

SmartObject:	[Denallix].[Region]
Method:	List
Value:	RegionId

[✓ Show items from the selected SmartObject]

Display:	[RegionName]

Show items from an associated SmartObject

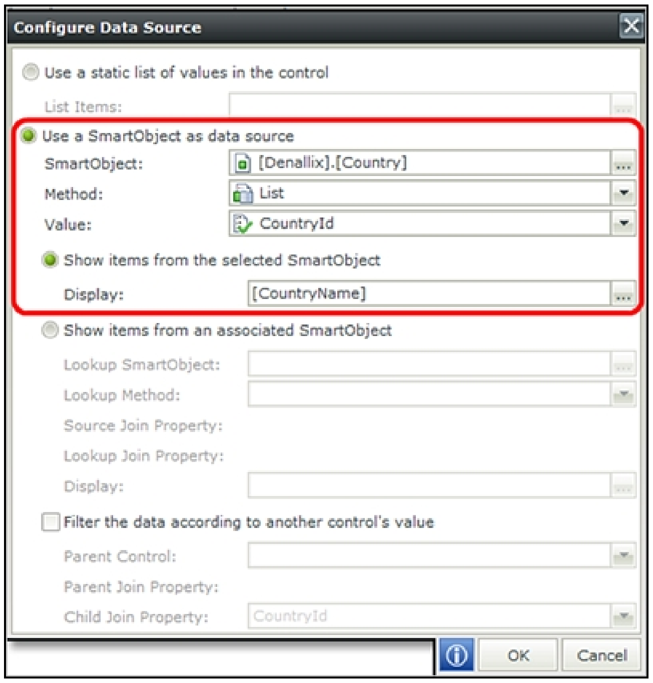
Lookup SmartObject:	
Lookup Method:	
Source Join Property:	
Lookup Join Property:	
Display:	

Filter the data according to another control's value

Parent Control:	
Parent Join Property:	
Child Join Property:	RegionId

3. Set up the second drop-down's data source

Next, configure the second drop-down list's data source (think of this as the "child" drop-down). Here, we are configuring the Control so that it gets the list of countries from the data store, the same as you did for the first drop-down.



Configure Data Source

Use a static list of values in the control

List Items:

[✓ Use a SmartObject as data source]

SmartObject:	[Denallix].[Country]
Method:	List
Value:	CountryId

[✓ Show items from the selected SmartObject]

Display:	[CountryName]

Show items from an associated SmartObject

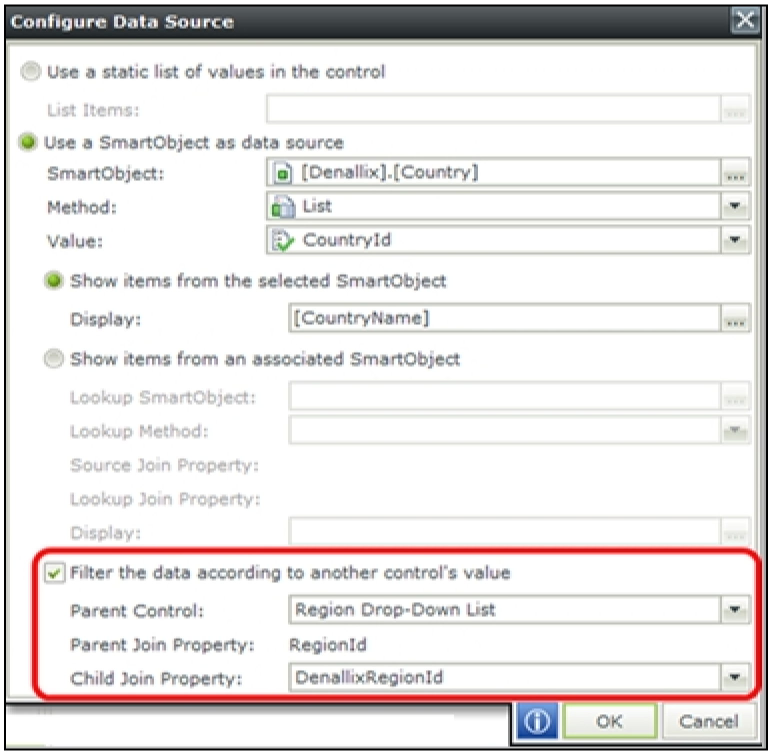
Lookup SmartObject:	
Lookup Method:	
Source Join Property:	
Lookup Join Property:	
Display:	

Filter the data according to another control's value

Parent Control:	
Parent Join Property:	
Child Join Property:	CountryId

4. Set up the Filter option for the next-level drop-down menu

Once you have set up each drop-down list's data source, go to the second drop-down menu, and select the **Filter the data according to another control's value** checkbox to filter the list of child drop-down values (Countries) based on the ID of the value selected in the parent drop-down (Regions), as shown below.



( ) Use a static list of values in the control

List Items:

(●) Use a SmartObject as data source

SmartObject: [Denallix].[Country]

Method: List

Value: CountryId

(●) Show items from the selected SmartObject

Display: [CountryName]

( ) Show items from an associated SmartObject

Lookup SmartObject:

Lookup Method:

Source Join Property:

Lookup Join Property:

Display:

[✓] Filter the data according to another control's value

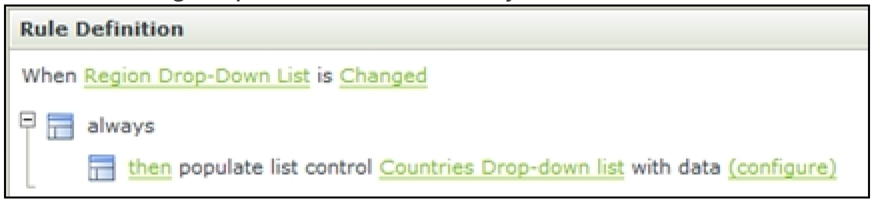
Parent Control: Region Drop-Down List

Parent Join Property: RegionId

Child Join Property: DenallixRegionId

Behind the scenes, K2 will add a Rule to the Form that will fire when the first drop-down list is changed. This Rule will tell the second drop-down list to refresh itself, and filter the values in the list based on the ID of the item selected in the first list. You don't need to define or configure this Rule yourself: it is done automatically for you, but feel free to adjust the Rule if necessary.

The cascading drop-down Rule added by K2



When Region Drop-Down List is Changed

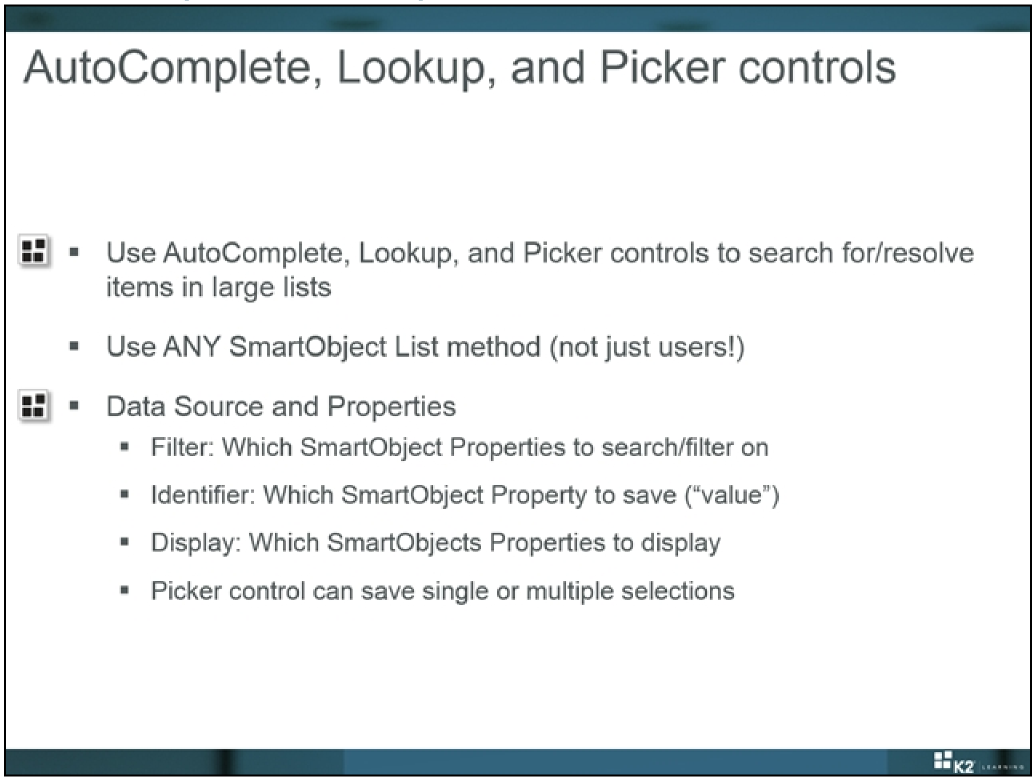
always

then populate list control Countries Drop-down list with data (configure)

Summary

1. Determine the common property between the two data sources
2. Set up the first (parent) drop-down's data source
3. Set up the second (child) drop-down's data source
4. Configure the Filter option for the second (child) drop-down menu based on the value selected in the first (parent) drop-down.

AutoComplete, Lookup, and Picker Controls



AutoComplete, Lookup, and Picker controls

- Use AutoComplete, Lookup, and Picker controls to search for/resolve items in large lists
- Use ANY SmartObject List method (not just users!)
- Data Source and Properties
  - Filter: Which SmartObject Properties to search/filter on
  - Identifier: Which SmartObject Property to save ("value")
  - Display: Which SmartObjects Properties to display
  - Picker control can save single or multiple selections

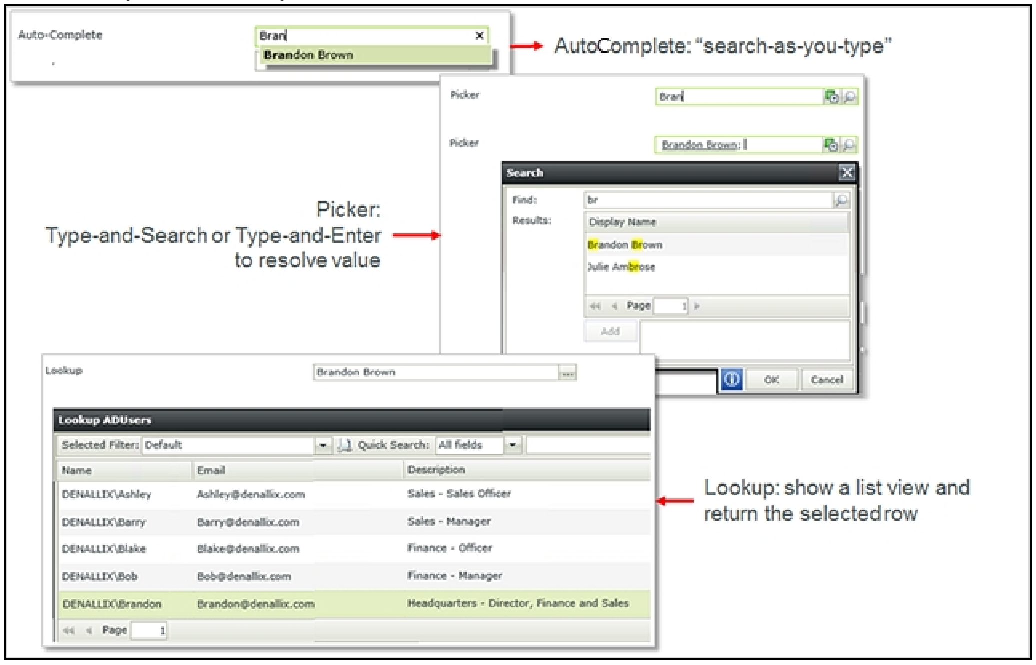
Video

While drop-down lists and cascading drop-down lists are a useful mechanism to allow end users to select data from small lists, sometimes you have a very large data set where it is more practical to give the user some way to search for and then select a value. There may also be a requirement to let the user select multiple items from a large set of data. Because drop-down lists are not very user-friendly for very large data sets and cannot allow multiple-item selections, SmartForms provides AutoComplete, Lookup and Picker Controls that can be used to search for records in a large data set.

The AutoComplete, Lookup and Picker Controls work very similarly. (The major differences are that the AutoComplete and Lookup Controls are used to select a single value based on a single search property, while the Picker Control can be used to select multiple items that can be searched for using multiple properties.)

The AutoComplete Control is a "search-as-you-type experience, where K2 will attempt to find records as the user types text into the Control. The Picker Control is more like a "resolve" or "search" experience where a user can just start typing a value into the Control and click the "Resolve" button or hit <ENTER> to find a record containing the value typed in, or they may click the search icon to launch a dialog box where they may search for records. The Lookup Control requires the user to open a Sub-view to select the record, and you (as the Form designer) need to create the Sub-view and configure the Rules to pass the data back from the Sub-view to the Lookup Control.

AutoComplete, Lookup and Picker Controls

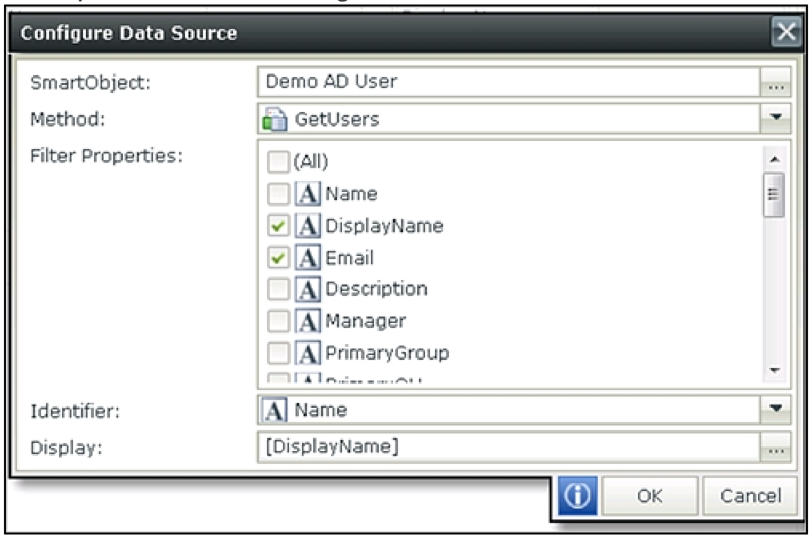

AutoComplete: "search-as-you-type"
Picker: Type-and-Search or Type-and-Enter to resolve value
Lookup: show a list view and return the selected row
Auto-Complete
Picker
Search
Find:
Results:
Display Name
Add
OK
Cancel
Lookup
Lookup ADUsers
Selected Filter: Default
Quick Search: All fields
Page

Name	Email	Description
DENALLIX\Ashley	Ashley@denallix.com	Sales - Sales Officer
DENALLIX\Barry	Barry@denallix.com	Sales - Manager
DENALLIX\Blake	Blake@denallix.com	Finance - Officer
DENALLIX\Bob	Bob@denallix.com	Finance - Manager
DENALLIX\Brandon	Brandon@denallix.com	Headquarters - Director, Finance and Sales

Picker Control

As with a drop-down list, a Picker Control uses a SmartObject List method as its data source. You can decide which properties should be displayed in the Control and which properties should be saved. The additional configuration you can select is which Filter Properties should be used when the user types a value into the search box. Consider the example configuration below:

Example data source configuration for a Picker Control



SmartObject: Demo AD User
Method: GetUsers
Filter Properties:
- [ ] (All)
- [ ] Name
- [x] DisplayName
- [x] Email
- [ ] Description
- [ ] Manager
- [ ] PrimaryGroup

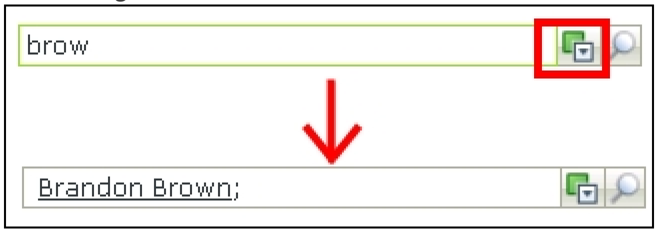
Identifier: Name
Display: [DisplayName]

Property	Purpose
SmartObject	Which SmartObject will be used to look up data for the Picker Control. In this example, we are using a **Demo AD User** SmartObject (based on an **AD User** Service Object), because we want to show a list of users from Active Directory. Note that you could use any SmartObject List method for the Picker and Lookup Controls, it is not just limited to searching for users.
Method	Which List method will be used to retrieve values from the Picker Control. In this example, we are using the **GetUsers** method of the Demo AD User SmartObject.
Filter Prop-erties	Which property or collection of properties K2 will search on when the user enters a value in the search text box or clicks the Resolve button. (You may select multiple properties to search on.) In this example, we want to search on the **DisplayName** and **Email** properties.

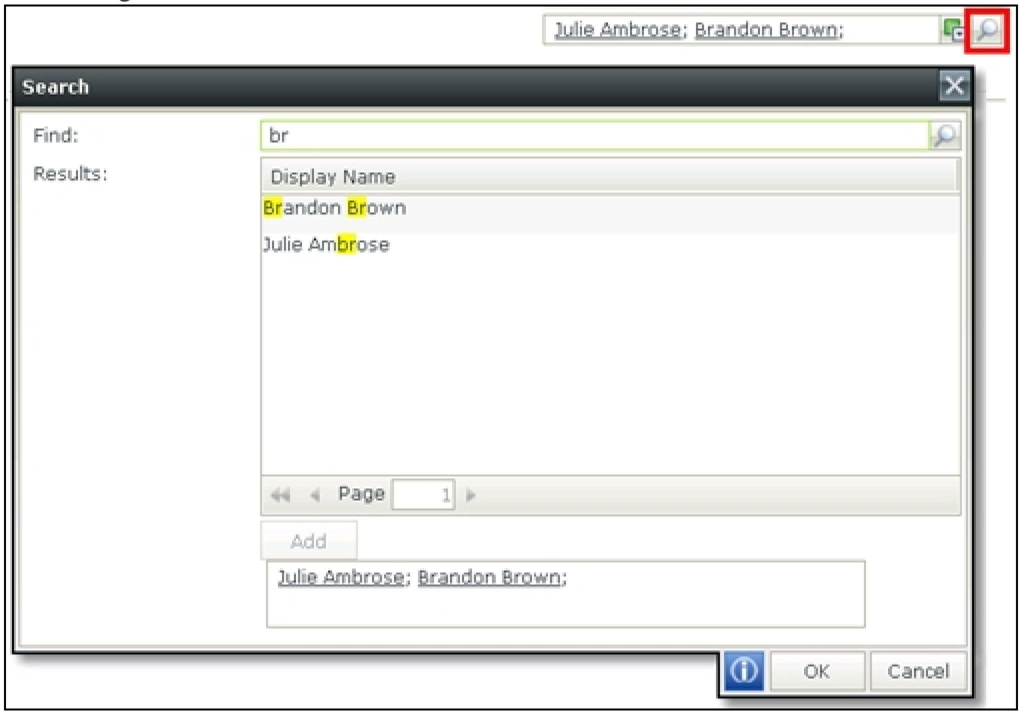
Identifier	Which property of the SmartObject will be saved by the Control. In this example, we want the Control to save the **Name** property.
Display	Which property (or combination or properties) K2 should display in the Picker Control once a value has been resolved. In this example, we want to display the user's **DisplayName** from Active Directory.

At runtime, the user can now use the Control to either Resolve or Search for records, as shown below:

Resolving a value with the Picker Control



Searching for records with the Picker Control

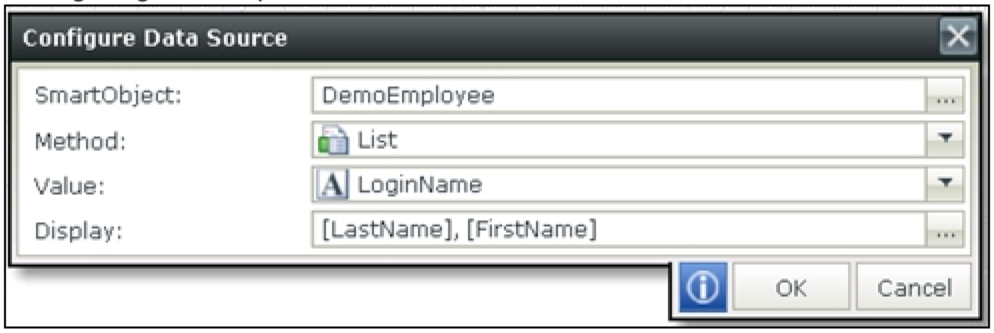


The Picker Control can save multiple items with a delimiter or as an XML structure, just select the appropriate option in the Control's Properties pane.

Lookup Control

A Lookup Control is similar to a Picker Control, but is used to search a data source based on a single property and select a single record. Note that users cannot directly enter data into a Lookup Control. Instead, when the Control is clicked, a List View will be launched as a Sub-view. The user will select a row from the Sub-view and that value will be returned to the Lookup Control. You need to configure the List View and the Rules to transfer data between the Sub-view and the parent View. See the help topic Lookup Control for more information.

Configuring a Lookup Control's data source



Configure Data Source

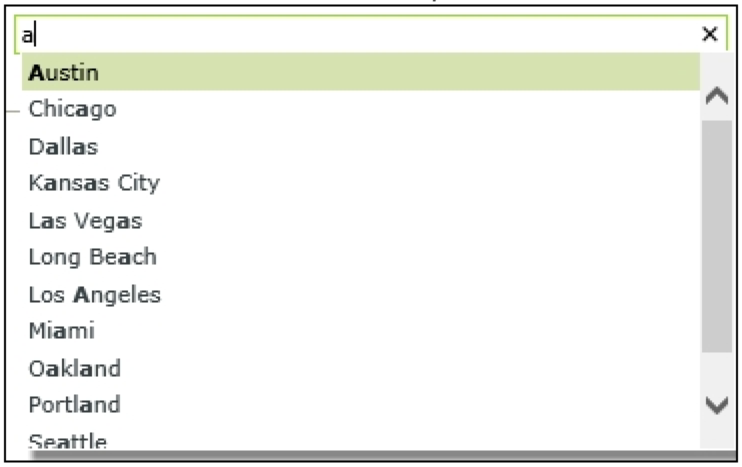
SmartObject:	DemoEmployee
Method:	List
Value:	LoginName
Display:	[LastName], [FirstName]

Property	Purpose
SmartObject	Which SmartObject will be used to look up data for the Lookup Control. In this example, we are using a **DemoEmployee** SmartObject. Note that you could use any SmartObject List method for the Lookup Control, it is not just limited to searching for users.
Method	Which List method will be used to retrieve values from the Picker Control. In this example, we are using the **List** method of the Demo Employee SmartObject.
Value	Which property of the SmartObject will be saved by the Control. In this example, we want the Control to save the **LoginName** property.
Display	Which property (or combination or properties) K2 should display in the Lookup Control once a record or value has been selected. In this example, we want to combine two properties (the **LastName, FirstName**) so that it is easier for our end users to determine whose name they have selected.

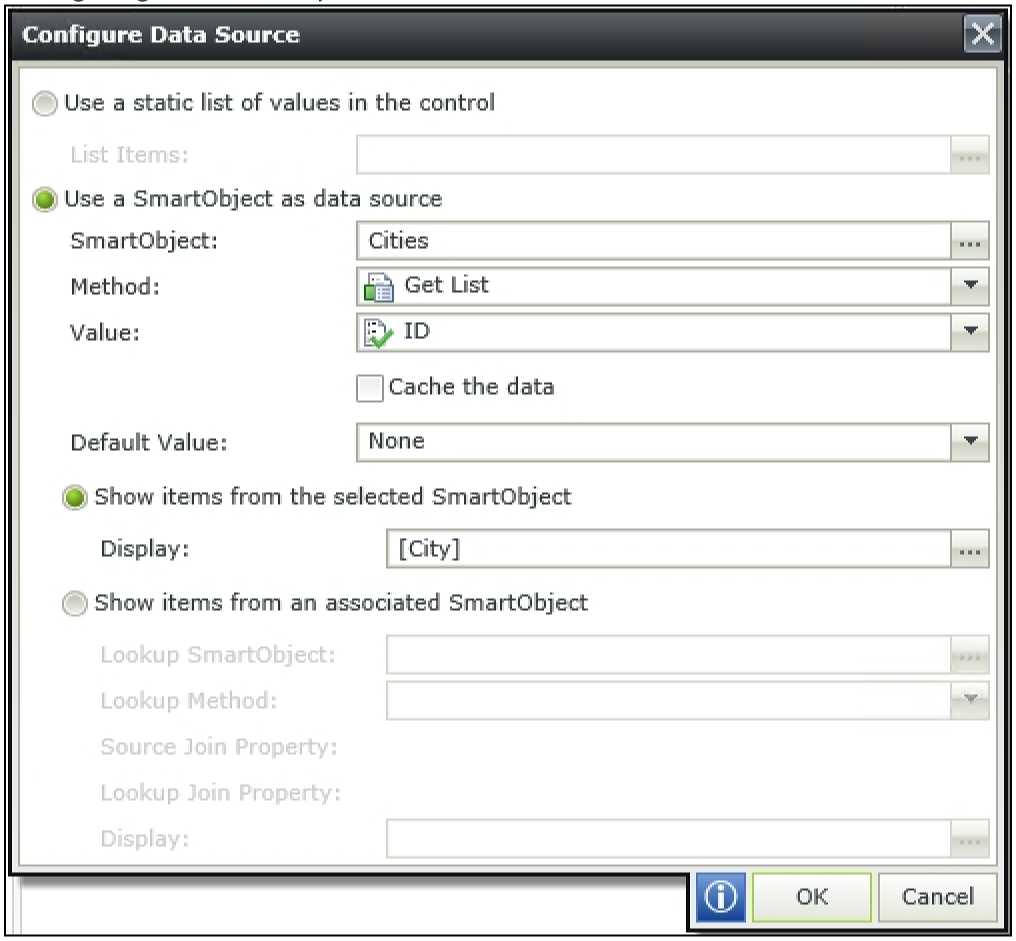
AutoComplete Control

The AutoComplete Control provides suggestions while you type into the field. The Control is in essence a text box that can be bound to a SmartObject which, at runtime, filters data in a drop-down by a "Contains" operator when a user enters a value.

Runtime behavior of the AutoComplete Control



Configuring an AutoComplete Control's data source


Configure Data Source

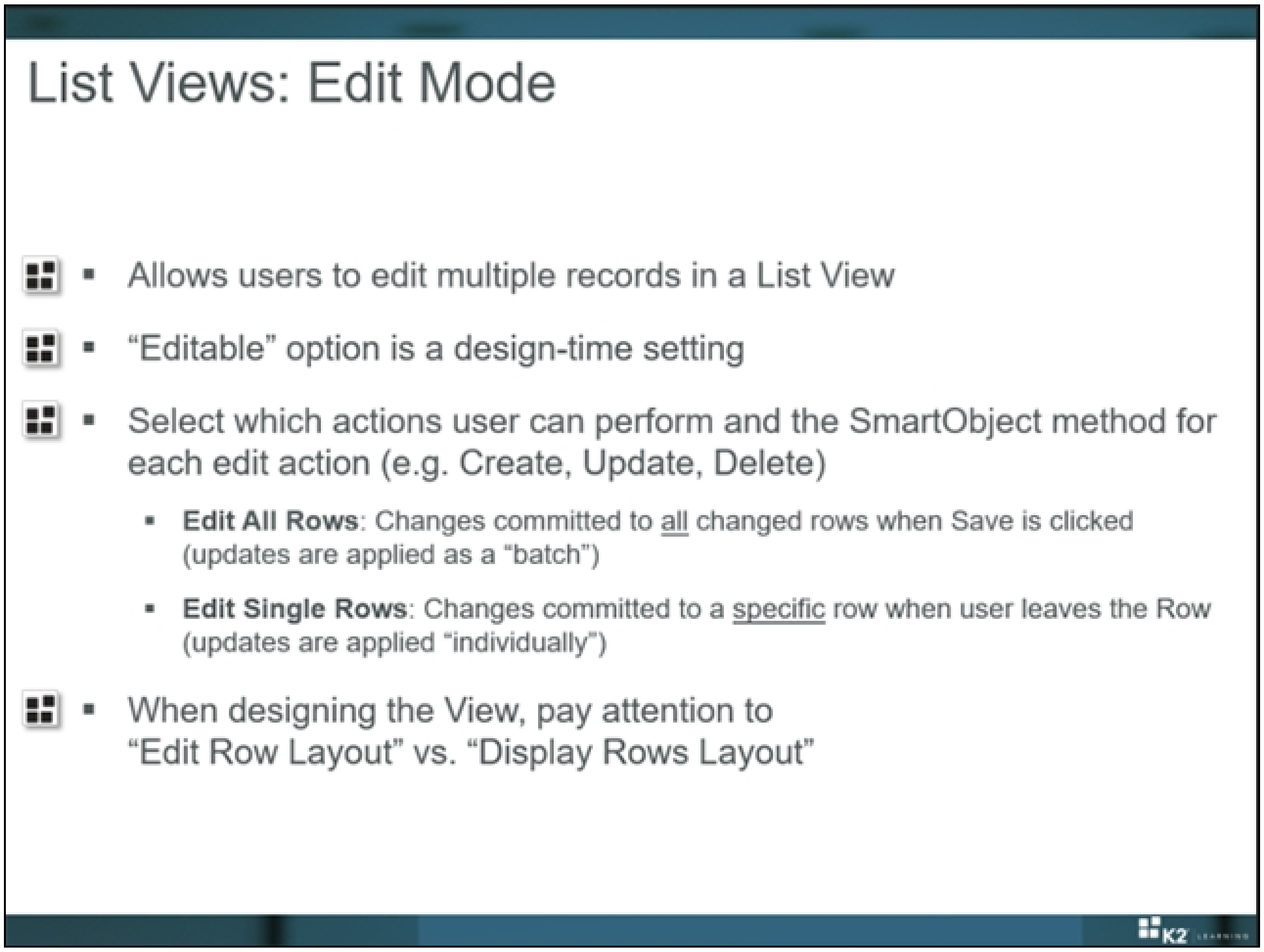
- Use a static list of values in the control
  - List Items:
- [✓ Use a SmartObject as data source]
  - SmartObject: Cities
  - Method: Get List
  - Value: ID
  - Cache the data
  - Default Value: None
  - [✓ Show items from the selected SmartObject]
    - Display: [City]
  - Show items from an associated SmartObject
    - Lookup SmartObject:
    - Lookup Method:
    - Source Join Property:
    - Lookup Join Property:
    - Display:

Property	Purpose
Use a list of static values...	Select this option and click the ellipsis next to List Items to configure the list of values for the Control.
SmartObject	Which SmartObject will be used to look up data for the AutoComplete Control. In this example, we are using the **Cities** SmartObject. Note that you could use any SmartObject List method for the AutoComplete Control.
Method	Which List method will be used to retrieve values. In this example, we are using the **Get List** method of the Cities SmartObject.
Value	Which property of the SmartObject will be saved by the Control. In this example, we want the Control to save the **ID** property.
Cache the Data	When selected, the SmartObject data will be cached to allow for client side filtering, enabling the data to be available offline. This can help performance, but at the expense of retrieving live data from the data provider.
Default Value	(Does not apply to this Control)
Show items from the selected SmartObject	Shows items from the selected data source SmartObject.
Display	Which property (or combination or properties) K2 should display in the AutoComplete Control once a value has been resolved. In this example, we are displaying the City name in the Control.
Show items from an associated SmartObject	Shows items from an associated SmartObject instead of the data source SmartObject. If you select this option, configure the appropriate values for the associated SmartObject.

Summary

- AutoComplete, Lookup and Picker Controls are often used to filter and search from large data sets
- AutoComplete: lets you type and it will attempt to find records that match (e.g. "search-as-you-type”)
- Lookup: launch a list Sub-view when the Control is clicked. Then select a row from the Sub-view and that value will be returned to the Lookup Control.
  - Lookup Controls require you to create a List View Sub-view and then lookup Rules to perform the data transfer
- Picker: lets you type and press <ENTER> to resolve the values, or use a search function to search on any of the selected search properties.
  - You can either search for items or type the first few letters and hit “search/<ENTER>”
  - The Picker Control can save multiple items with a delimiter or as an XML structure

List Views: Edit Mode



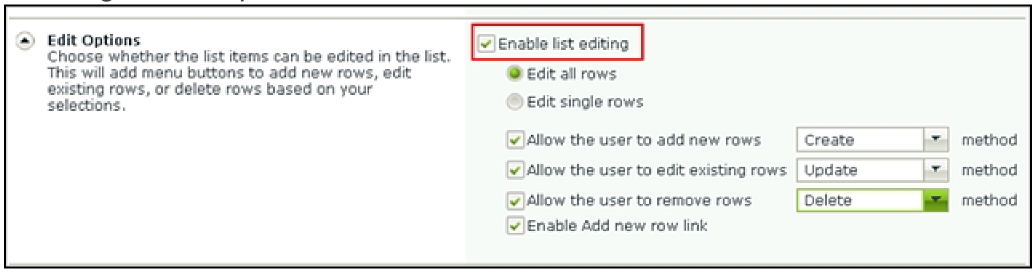
List Views: Edit Mode

- Allows users to edit multiple records in a List View
- "Editable" option is a design-time setting
- Select which actions user can perform and the SmartObject method for each edit action (e.g. Create, Update, Delete)
  - **Edit All Rows**: Changes committed to all changed rows when Save is clicked (updates are applied as a "batch")
  - **Edit Single Rows**: Changes committed to a specific row when user leaves the Row (updates are applied "individually")
- When designing the View, pay attention to "Edit Row Layout" vs. "Display Rows Layout"

List Views can be displayed in two modes: Read-only mode and Edit mode. The Edit mode is very useful for batch-style data entry and maintenance because the user can edit multiple items in the list without having to open a "details" page for each item. As the designer, you will decide how those changes should be applied: either when the user leaves the row being edited, or when the user clicks a button to commit all the changed rows at the same time.

To enable a List for Edit mode, select the **Enable List editing** option when designing the List View:

Enabling the Edit option for a List View



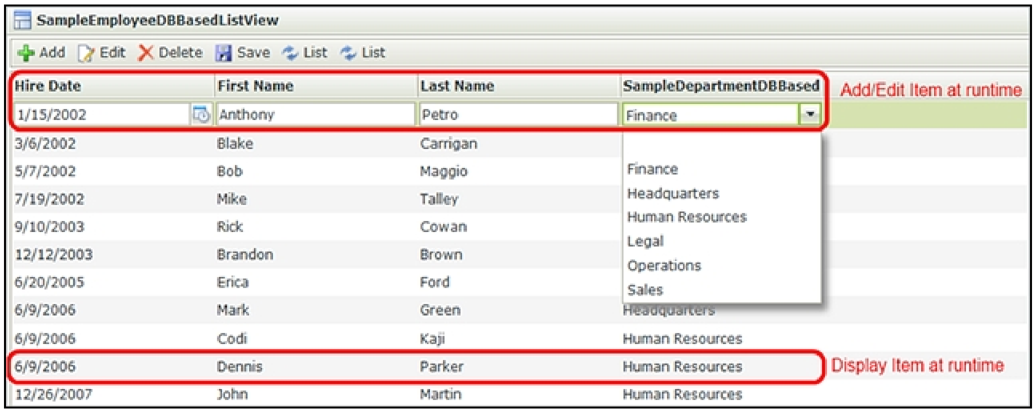
Edit Options
Choose whether the list items can be edited in the list. This will add menu buttons to add new rows, edit existing rows, or delete rows based on your selections.

[✓] Enable list editing
● Edit all rows
○ Edit single rows

[✓] Allow the user to add new rows — Create method
[✓] Allow the user to edit existing rows — Update method
[✓] Allow the user to remove rows — Delete method
[✓] Enable Add new row link

When this setting is enabled, the runtime experience allows the end user to click on rows in the List View and then add, update or delete the selected row.

Runtime behavior of an editable List View



Hire Date	First Name	Last Name	SampleDepartmentDBBased
1/15/2002	Anthony	Petro	Finance
3/6/2002	Blake	Carrigan	
5/7/2002	Bob	Maggio	Finance
7/19/2002	Mike	Talley	Headquarters
9/10/2003	Rick	Cowan	Human Resources
12/12/2003	Brandon	Brown	Legal
6/20/2005	Erica	Ford	Operations
6/9/2006	Mark	Green	Sales
6/9/2006	Codi	Kaji	Human Resources
6/9/2006	Dennis	Parker	Human Resources
12/26/2007	John	Martin	Human Resources

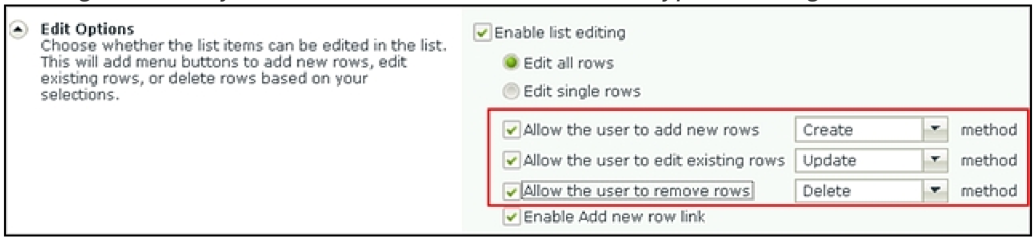
When you enable list editing, you will need to select which Rules/Buttons should be available to the user, and indicate which SmartObject methods K2 should use when the user commits the changes to the list. In the majority of cases, K2 can automatically select the correct methods, but just double-check these and select the correct method, if appropriate.

The **Edit all rows** radio button is a shortcut to set up the editable list so that a 'Save' button is shown on the list. The Save button contains the logic to update all of the edited rows in a “batch” mode and will call the Create method for each new row, the Update method for each row that was changed and the Delete method for each row that was deleted. Essentially, this option will commit all of the changes to the SmartObjects only when the user clicks the Save button. In other words, the updates are applied as a “batch” so that when the user clicks “Save”, K2 will batch up all the operations and call the appropriate method for each changed row as a batch operation. Think of it as an "apply-all-changes-at-same-time" operation.

The **Edit Single Rows** radio button will inject Rules into the list that will immediately commit the changes to the list when the user completes an action on a single row: calling the Create method if the row is new, the Delete method is the row is removed and the Update method if the row values are updated. Think of this option as inline update: the changes are committed as soon as the user leaves the row. In other words, the updates are applied “individually” as soon as the user clicks out of the row they edited. Think of this as a "one-at-a-time" operation.

In most cases, the **Edit all rows** option is used, so that the user can make changes to the list “locally” and then update them all at the same time in the data store by clicking a button.

Setting the edit style and the methods used for each type of change

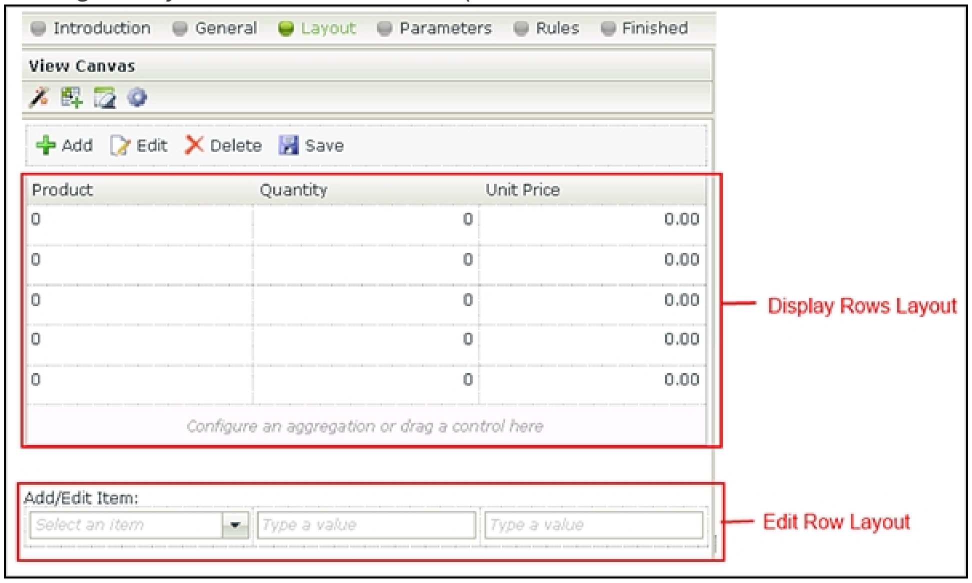


Edit Options
Choose whether the list items can be edited in the list. This will add menu buttons to add new rows, edit existing rows, or delete rows based on your selections.

✓ Enable list editing
● Edit all rows
○ Edit single rows
✓ Allow the user to add new rows: Create method
✓ Allow the user to edit existing rows: Update method
✓ Allow the user to remove rows: Delete method
✓ Enable Add new row link

When editing the layout for the List View at design time, you will notice two sections in the layout screen. The top part (known as the List Display table) determines how the read-only items in the list will be displayed. The bottom section (the Add/Edit Item section) will determine how a row is displayed when the user is editing an item in the list. You will most likely need to edit this display and potentially change the control types for the Controls in the Add/Edit item layout. In the example below, we have changed the **Product** text box to a drop-down menu and configured the data source for that drop-down menu so that the user can just select from a drop-down list when editing the row.

Editing the layout of an Editable List. (Note the difference between the Edit Row layout and the Display Rows layout)



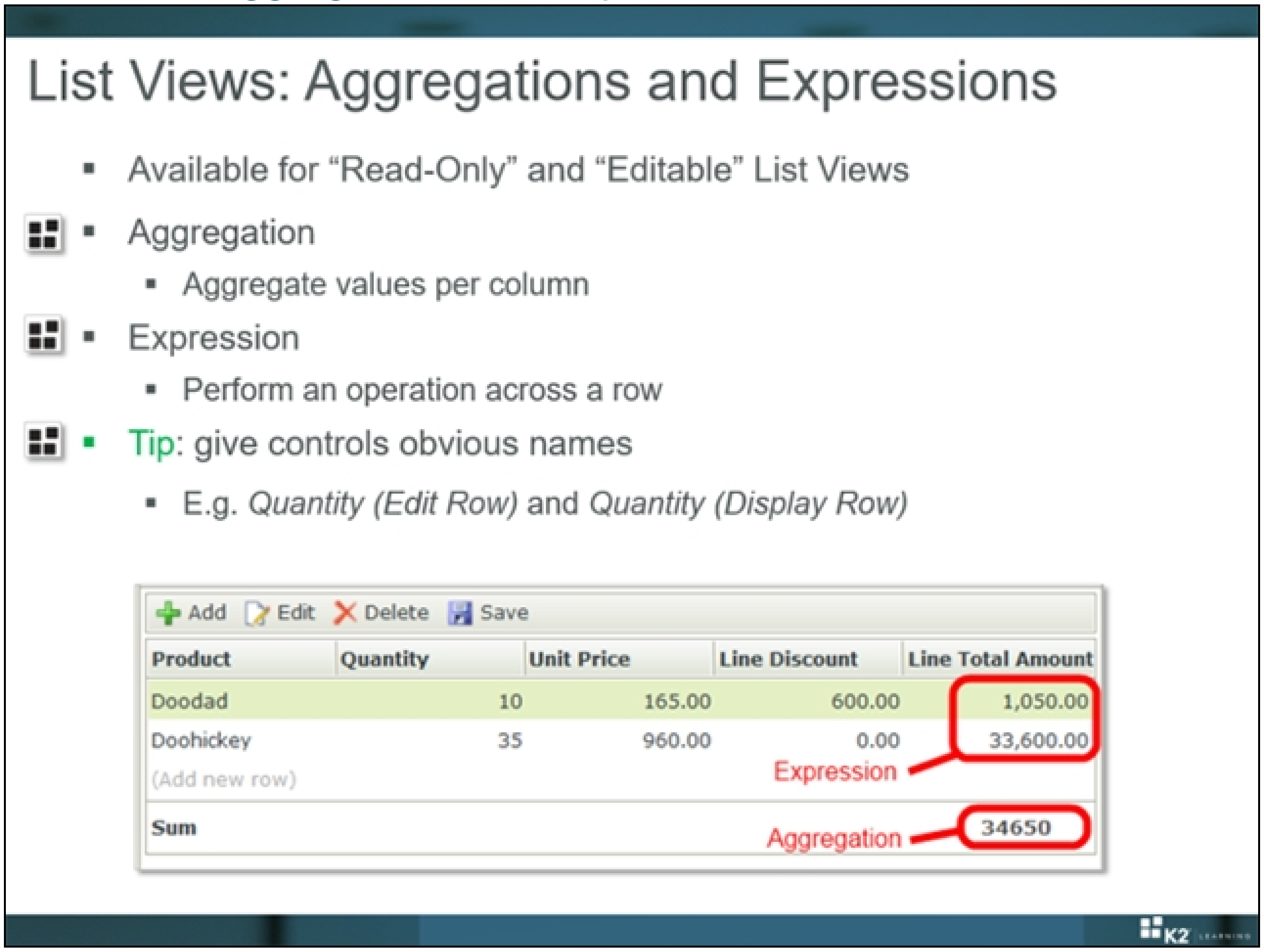
Product	Quantity	Unit Price
0	0	0.00
0	0	0.00
0	0	0.00
0	0	0.00
0	0	0.00

Tip
Use descriptive names for the Controls in the edit row layout (e.g. *Unit Price Edit Text box*) and display rows layout (e.g. *Unit Price Display Label*) so you know which Controls are in which type of row layout - this is useful when you are configuring Rules and expressions and need to know which Controls should be targeted.

Summary

- List Views in Edit mode allow users to Create/Edit/Delete multiple records using a List View
- This is common in data capture scenarios where users need to update multiple records in one screen
- The Editable mode is a design-time setting you apply when you create the List View
- You need to tell K2 which actions the user may perform (e.g. Create, Update, Delete) and which SmartObject method K2 must call for each of those Actions.
  - “Edit All Rows" setting: the changes committed to all changed rows when Save is clicked.
  - “Edit single rows” setting: the changes are committed to a specific Row when the user exists the row.
  - Edit row layout is where users capture data (row is being edited)
  - Display row layout is how the row will look when the row is in display mode (row is not being edited)

List Views: Aggregations and Expressions



List Views: Aggregations and Expressions

- Available for "Read-Only" and "Editable" List Views
- Aggregation
  - Aggregate values per column
- Expression
  - Perform an operation across a row
- Tip: give controls obvious names
  - E.g. *Quantity (Edit Row)* and *Quantity (Display Row)*

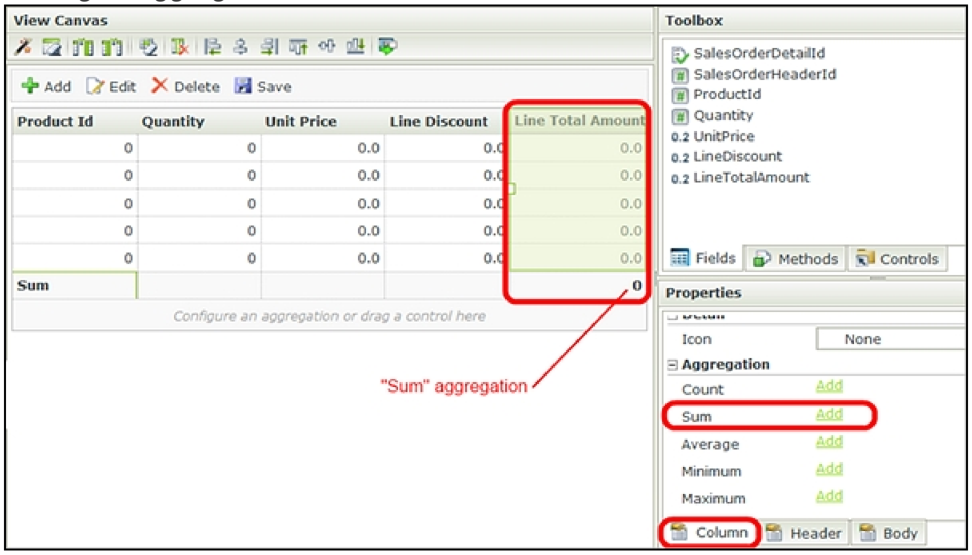
Product	Quantity	Unit Price	Line Discount	Line Total Amount
Doodad	10	165.00	600.00	1,050.00
Doohickey	35	960.00	0.00	33,600.00
(Add new row)				
Sum				34650

You can add expressions and aggregations to List Views to calculate values. Aggregations are actually a type of expression, but they are configured slightly differently since they usually apply to List Display View columns. Let's look at each approach separately.

Aggregation

Aggregation values are normally applied to a column in a List Display View (for example, calculate the total amount for all values in a column). It is very easy to configure an aggregation: just select the column you want to aggregate, and then click the relevant option under the Aggregation heading for the column properties, as shown below. In this example, we want to show the **Sum** of the **Line Total Amount** column.

Adding an aggregation for a column



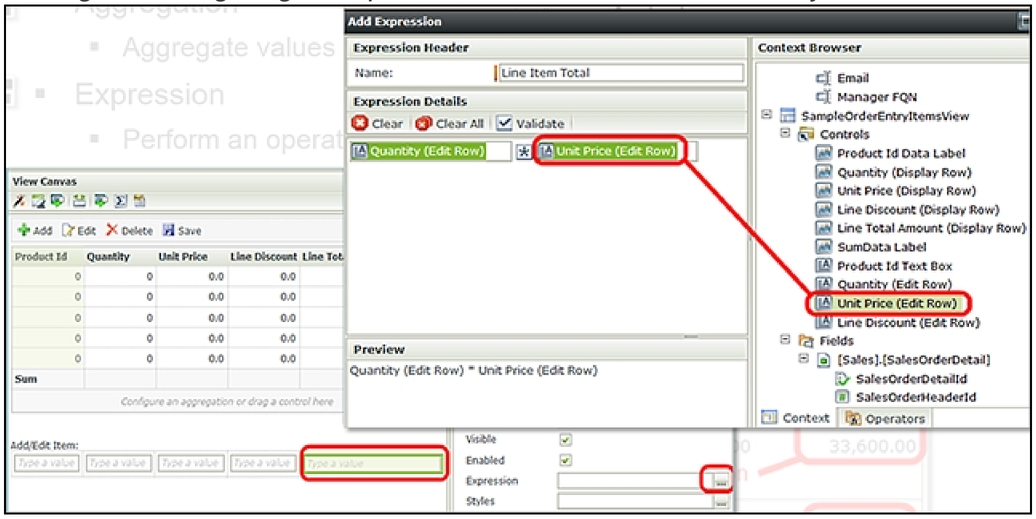
Once you have configured the aggregation, you can format and style the Control or move it around on the layout of the View. Note that the aggregation will automatically be re-evaluated when a value in the list changes. If you were using an editable list, as soon as you move outside of the current list item, the aggregation will be recalculated.

Expressions

You can also add expressions to List Views. These are normally used to perform some calculation across a line item, although you could also define expressions for columns. At runtime, when a value used in the expression changes, K2 will automatically recalculate the expression.

Here is an example: suppose you have an “Order Details” table and you want to auto-calculate the total amount for a line item by multiplying the **Quantity** with the **Unit Price**. This is very easy: just select the control where the expression should be displayed, then use the expression builder to define the new expression, as shown below.

Adding and configuring an expression to the Add/Edit item row layout



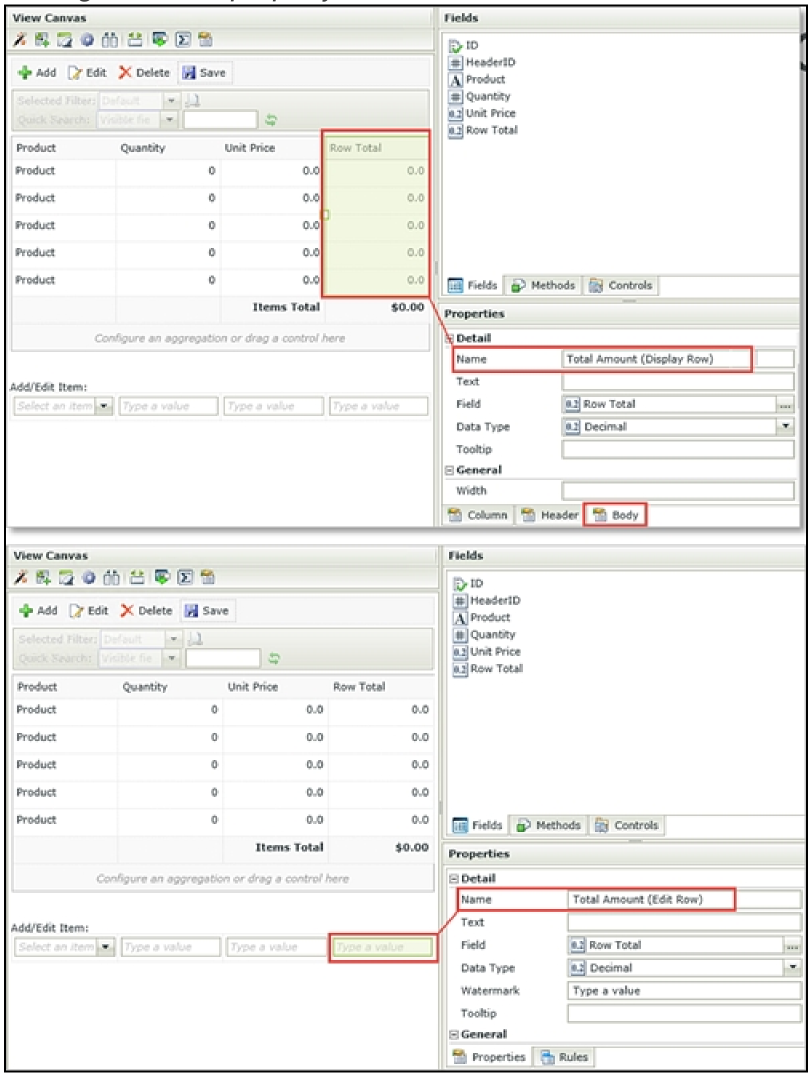
Note that you should select the correct Control when configuring the expression. When you have an editable list, there are actually two Controls for each column: one Control is used for the “List Display” layout, and the other Control is used for the “Edit/Add Item Row” layout. If you are defining an expression that should calculate when the user adds or edits an item, make sure you drag and drop the “Add/Edit Item Row” Controls into the expression, not the Control in the “List Display” layout.

In the example above, we wanted to calculate the line item total amount when the user adds a new item, so we use the “Edit Row” Controls in the expression. (If we were creating an expression for items in the display row, we would use the “Display Row” Controls.)

We highly recommend that you give the Controls clear and descriptive names, especially when using lists in Edit mode. This will help to avoid confusion when defining Rules and expressions.

To set a Control's name, change the **Name** property of the Control. To change the name for a Control in the List Display layout, use the **Body** tab of the Properties Pane to access and change the **Name** property. For a control in the Add/Edit Item layout, just select the Control and change the **Name** property on the Properties Pane. See the screen shots below for examples.

Setting the Name property for Controls on an editable List View



Product	Quantity	Unit Price	Row Total
Product	0	0.0	0.0
Product	0	0.0	0.0
Product	0	0.0	0.0
Product	0	0.0	0.0
Product	0	0.0	0.0
		Items Total	$0.00

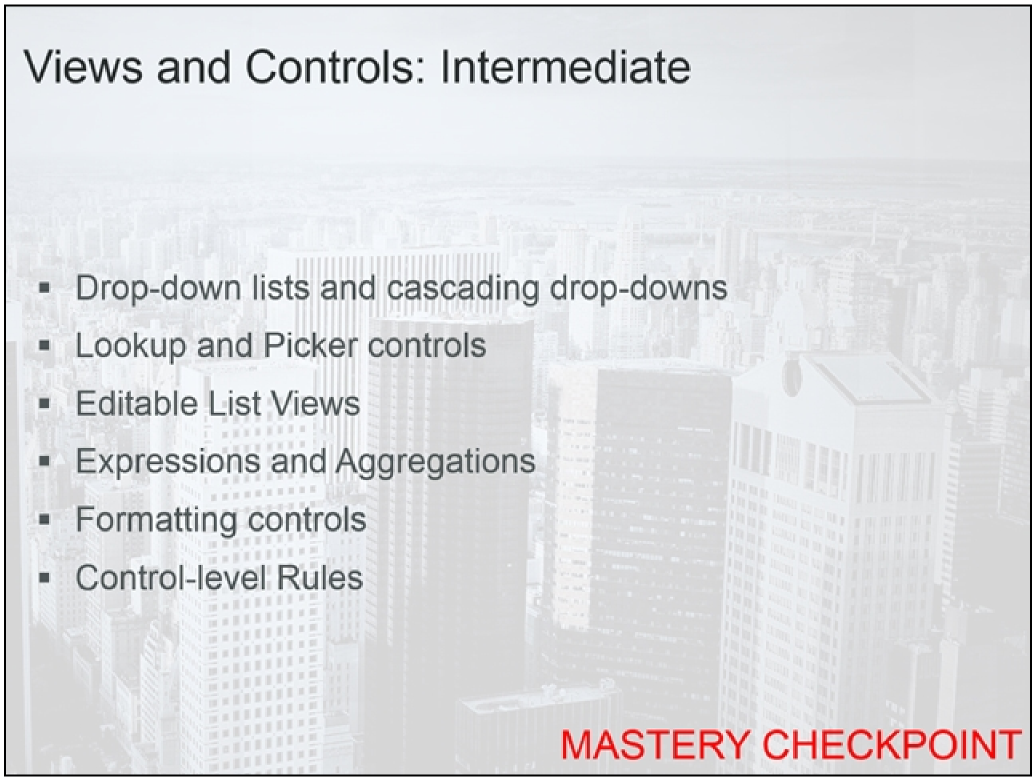
Detail	
Name	Total Amount (Display Row)
Text	
Field	Row Total
Data Type	Decimal
Tooltip	
General	
Width	

Detail	
Name	Total Amount (Edit Row)
Text	
Field	Row Total
Data Type	Decimal
Watermark	Type a value
Tooltip	
General	

Summary

- You can use aggregations and expressions in both Edit-mode and Display-only List Views
- Aggregations are sums for columns (e.g. total for a column or average for a column)
- Expressions are usually used to perform a calculation across a row (e.g. quantity * amount)
- It is a good idea to give Controls in the Edit Row layout obvious names so that when you define the expressions, it is easier to know which Controls are being used in the expression.

MASTERY CHECKPOINT: Views and Controls: Intermediate



Views and Controls: Intermediate

- Drop-down lists and cascading drop-downs
- Lookup and Picker controls
- Editable List Views
- Expressions and Aggregations
- Formatting controls
- Control-level Rules

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 2 of this module. If you are attending instructor-led training, this is an opportunity to answer any questions around working with Editable List Views, drop-down lists, Picker Controls and so on.

After completing Part 2 you should know:

- How to create editable List Views
- How to add drop-down lists to a View and populate them with a SmartObject
  - How to implement cascading drop-down lists
- How to configure and use Picker Controls and AutoComplete Controls
- How to define expressions (Item Views and List Views) and adding aggregations to List Views
- Defining Control-level Rules to perform an action when a Control is changed

Knowledge-check questions

Q: Can any SmartObject List method be used as the data source for a Picker Control, or only methods from "Users" type SmartObjects?
Reveal answer**A:** Any SmartObject List method can be used as the data source for a Picker Control.

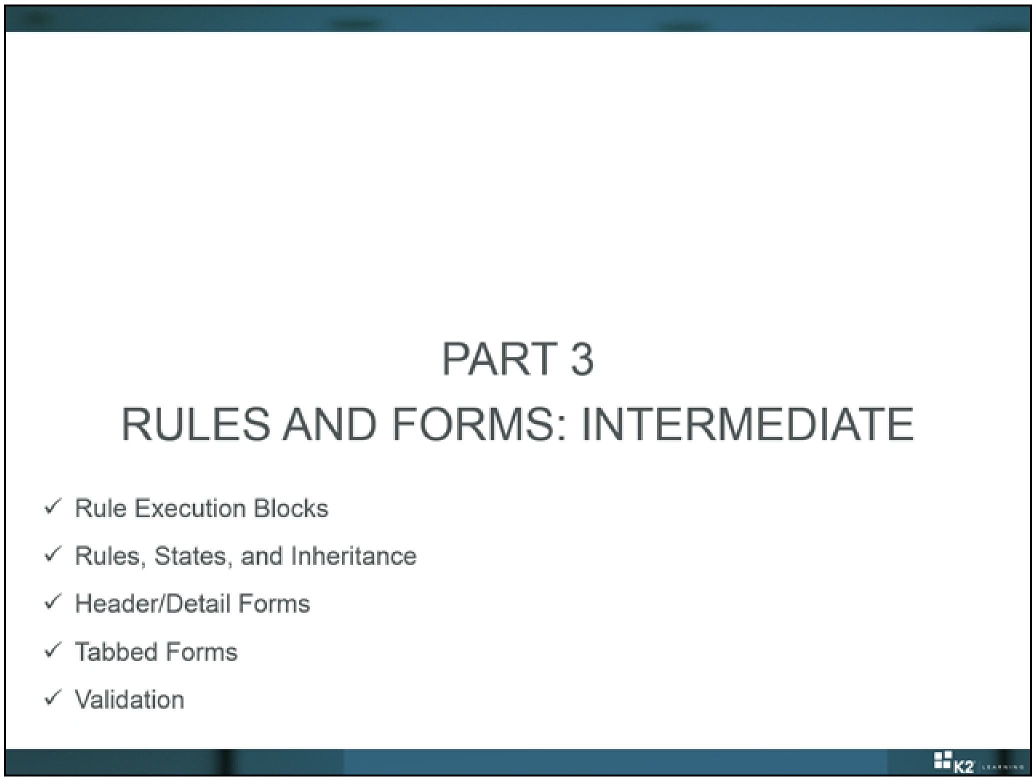
Q: What is the difference between the AutoComplete Control and the Picker Control?
Reveal answer**A:** AutoComplete is a search-as-you-type Control, the Picker is a type and then click search or resolve Control.

Q: What is the difference between aggregations and expressions?
Reveal answer**A:** Aggregations are a calculation on a column in a List View. Expressions are calculations that can be inside or outside a List View.

Q: How do you set a List View to be editable?
Reveal answer**A:** In the List View Settings screen, select the "Enable List Editing" setting.

Q: What would you do differently in the Header View and Items View that we created in this exercise? Would you use different Controls? Why?
A: (discussion question)

PART 3: Rules and Forms: Intermediate


PART 3

RULES AND FORMS: INTERMEDIATE

✓ Rule Execution Blocks

✓ Rules, States, and Inheritance

✓ Header/Detail Forms

✓ Tabbed Forms

✓ Validation

Video

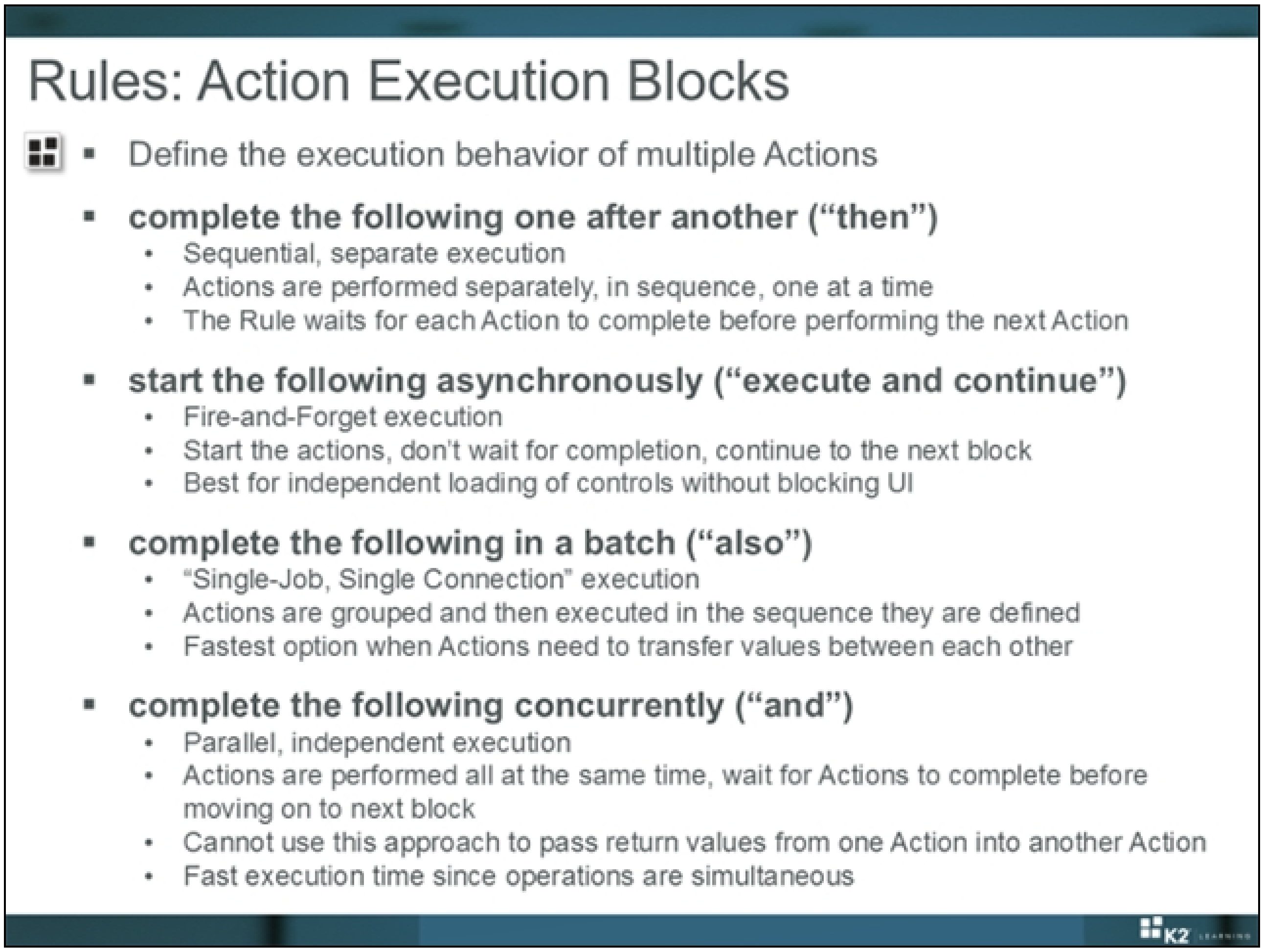
In Part 3 we will look at some more complex things you can do with Rules and Forms, such as:

- Using Rule Execution Blocks to control how Rule Actions are grouped, ordered and executed
- How States and Inheritance affect Rules
- Building header/detail Forms
- Creating Tabbed Forms
- More advanced validation approaches

At the end of Part 3, we will create the master-detail Form that hosts the Header View and the Details View that were created in Exercise 2.

Note
Remember that Rules can exist on Views and Controls as well, we're just grouping them into Part 3 because of the way the lab exercises are structured.

Rules: Action Execution Blocks



Rules: Action Execution Blocks

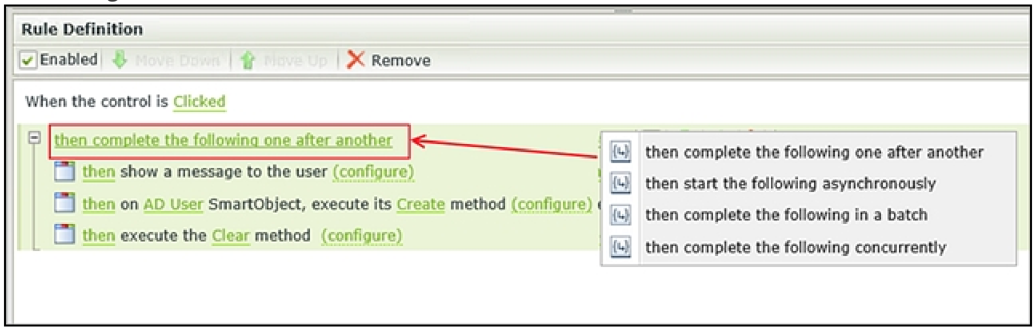
- Define the execution behavior of multiple Actions
- **complete the following one after another ("then")**
  - Sequential, separate execution
  - Actions are performed separately, in sequence, one at a time
  - The Rule waits for each Action to complete before performing the next Action
- **start the following asynchronously ("execute and continue")**
  - Fire-and-Forget execution
  - Start the actions, don't wait for completion, continue to the next block
  - Best for independent loading of controls without blocking UI
- **complete the following in a batch ("also")**
  - "Single-Job, Single Connection" execution
  - Actions are grouped and then executed in the sequence they are defined
  - Fastest option when Actions need to transfer values between each other
- **complete the following concurrently ("and")**
  - Parallel, independent execution
  - Actions are performed all at the same time, wait for Actions to complete before moving on to next block
  - Cannot use this approach to pass return values from one Action into another Action
  - Fast execution time since operations are simultaneous

Video

It is possible to perform many operations or Actions in a Rule. You can use Action Execution Blocks to tell the Rule how to execute the sequence of operations. This is used mostly for performance reasons to make Forms load in batch or asynchronously or for relationship purposes (e.g. master/detail Forms where the header must be created first and then all the line items).

Execution Blocks tell K2 whether to execute the methods in sequence (one after another), asynchronously (execute and continue), all at once (concurrently) or wrap the methods into a batch statement (sequentially, but in a batch).

Selecting an Execution Block mode for a set of Rule Actions



Rule Definition

Enabled | Move Down | Move Up | Remove

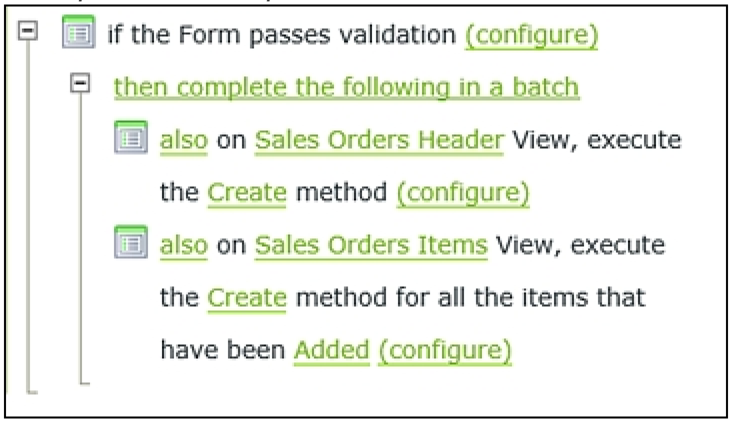
When the control is Clicked

then complete the following one after another
- then show a message to the user (configure)
- then on AD User SmartObject, execute its Create method (configure)
- then execute the Clear method (configure)

then complete the following one after another
then start the following asynchronously
then complete the following in a batch
then complete the following concurrently

Consider the Sales Orders Form scenario we have been building in this module. We want to add the two Views (Order Header and Item Details) onto one Form so that a user can capture both the header and details information in one Form. When the user clicks the "Submit Order" button, we want to execute a Rule that will create the Sales Orders Header and then create each of the items in the Sales Orders Details list as a single Action. Normally, you will also want these actions wrapped into a batch block so that the header record is created and then each line item is created, with all of these operations performed as a single job.To achieve this, you would define a “batch” Execution Block that will create the header and details in one job. See the example below:

Example of a batch operation



if the Form passes validation (configure)
then complete the following in a batch
also on Sales Orders Header View, execute the Create method (configure)
also on Sales Orders Items View, execute the Create method for all the items that have been Added (configure)

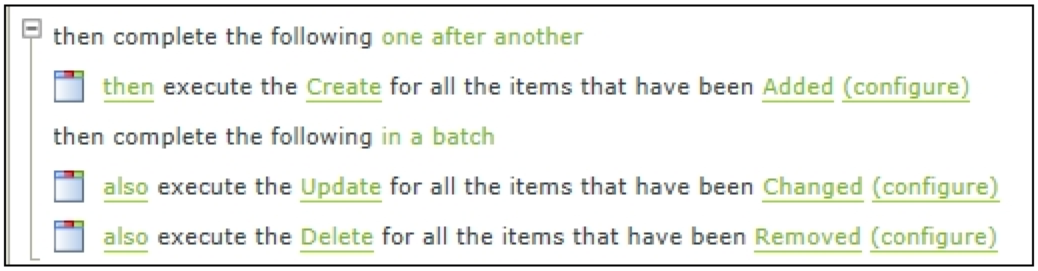
The following execution modes are available:

Execution Mode	Description
complete the following one after another (“then”)	This is a sequential, separate execution of each Action and the actions are performed separately, in sequence, one at a time. The Rule waits for each Action to complete before performing the next Action. This is usually the slowest execution mode.
start the following asynchronously (“execute and continue”)	This is a Fire-and-Forget style of execution where the actions are started and the Rule will not wait for completion but instead immediately continues to the next block. This style is best for independent loading of Controls without blocking the Form with "loading" spinners.
complete the following in a batch (“also”)	This is a “Single-Job, Single Connection” execution. The Actions are grouped and then executed in the sequence they are defined, using a single connection to the K2 Server. This is usually the fastest option when Actions need to transfer values between each other.
complete the following concurrently (“and”)	This is a parallel, independent execution mode. Actions are performed all at the same time, and wait for Actions to complete before moving on to the next block. You cannot use this approach to pass return values from one Action into another Action because the Actions are executed completely separately and it is not possible to tell which Action will complete first. This option usually results in a fast execution time since all operations are performed simultaneously.

Note
"In a batch" is not equivalent to true transactional awareness. If a subsequent method call fails, K2 will not automatically roll back the previous statements. "Concurrently" is not the same as true parallel, multi-threaded processing.

If needed, you can also combine different Execution Blocks into the same Action statement. This is useful if you have a dependency to create the first item but then each additional item can be executed in a batch. Here is an example of two Execution Blocks in the same Action:

Two Execution Blocks in the same action



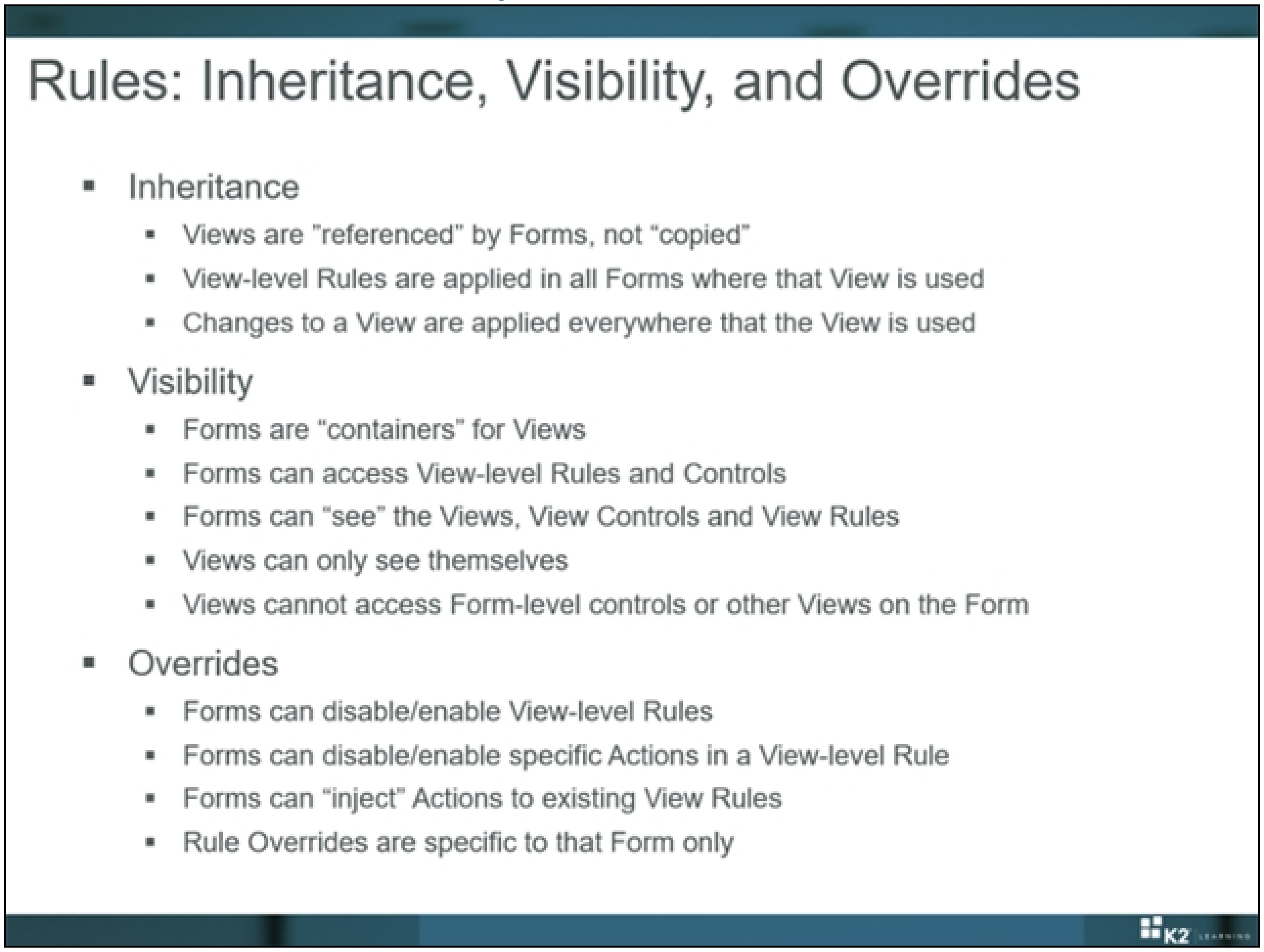
then complete the following one after another
then execute the Create for all the items that have been Added (configure)
then complete the following in a batch
also execute the Update for all the items that have been Changed (configure)
also execute the Delete for all the items that have been Removed (configure)

Tip
Executing a set of Actions Concurrently (all at once) or Asynchronously (execute and continue) in an Execution Block can improve the perceived performance of your Forms and Views, since all operations are performed simultaneously on the K2 Server. If you have a Form with multiple Views that are independent of each other, you can execute the Initialize method of each View Concurrently/Asynchronously to improve the user experience. However, bear in mind the multiple connections to the K2 Server and the impact this may have in an environment with many users.

Summary

- Execution Blocks define how Actions are grouped together and executed
- complete the following one after another (“then”)
  - Sequential, separate execution, actions are performed separately, in sequence, one at a time
- start the following asynchronously (“execute and continue”)
  - Fire-and-Forget execution: start the Actions, don’t wait for completion, continue to the next block
  - Best for independent loading of Controls without blocking the User Interface
- complete the following in a batch (“also”)
  - “Single-Job, Single Connection” execution: Actions are grouped and then executed in the sequence they are defined
  - Fastest option when Actions need to transfer values between each other
- complete the following concurrently (“and”)
  - Parallel, independent execution: Actions are performed all at the same time, wait for Actions to complete before moving on to the next block
  - Fast execution time since operations are simultaneous

Rules: Inheritance, Visibility and Overrides



Rules: Inheritance, Visibility, and Overrides

- Inheritance
  - Views are "referenced" by Forms, not "copied"
  - View-level Rules are applied in all Forms where that View is used
  - Changes to a View are applied everywhere that the View is used
- Visibility
  - Forms are "containers" for Views
  - Forms can access View-level Rules and Controls
  - Forms can "see" the Views, View Controls and View Rules
  - Views can only see themselves
  - Views cannot access Form-level controls or other Views on the Form
- Overrides
  - Forms can disable/enable View-level Rules
  - Forms can disable/enable specific Actions in a View-level Rule
  - Forms can "inject" Actions to existing View Rules
  - Rule Overrides are specific to that Form only

Video

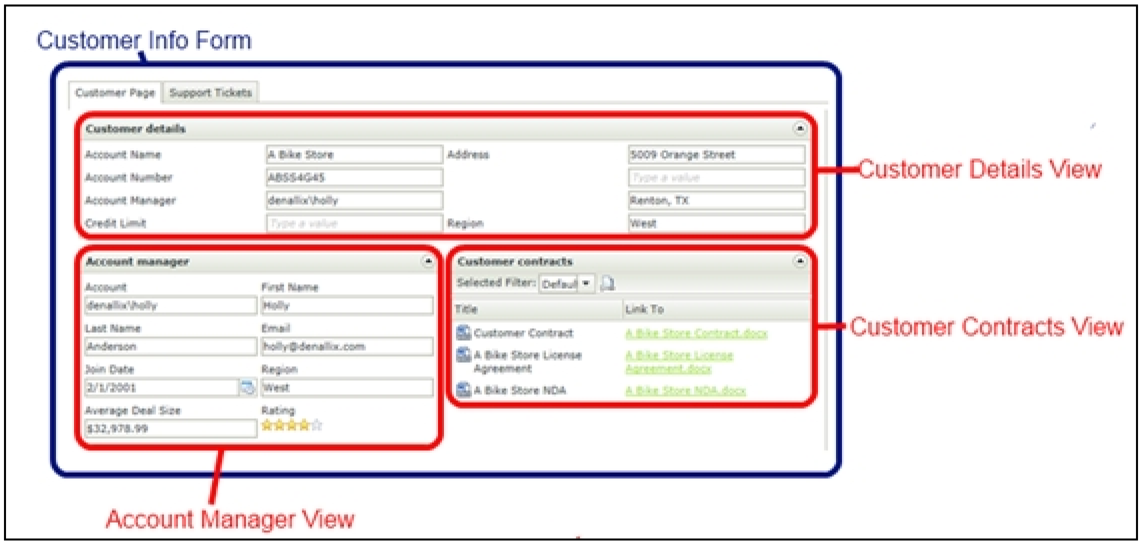
From a Rules perspective, it is important to understand how Rules can override and extend other Rules, and what Rules have access to depending on where they are defined.

Inheritance

Rule Inheritance (in other words, how do Forms inherit the Rules from the Views on that Form) is actually quite simple if you remember the following statement: Views are re-used or "referenced" by Forms, not "copied". This means that when you make changes to a View, those changes are automatically applied everywhere the View is used, because the same View is re-used by multiple Forms. It also means that whatever Rules you define for a View will automatically be applied everywhere that the View is used, unless a Form specifically overrides a View Rule or Action.

Visibility

Remember that Forms are "containers" for one or more Views. Consider the sample Form below: here we have three separate Views (marked in red) on one Form (marked in blue). The key is that the Views are contained in the Form, so the Form is effectively the “parent” or “container” for the three Views.



Customer details

Account Name	A Bike Store	Address	5009 Orange Street
Account Number	ABS54G45		Type a value
Account Manager	denallix\holly		Renton, TX
Credit Limit	Type a value	Region	West

Account manager

Account	First Name
denallix\holly	Holly
Last Name	Email
Anderson	holly@denallix.com
Join Date	Region
2/1/2001	West
Average Deal Size	Rating
$32,978.99	★★★★☆

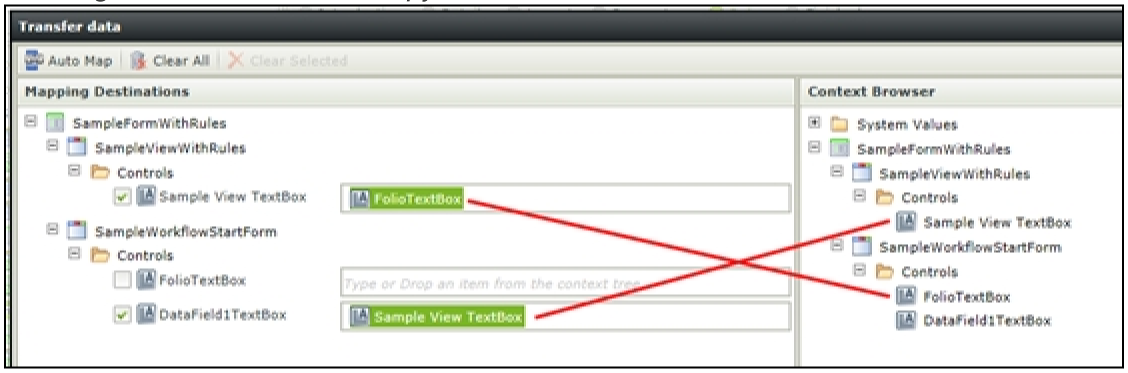
Customer contracts

Selected Filter: Default

Title	Link To
Customer Contract	A Bike Store Contract.docx
A Bike Store License Agreement	A Bike Store License Agreement.docx
A Bike Store NDA	A Bike Store NDA.docx

Because the Form contains the Views, the Form can access the Controls and Rules of each View. Therefore, you could define a Rule on the Form that executes the "Load" method for each View whenever the Form is opened. The Views, however, cannot see anything "outside" of themselves: the Views cannot execute Rules on the Form level, nor can a View "see" another View's Controls or Rules. For example, the Customer Details View doesn't know anything about the Account Manager View. If you had to transfer values between those Views you would have to define a Rule on the Form level.

Defining a Form-level Rule to copy data from one View to another View on the same Form



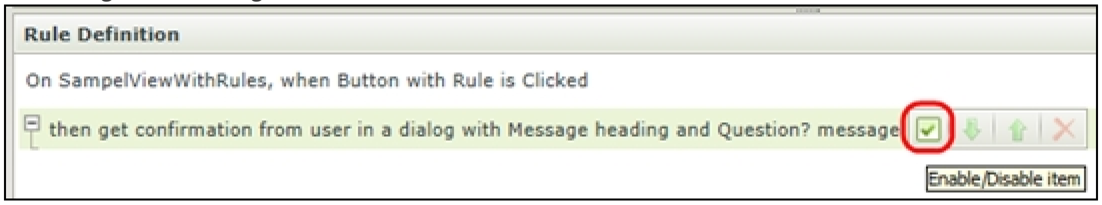
Overrides

By default, View-level Rules will always execute when the associated Event happens on a View. For example, let's say you have a button on a View. This button has a Rule that populates the View when the button is clicked. When you add that View to a Form, the button with its associated Rule is still operational and you do not need to do anything to enable the Rule: because of Inheritance, the Rule continues to work.

Now suppose you want to disable that button Rule when the View is used on a specific Form. Because the parent Form can see the View's Rules, you can use the Form Rules editor to disable the button click Event for the View.You can even "inject" Actions into the Rules executed in Views; for example, if you want your Form to execute an Action halfway between the existing Actions of a View-level Rule.

When a View Rule is disabled by a Form, it only affects the View Rule for that particular Form: the View Rule will still work as it was defined in the View for all other Forms.

Enabling or disabling a View Rule Action from a Form



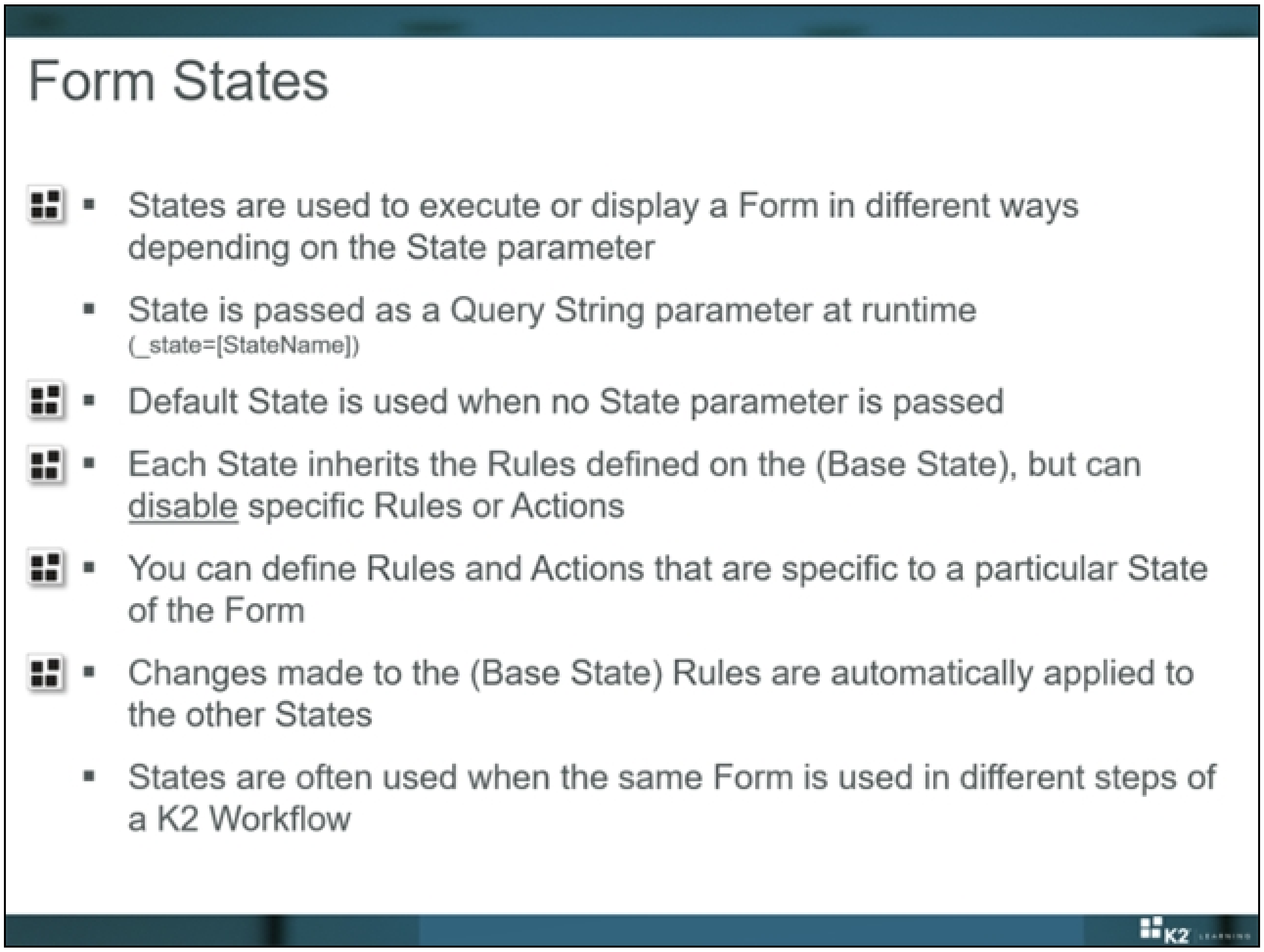
On SampelViewWithRules, when Button with Rule is Clicked

then get confirmation from user in a dialog with Message heading and Question? message

Summary

- Inheritance
  - Views are ”referenced” by Forms, not “copied”. In other words, the exact same View is used, a Form does not create a “clone” or “copy” of a View.
  - View-level Rules are applied in all Forms where that View is used, changes to a View are applied everywhere that the View is used.
- Visibility
  - Forms can access View-level Rules and Controls
  - Forms can “see” the Views, View Controls and View Rules
  - Views can only see themselves: Views cannot access Form-level controls or other Views on the Form
- Overrides (i.e. how can Forms override the Rules for Views on that Form)
  - Forms can disable/enable View-level Rules
  - Forms can disable/enable specific Actions in a View-level Rule
  - Forms can “inject” Actions to existing View Rules
  - Rule Overrides are specific to that Form only

Form States



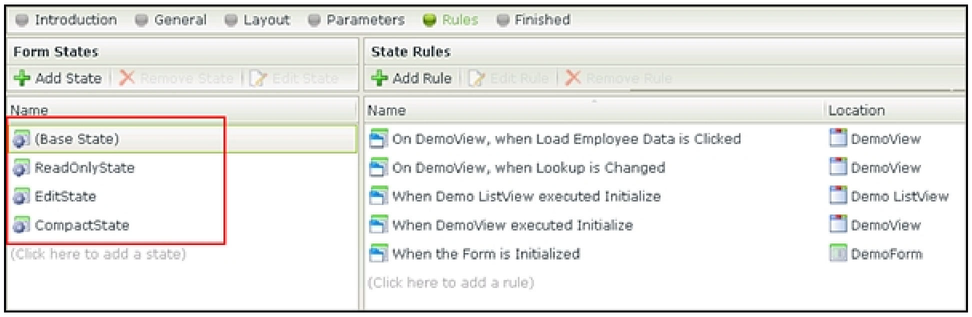
Form States

- States are used to execute or display a Form in different ways depending on the State parameter
  - State is passed as a Query String parameter at runtime (_state=[StateName])
- Default State is used when no State parameter is passed
- Each State inherits the Rules defined on the (Base State), but can disable specific Rules or Actions
- You can define Rules and Actions that are specific to a particular State of the Form
- Changes made to the (Base State) Rules are automatically applied to the other States
  - States are often used when the same Form is used in different steps of a K2 Workflow

Video

We briefly discussed States in the *100.YYZ SmartForms Fundamentals* learning module. You should remember that States are just a way to run the same Form in different ways depending on the State parameter. States are most often used when a Form is associated with a Workflow. Here is an example: suppose you are using the same Form to Start a Workflow and for the User Tasks in the Workflow. When you associate the Form with the Workflow, you would normally define at least two States for the Form: the First State ("Submission") is used when the Form will Start a Workflow and the second State ("Approval") is used when the Form is used to complete a user task in the Workflow. That said, you can also use States for other purposes, like defining a "Read-Only" state versus an "Edit State" for the same Form. The different States are required because the Form Rules must behave differently depending on the State of the Form.

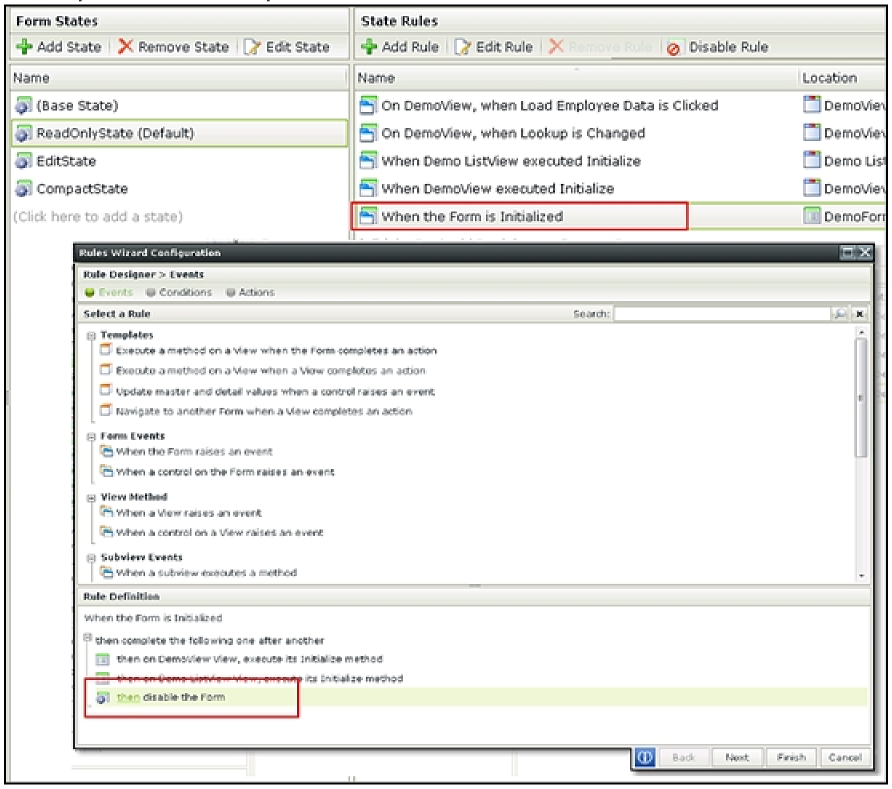
Different States defined for the same Form



Name
(Base State)
ReadOnlyState
EditState
CompactState
(Click here to add a state)

Name	Location
On DemoView, when Load Employee Data is Clicked	DemoView
On DemoView, when Lookup is Changed	DemoView
When Demo ListView executed Initialize	Demo ListView
When DemoView executed Initialize	DemoView
When the Form is Initialized	DemoForm
(Click here to add a rule)	

Example of a State-specific Action that disables the Form if the State is ReadOnlyState



Form States
(Base State)
ReadOnlyState (Default)
EditState
CompactState
(Click here to add a state)

Name	Location
On DemoView, when Load Employee Data is Clicked	DemoViev
On DemoView, when Lookup is Changed	DemoViev
When Demo ListView executed Initialize	Demo List
When DemoView executed Initialize	DemoViev
When the Form is Initialized	DemoForr

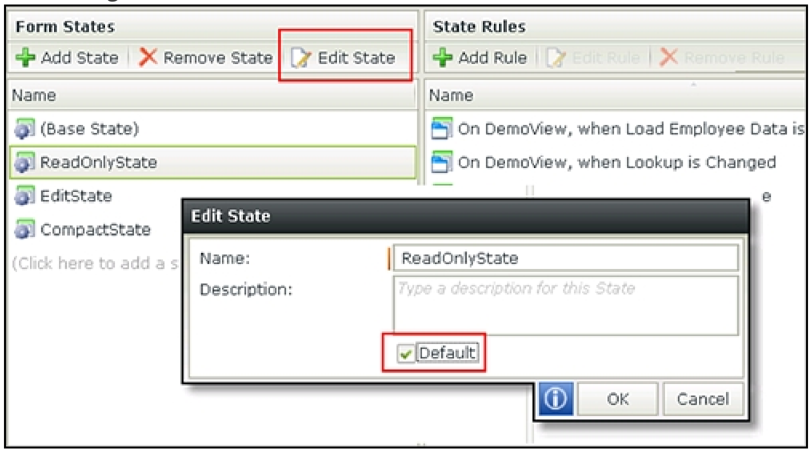
Rule Definition

When the Form is Initialized
then complete the following one after another
then on DemoView View, execute its Initialize method
~~then on Demo ListView View, execute~~ its Initialize method
then disable the Form

At runtime, the Form State is passed as a Query String Parameter: **_state** (e.g. http://SmartFormsServer/Runtime/Project/FormName?**_state=Approval**).

When multiple States are defined for a Form, it will automatically check the value of the State query string parameter and run the associated State Rules. If the State parameter is not found, the Form will run in the State that was set to be the Default State.

Selecting the Default State of a Form



Edit State

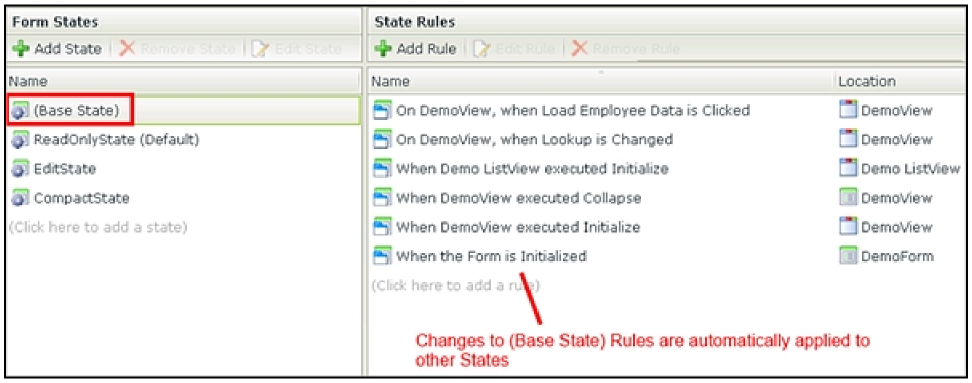
Name:	ReadOnlyState
Description:	Type a description for this State

[✓ Default]

States and Rule Inheritance

New States defined for a Form will always “inherit” the Rules defined for the (Base State) of a Form. You can edit the Rules for each State to disable the Base State’s Rules, add additional actions or even perform other operations between the actions defined in the Base State. Furthermore, any changes made to the (Base State) Rules are automatically applied to the other States of the Form, which allows you to define the majority of the Form Rules in a common, shared State and then only add more Rules or Actions for the specific States that require different behavior.

Base State Rules


Form States
Add State | Remove State | Edit State
State Rules
Add Rule | Edit Rule | Remove Rule
(Click here to add a state)
(Click here to add a rule)
Changes to (Base State) Rules are automatically applied to other States

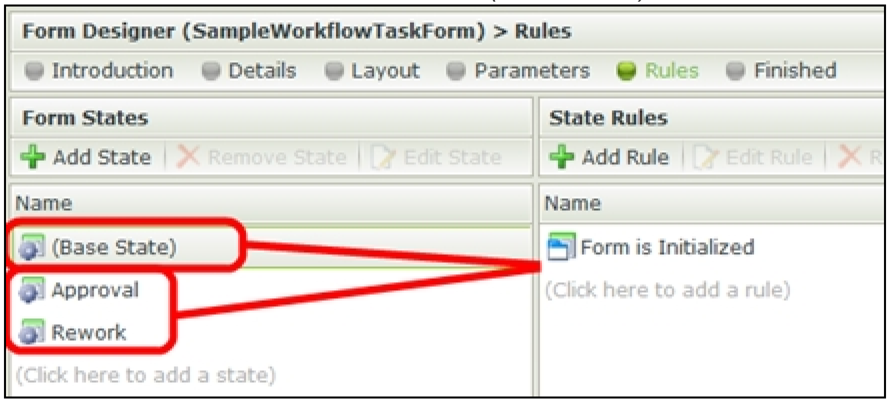
Name
(Base State)
ReadOnlyState (Default)
EditState
CompactState

Name	Location
On DemoView, when Load Employee Data is Clicked	DemoView
On DemoView, when Lookup is Changed	DemoView
When Demo ListView executed Initialize	Demo ListView
When DemoView executed Collapse	DemoView
When DemoView executed Initialize	DemoView
When the Form is Initialized	DemoForm

Think of this inheritance approach as an “extension” of the Base State’s Rules rather than a “copy” of the Base State's Rules. If you make a change to the Base State’s Rules, all other States will automatically include these changes. This makes maintaining a Form easy: put all the generic, Form-level Rules into the Base State, and then only add overrides to each State of the Form where necessary.

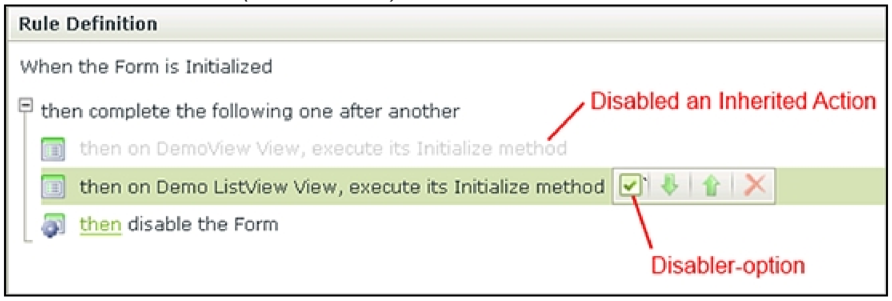
Consider the example below. Here, we have defined two States for a Form (Approval and Rework). Each of these States will automatically “inherit” the Rules defined for the (Base State) of the Form. If needed, you can edit a particular State to override how the Rules in the Base State are executed, or add additional Rules that are only relevant for that State of the Form. If you were to add more Rules to the (Base State), the Approval and Rework States would automatically “inherit” the changes made on the (Base State).

States inherit the Rules defined for the (Base State)



Subsequent States of the Form can selectively disable Actions defined for the Base State. You can configure this by just checking the option to disable a specific Rule Action defined in the Base State, as shown below.

States can disable (Base State) Rules and Actions



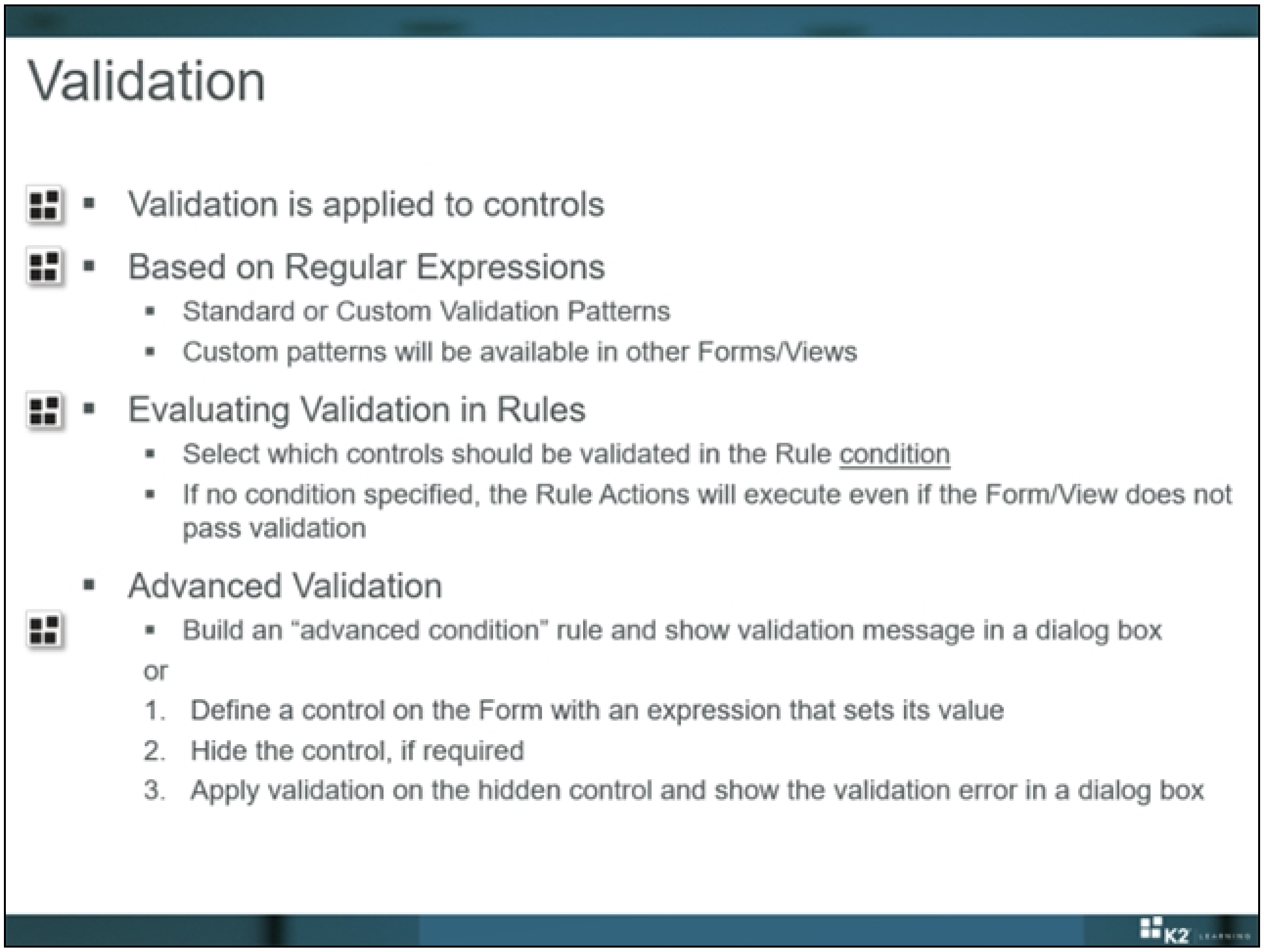
When the Form is Initialized
then complete the following one after another
then on DemoView View, execute its Initialize method
then on Demo ListView View, execute its Initialize method
then disable the Form

Summary

- States are used to execute or display a Form in different ways depending on the State parameter
- The State is passed as a query string parameter at runtime (*_State=[StateName]*)
- The “Default” State is used when no State parameter is passed
- Each State inherits the Rules defined on the (Base State), but can disable specific Rules or Actions from the Base State

- You can define Rules and Actions that are specific to a particular State of the Form
- Changes made to the (Base State) Rules are automatically applied to the other States

Validation



Validation

- Validation is applied to controls
- Based on Regular Expressions
  - Standard or Custom Validation Patterns
  - Custom patterns will be available in other Forms/Views
- Evaluating Validation in Rules
  - Select which controls should be validated in the Rule condition
  - If no condition specified, the Rule Actions will execute even if the Form/View does not pass validation
- Advanced Validation
  - Build an "advanced condition" rule and show validation message in a dialog box

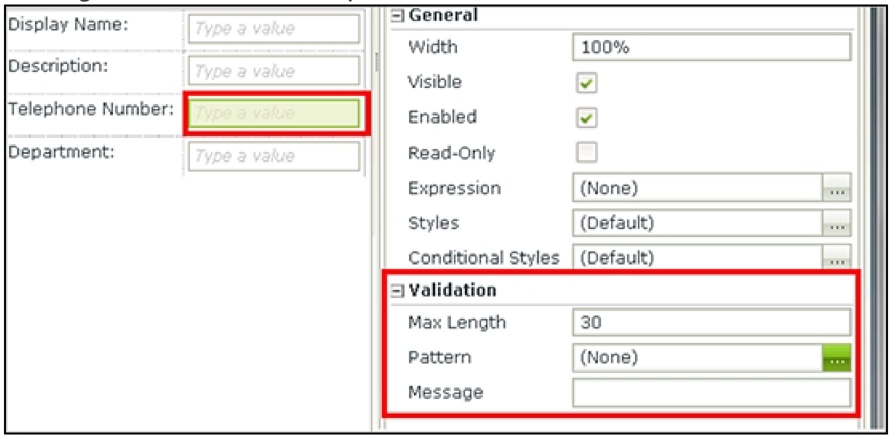
  or

  1. Define a control on the Form with an expression that sets its value
  2. Hide the control, if required
  3. Apply validation on the hidden control and show the validation error in a dialog box

Video

You should recall that validation is applied to Controls. Using the Properties Pane, you can use the validation options to select a standard or custom expression for a particular Control, and you can add values to limit the length of the input or show a custom message to the user.

Setting the validation for a specific control



Display Name:	Type a value
Description:	Type a value
Telephone Number:	Type a value
Department:	Type a value

General

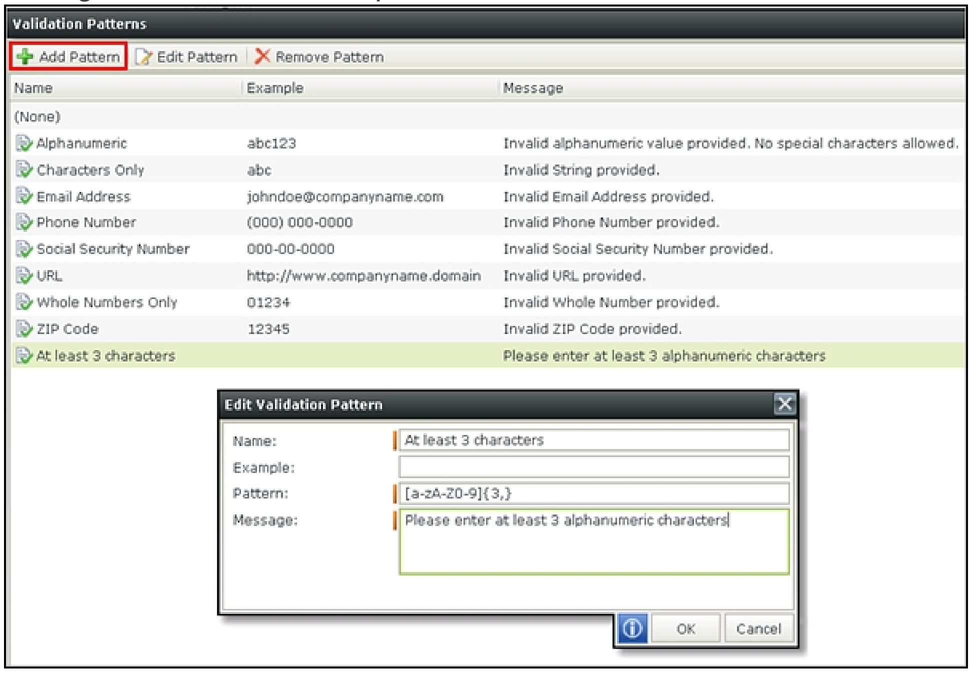
Width	100%
Visible	✓
Enabled	✓
Read-Only	
Expression	(None)
Styles	(Default)
Conditional Styles	(Default)

Validation

Max Length	30
Pattern	(None)
Message	

You can add your own validation expressions as well, and if you have done so, those custom validation expressions will be available to other Forms and Views in your K2 environment.

Adding a custom validation expression



Name	Example	Message
(None)		
Alphanumeric	abc123	Invalid alphanumeric value provided. No special characters allowed.
Characters Only	abc	Invalid String provided.
Email Address	johndoe@companyname.com	Invalid Email Address provided.
Phone Number	(000) 000-0000	Invalid Phone Number provided.
Social Security Number	000-00-0000	Invalid Social Security Number provided.
URL	http://www.companyname.domain	Invalid URL provided.
Whole Numbers Only	01234	Invalid Whole Number provided.
ZIP Code	12345	Invalid ZIP Code provided.
At least 3 characters		Please enter at least 3 alphanumeric characters

Edit Validation Pattern

Name:	At least 3 characters
Example:	
Pattern:	[a-zA-Z0-9]{3,}
Message:	Please enter at least 3 alphanumeric characters

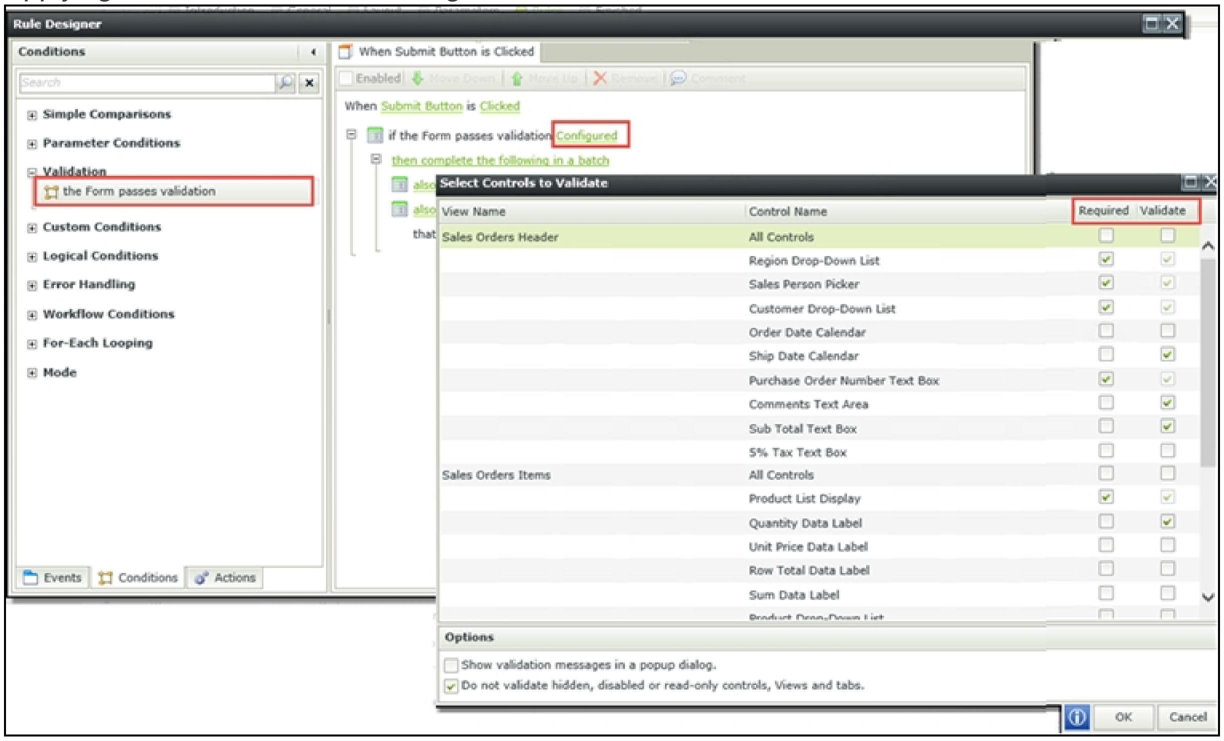
Tip
Regular Expressions can get complex to build. Two of the more useful resources to help you define custom Regular Expressions are http://regexlib.com and http://renschler.net/RegexBuilder/.

Evaluating Validation in Rules

An important point is that validation errors in a Form or View will not prevent Rule Actions from being executed on the Form or View. Let's take the Sales Orders Form again as an example: if the user entered an invalid value for the discount (more than the total cost of the line items, for example), that error will not prevent the user from saving the new sales order.

To avoid this situation, you can use a Rule condition that will check if the Form passes validation before continuing. The screenshot below shows how this Rule condition is set up. Notice that you can **configure** the condition. In the configuration screen, you can select which Controls should be **Validated** or indicate which Controls are **Required**. (Just because a Control must be validated does not mean it is always required, which is why you have two options to select). You can also decide whether to show the validation errors as a dialog box, and whether to validate hidden/read-only components on the Form.

Applying Validation before executing Rule Actions



Rule Designer

Conditions

- Simple Comparisons
- Parameter Conditions
- Validation
  - the Form passes validation
- Custom Conditions
- Logical Conditions
- Error Handling
- Workflow Conditions
- For-Each Looping
- Mode

Events | Conditions | Actions

When Submit Button is Clicked

Enabled | Move Down | Move Up | Remove | Comment

When Submit Button is Clicked

if the Form passes validation Configured

then complete the following in a batch

Select Controls to Validate

View Name	Control Name	Required	Validate
Sales Orders Header	All Controls	☐	☐
	Region Drop-Down List	✓	✓
	Sales Person Picker	✓	✓
	Customer Drop-Down List	✓	✓
	Order Date Calendar	☐	☐
	Ship Date Calendar	☐	✓
	Purchase Order Number Text Box	✓	✓
	Comments Text Area	☐	✓
	Sub Total Text Box	☐	✓
	5% Tax Text Box	☐	☐
Sales Orders Items	All Controls	☐	☐
	Product List Display	✓	✓
	Quantity Data Label	☐	✓
	Unit Price Data Label	☐	☐
	Row Total Data Label	☐	☐
	Sum Data Label	☐	☐

Options

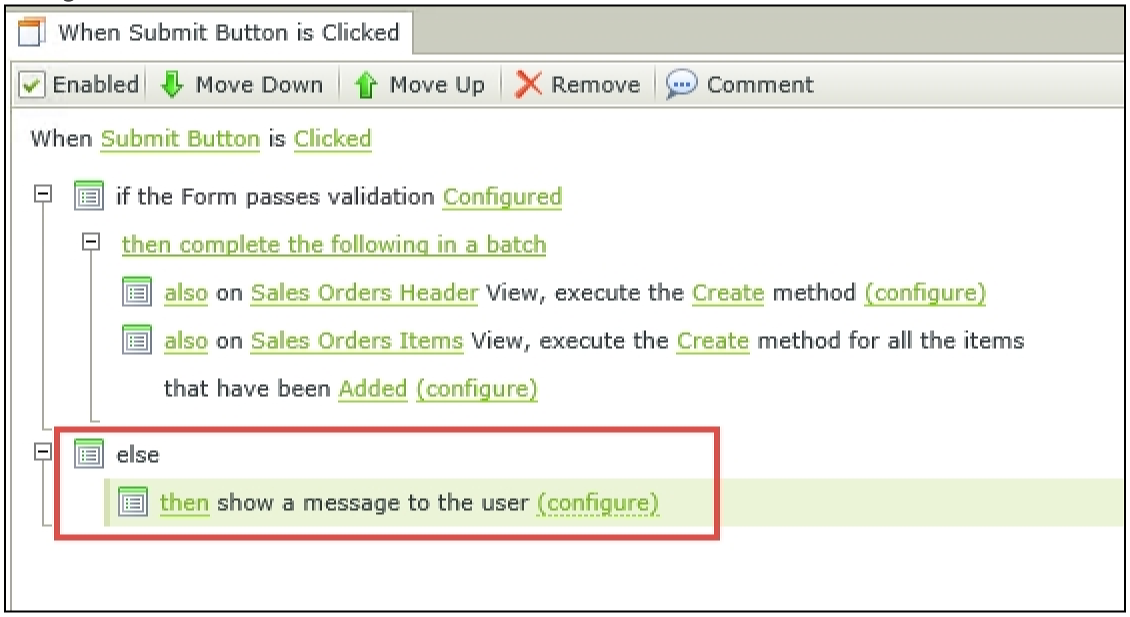
- ☐ Show validation messages in a popup dialog.
- ✓ Do not validate hidden, disabled or read-only controls, Views and tabs.

OK | Cancel

You should definitely leverage the validation condition to ensure that users cannot execute methods if the Form has not successfully passed validation. That said, you may not necessarily want to hold the user back or disable all Rule Actions if the form did not pass validation, so use whatever approach is most appropriate for your requirement.

If the validation passes, all of the Actions in the "then" block will be executed. If you wish, you can add an "else" condition to execute Rules if the validation did not pass.

Using an else condition in a validation check



When Submit Button is Clicked

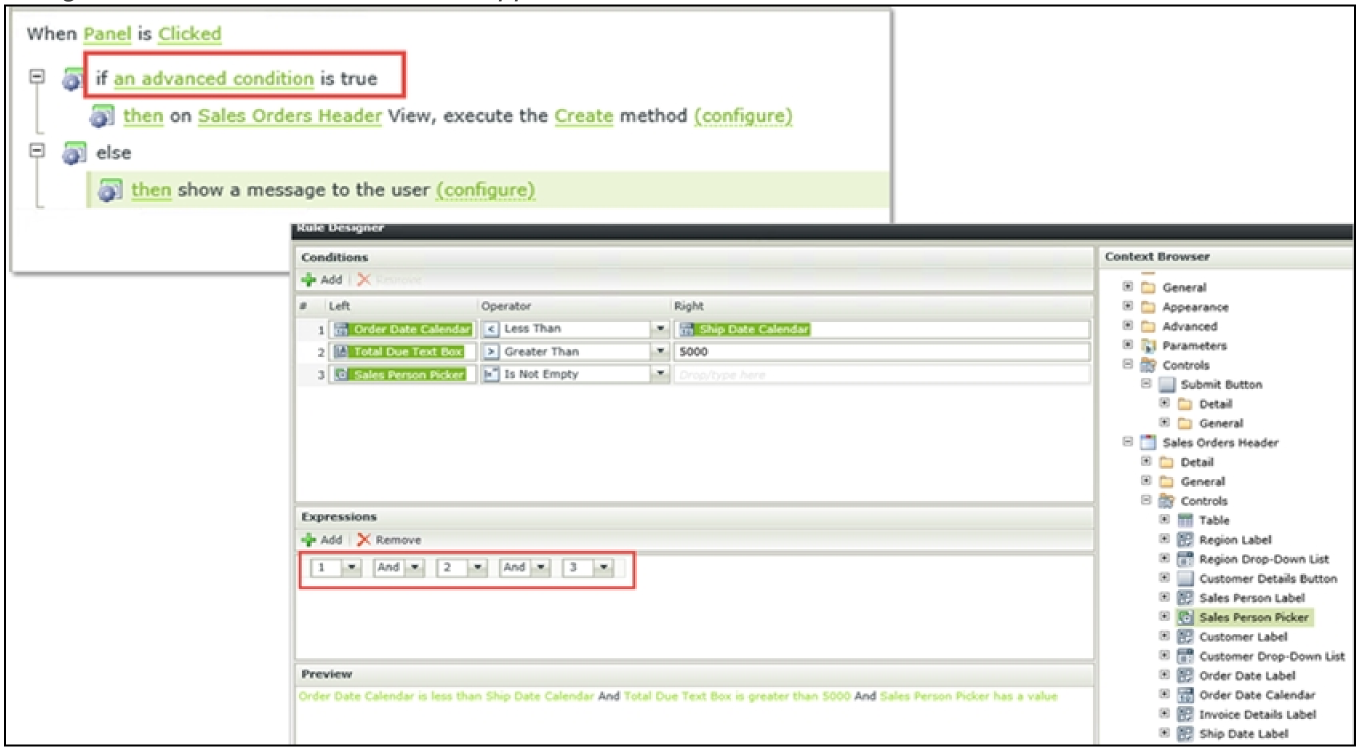
Enabled | Move Down | Move Up | Remove | Comment

```
When Submit Button is Clicked
  if the Form passes validation Configured
    then complete the following in a batch
      also on Sales Orders Header View, execute the Create method (configure)
      also on Sales Orders Items View, execute the Create method for all the items
      that have been Added (configure)
  else
    then show a message to the user (configure)
```

Advanced Validation

You can make use of the **an advanced condition is true** condition to perform more complex validation, such as validating that several controls have certain values. With this approach, you would normally show a message box to the user if the condition fails, or alternatively show a hidden Control on the Form that will explain the validation errors.

Using the "advanced condition is true" approach



When Panel is Clicked

if an advanced condition is true

then on Sales Orders Header View, execute the Create method (configure)

else

then show a message to the user (configure)

Rule Designer

Conditions

#	Left	Operator	Right
1	Order Date Calendar	Less Than	Ship Date Calendar
2	Total Due Text Box	Greater Than	5000
3	Sales Person Picker	Is Not Empty	Drop/type here

Expressions

1 And 2 And 3

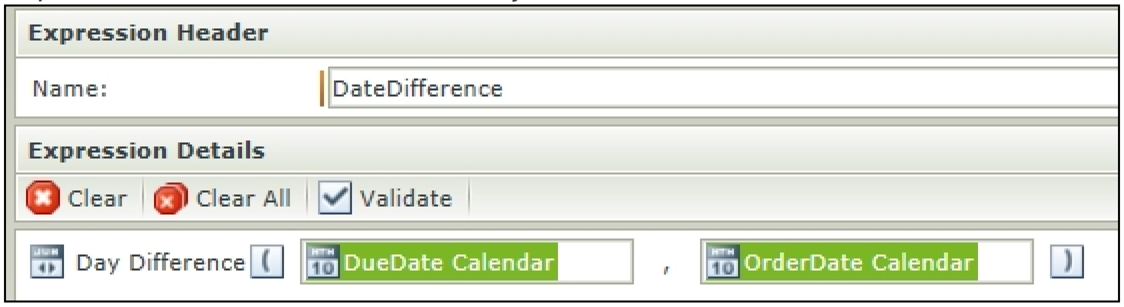
Preview

Order Date Calendar is less than Ship Date Calendar And Total Due Text Box is greater than 5000 And Sales Person Picker has a value

Regular Expressions are more focused toward text validation, so using them to validate something like dates and number values requires a slightly different approach. For an advanced validation scenario, you would usually define a Control on the View or Form and use an expression to set the Control's value. You would then define a validation Rule for the Control that compares the data from the expression to a Regular Expression pattern.

Here is an example: suppose you wanted to ensure that one date is earlier than another date on the same View. The easiest way is to define another Control on the View where the Control's value is set by an expression as the number of days between the two dates:

Expression to calculate the number of days between two dates



Expression Header

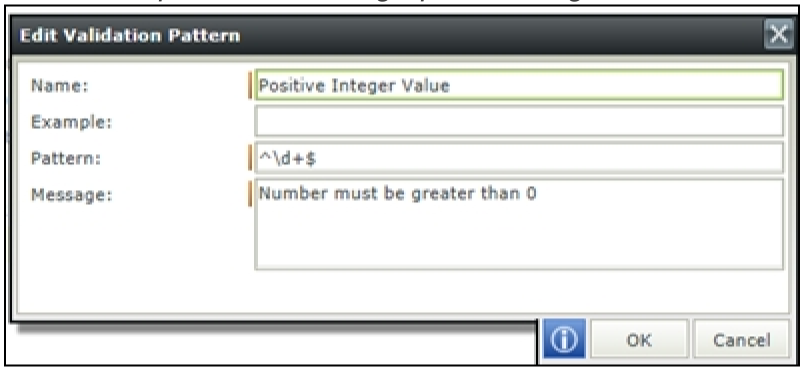
Name: DateDifference

Expression Details

Day Difference (DueDate Calendar , OrderDate Calendar)

On the View or Form, you can then define the validation for this calculated Control. In this case, we are using a custom expression to ensure that a positive integer in returned.

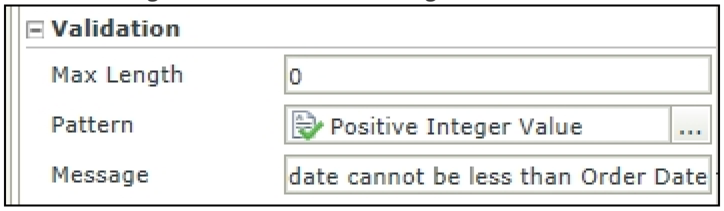
A custom expression validating a positive integer value



Name: Positive Integer Value
Example:
Pattern: ^\d+$
Message: Number must be greater than 0

Finally, you can configure the validation properties for the Control to display a message of your choice.

Customizing the validation message

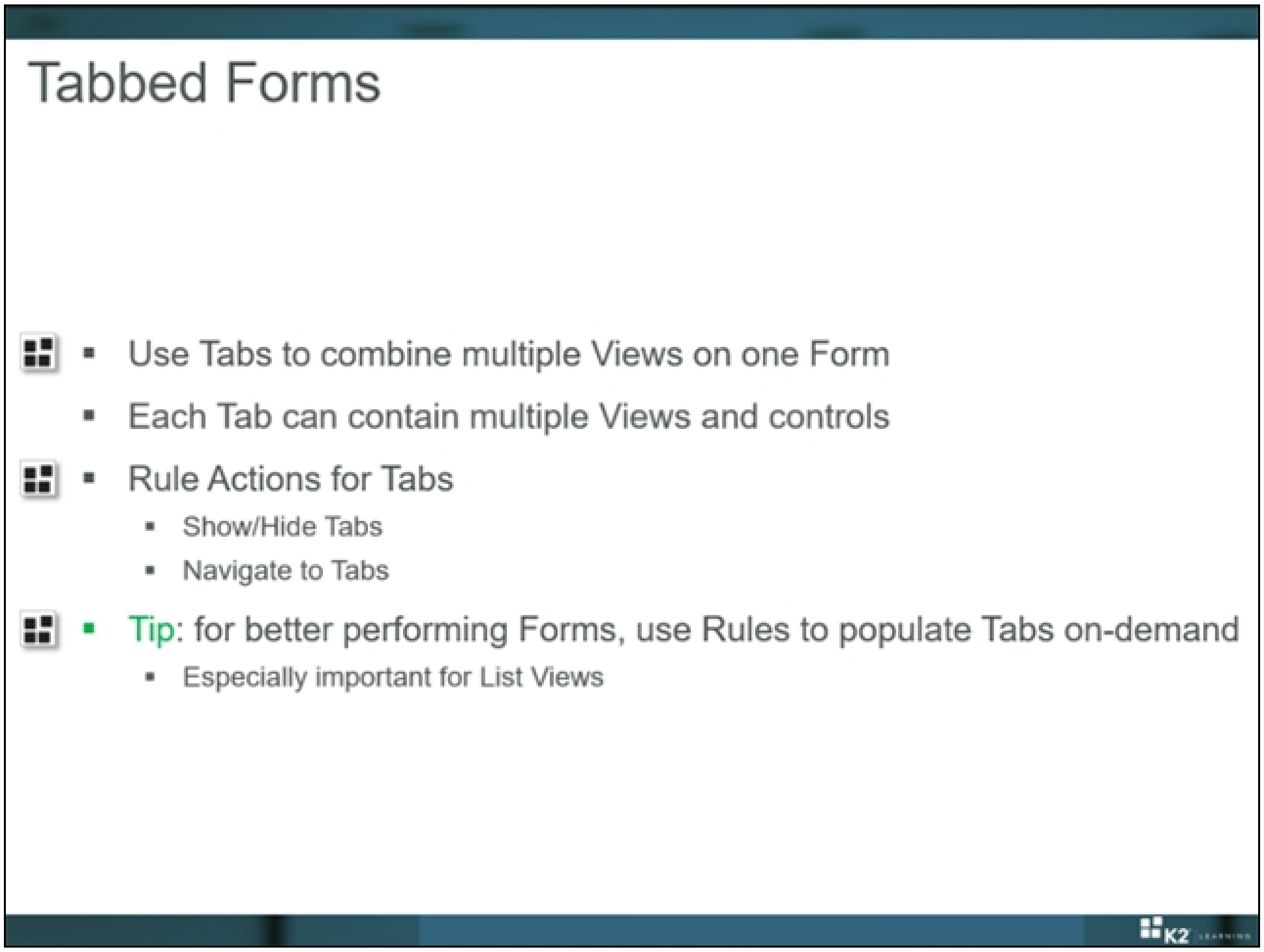


Max Length: 0
Pattern: Positive Integer Value
Message: date cannot be less than Order Date

Summary

- Validation is applied to Controls in SmartForms
- You define the validation as a pattern based on a Regular Expression
  - You can create (or copy in) custom Regular Expressions for custom validation
  - When you create custom validation expressions, those expressions are available to all other Forms in your environment.
- To apply a validation, define a Rule condition to validate a field when the Rule executes.
  - If the “passes validation” condition is not set, the Rule will still execute even if validation failed!
- To build more advanced validation, you can:
  - Use the “advanced condition is true” condition and show a message in a dialog box.
  - Alternatively, you may need to create a separate Control, set the Control’s value such that it can be validated, and then validate that Control.

Tabbed Forms



Tabbed Forms

- Use Tabs to combine multiple Views on one Form
- Each Tab can contain multiple Views and controls
- Rule Actions for Tabs
  - Show/Hide Tabs
  - Navigate to Tabs
- Tip: for better performing Forms, use Rules to populate Tabs on-demand
  - Especially important for List Views

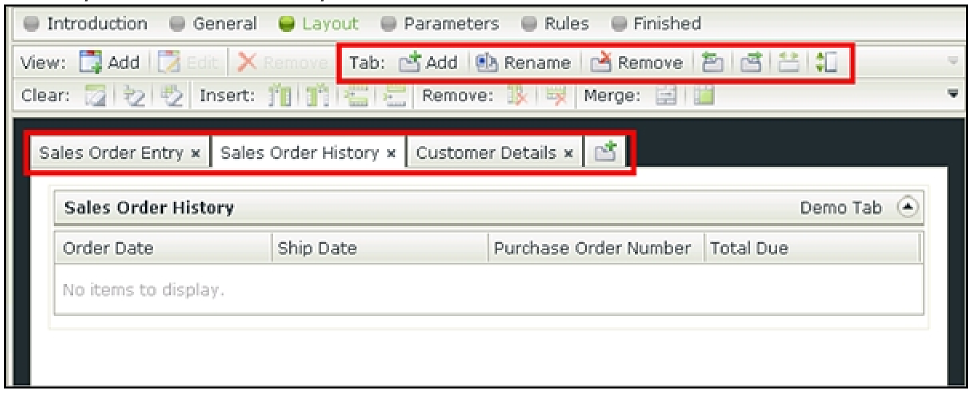
Video

Tabbed Forms are a nice way to present a lot of data or gathering a lot of input from a user without designing a very large and complex Form.

Note that Tabs can only be added to a Form, not to a View, and you can add as many Tabs as needed. However, you should take care not to add too many Tabs on a Form since very large Forms may not perform well at runtime.

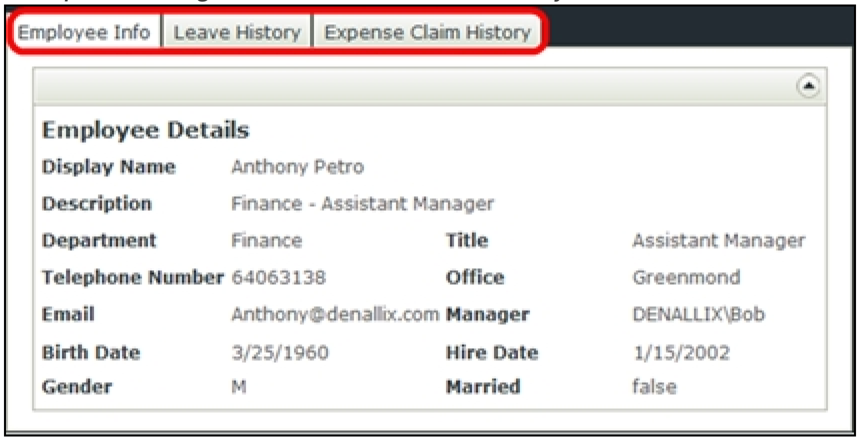
Normally, each Tab would display one or more Views. If you add a Control to a Tab, the Control actually exists on the Form level. In fact, Tabs are also Form-level Controls and expose methods that you can respond to in a Form-level Rule.

A sample Form with multiple Tabs



You can also use Tabs to implement a “more information”-style requirement, where only the basic information is shown in the initial Tab and the user can then open other Tabs to get more information. In the example below, we are showing only the basic employee information on the initial Tab, with other Tabs that display the Employee's Leave History and Expense Claim History.

Sample of a Page with "more information" style Tabs

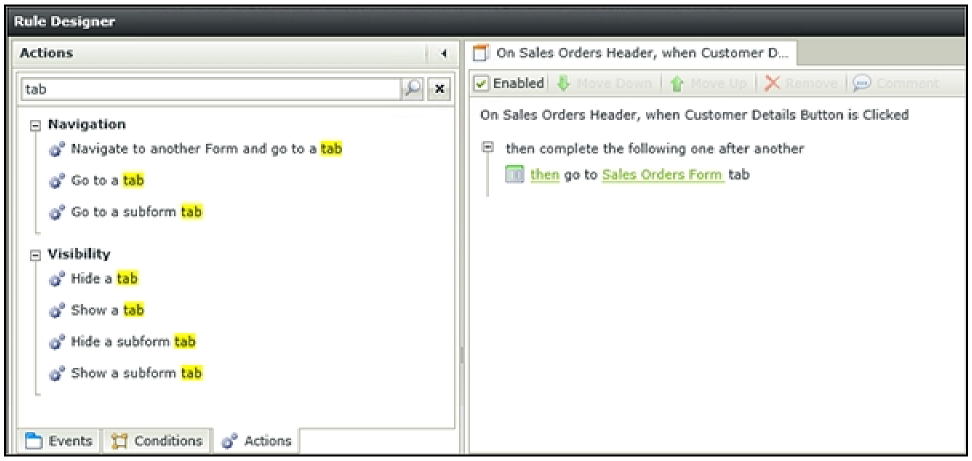


Employee Details

Display Name	Anthony Petro		
Description	Finance - Assistant Manager		
Department	Finance	Title	Assistant Manager
Telephone Number	64063138	Office	Greenmond
Email	Anthony@denallix.com	Manager	DENALLIX\Bob
Birth Date	3/25/1960	Hire Date	1/15/2002
Gender	M	Married	false

Form-level Actions can conditionally show or hide the Tabs in a Form, or you can navigate to Tabs. This is very useful in complex User Input scenarios, where (depending on some condition) the user may or may not need to complete additional Views, or you want to automatically go to a Tab in the Form based on some user input.

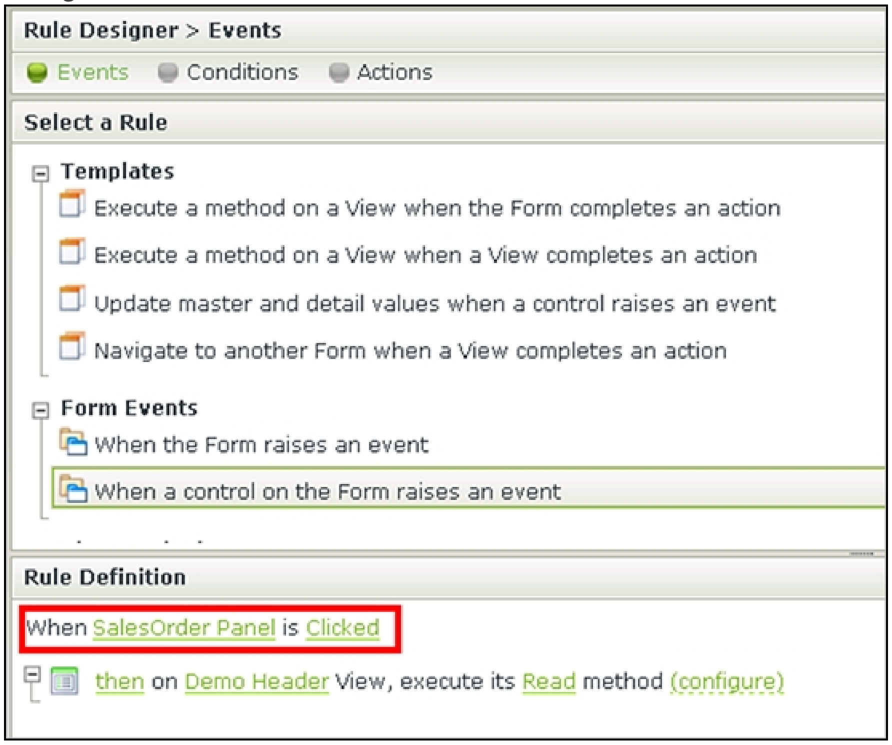
Some available Tab Actions for Rules



On Sales Orders Header, when Customer Details Button is Clicked
then complete the following one after another
then go to Sales Orders Form tab

Many Views on a Form can negatively impact performance, especially when you have List Views on a Form. When using multiple Tabs on a Form, you can improve the runtime performance of a Form by populating the Views inside a Tab on-demand, in other words, only populate those List Views when the Tab is clicked. You can do this by implementing a Rule that responds to the Clicked or Selected events for Tab Controls on the Form, and then only loading the data in the Views on that Tab based on those events. This way, you are not populating the data in the Tab until the user explicitly looks at that Tab. This approach is especially useful for large List Views that may take a few seconds to populate.

Using a Tab Event to load data on-demand


Rule Designer > Events
Events
Conditions
Actions
Select a Rule
Templates
Execute a method on a View when the Form completes an action
Execute a method on a View when a View completes an action
Update master and detail values when a control raises an event
Navigate to another Form when a View completes an action
Form Events
When the Form raises an event
When a control on the Form raises an event
Rule Definition

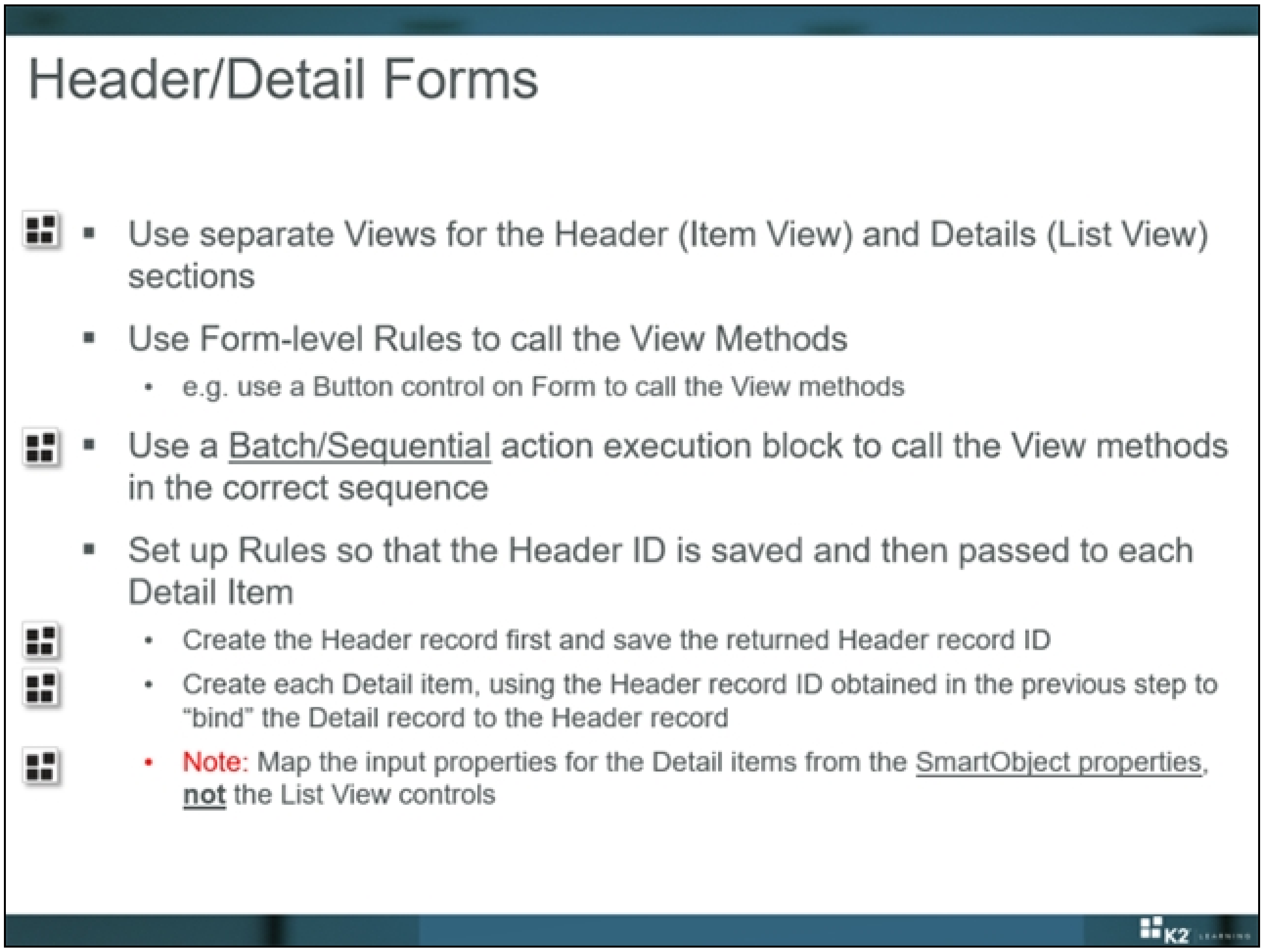
When SalesOrder Panel is Clicked

then on Demo Header View, execute its Read method (configure)

Summary

- You can use Tabs to show multiple Views on one Form
  - Tabs can only be used on Forms, they cannot be used on Views
- A Tab can in turn, contain multiple Views
- You can define Rule Actions to show or hide Tabs and navigate to Tabs
- Use Tab Events to populate List Views on-demand

Header/Detail Forms



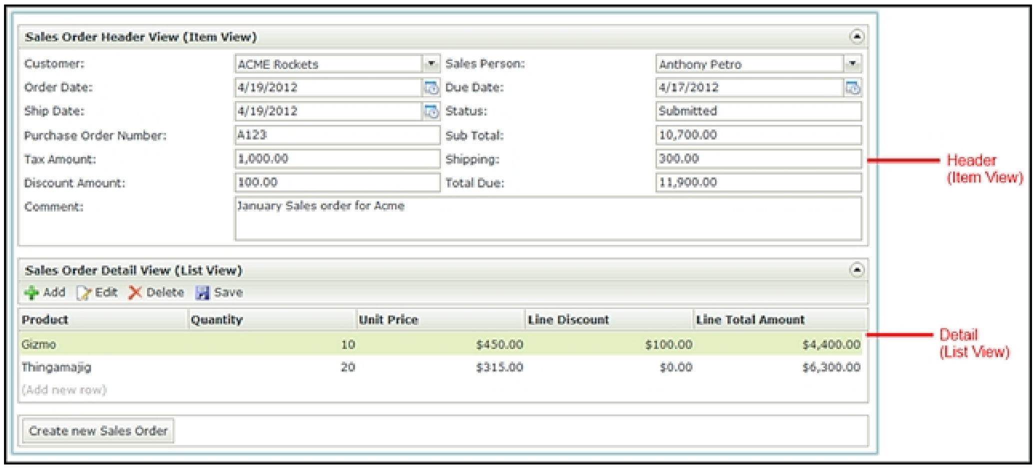
Header/Detail Forms

- Use separate Views for the Header (Item View) and Details (List View) sections
- Use Form-level Rules to call the View Methods
  - e.g. use a Button control on Form to call the View methods
- Use a Batch/Sequential action execution block to call the View methods in the correct sequence
- Set up Rules so that the Header ID is saved and then passed to each Detail Item
  - Create the Header record first and save the returned Header record ID
  - Create each Detail item, using the Header record ID obtained in the previous step to "bind" the Detail record to the Header record
  - Note: Map the input properties for the Detail items from the SmartObject properties, **not** the List View controls

Video

A common requirement is a Form where a user can enter data in a "header/detail" or "master/line items" style. Consider a Sales Order entry page: usually, a single sales order would have some header information like the customer details, order number, delivery address and so on. In the details section, the order would contain one or more products (line items) that make up the order. The individual line items, in turn, could be made up of several properties (such as the product number, quantity, unit price and so on).

Here is an example of a header/detail Form built with SmartForms.



Sales Order Header View (Item View)

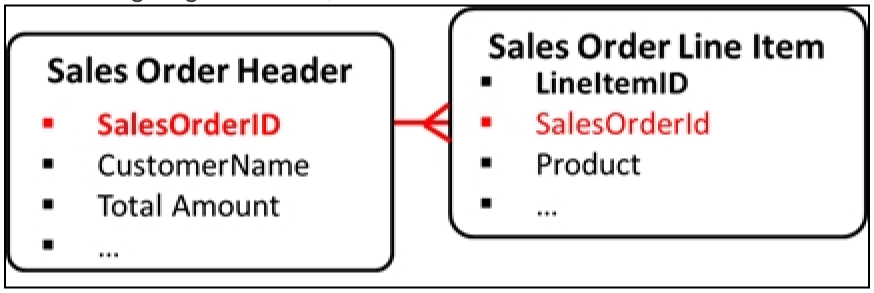
Customer:	ACME Rockets	Sales Person:	Anthony Petro
Order Date:	4/19/2012	Due Date:	4/17/2012
Ship Date:	4/19/2012	Status:	Submitted
Purchase Order Number:	A123	Sub Total:	10,700.00
Tax Amount:	1,000.00	Shipping:	300.00
Discount Amount:	100.00	Total Due:	11,900.00
Comment:	January Sales order for Acme		

Sales Order Detail View (List View)

Product	Quantity	Unit Price	Line Discount	Line Total Amount
Gizmo	10	$450.00	$100.00	$4,400.00
Thingamajig	20	$315.00	$0.00	$6,300.00
(Add new row)				

The first key to implementing Header/Detail Forms is to understand that the Form is made up of different Views. The "header" View is usually an Item View used to enter or display the header item data, while the "detail" View is usually a List View (read-only or editable) which displays the list of items associated with the header. These Views are then combined on a single Form to create the header/details layout. Rules behind the Form perform the necessary processing to load the header and its associated line items, or to call methods to create the line items added to the header form.

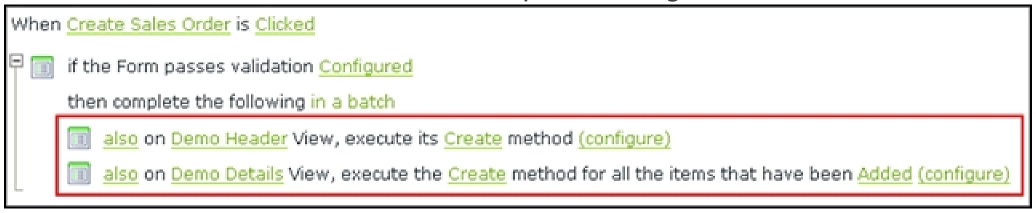
The essence of the master/detail is that there is a relationship between two SmartObjects, something like the sample below. Notice the common **SalesOrderId** property between the two SmartObjects. While this relationship does not have to be defined as an association between SmartObjects, defining the association will make your life a little easier when designing the Views, Rules and other User Interfaces that use the SmartObjects.



The second key to implementing header/detail Forms is to make sure that the associated properties are transferred correctly in Rules. As you saw in the relationship diagram above, each Sales Order Line Item has a SalesOrderId property, and this property is what binds each line item to the header item. When you configure Rules that will create the header and its line items, you would normally use a Form-level Control that calls the header view's Create method, saves the returned header record ID and then sets each line item's value to this returned ID before calling the same Create for each line item.

Here is an example of how this Rule would be configured for the sample Sales Order entry page. You will be creating a master/detail Form in the exercise, so you will see first-hand how the configuration of the Rules is done.

Call the Create methods in the correct sequence using a Batch Execution Block



When Create Sales Order is Clicked

if the Form passes validation Configured

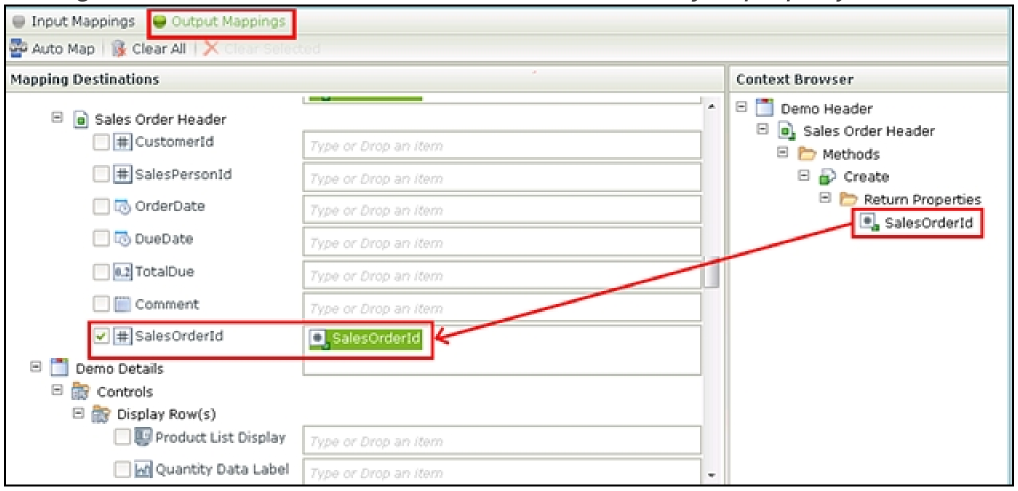
then complete the following in a batch

also on Demo Header View, execute its Create method (configure)

also on Demo Details View, execute the Create method for all the items that have been Added (configure)

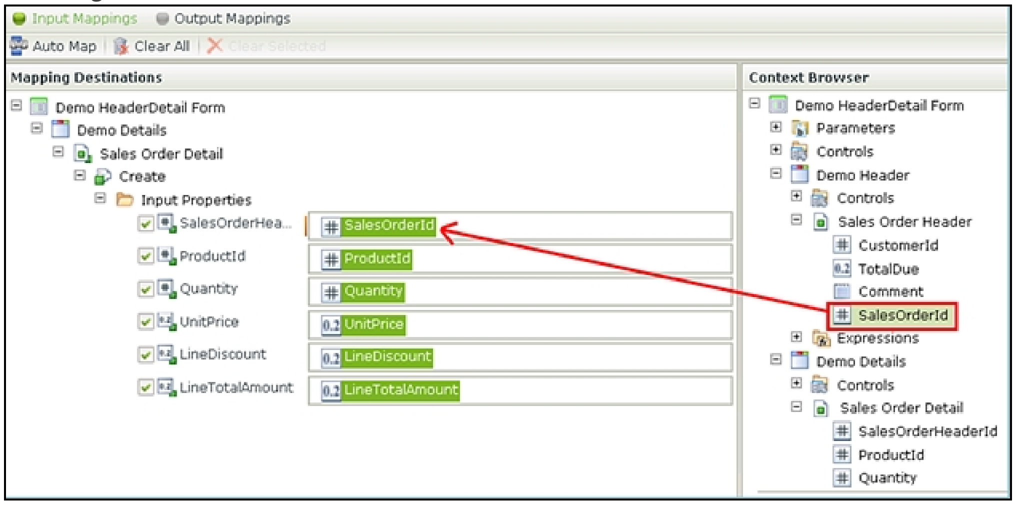
You have to save the ID of the header record somewhere after the header is created. Fortunately, you can just save the Header ID into the same property of the SmartObject in the Rule configuration as shown below.

Saving the returned Header record's ID into a SmartObject property



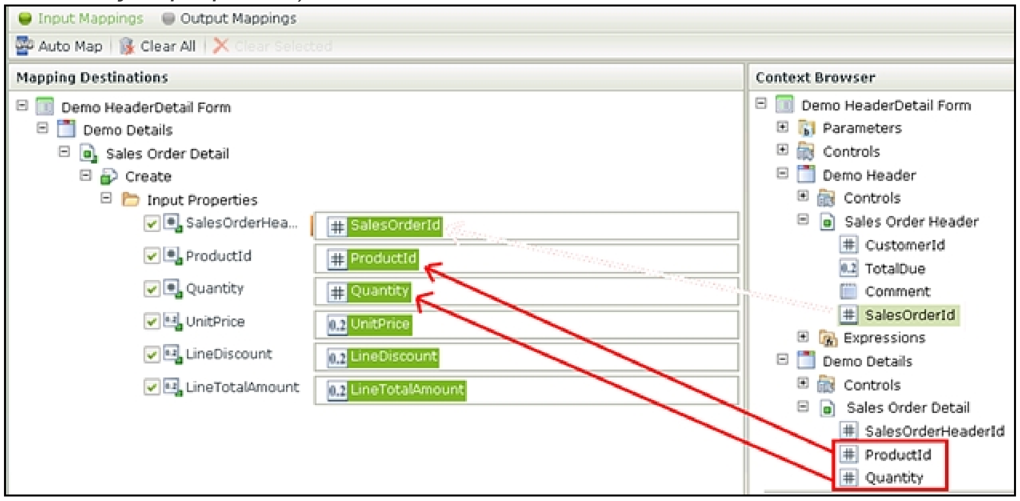
Next, when you create each detail item in the Form, pass the saved Header record ID into the appropriate linking property of the detail items, as shown below:

Passing the saved Header record's ID into the Detail item



When defining the Rule Action that maps the input properties into the records for the line items, note that you must map the input properties using the SmartObject that represents the detail items, <u>not</u> the controls on the Detail view. This is just how it works: behind the scenes, the Rule will know to repeat the action for each item in the List View.

Mapping input properties from the Details SmartObject (note: we do not use Controls in the mapping, we use the SmartObject properties)



Tip

Pay special attention to the Execution Block used to combine the Actions. In most cases, you would use the "in a batch" option so that the Header method and Detail method are all executed in the same Scope, and you would save the returned ID into a SmartObject property of the Header SmartObject.

You can also use the "one after another" option, but in this case, you would need to save the returned Header record's ID into a Control on the View or Form before passing it to the Detail records.

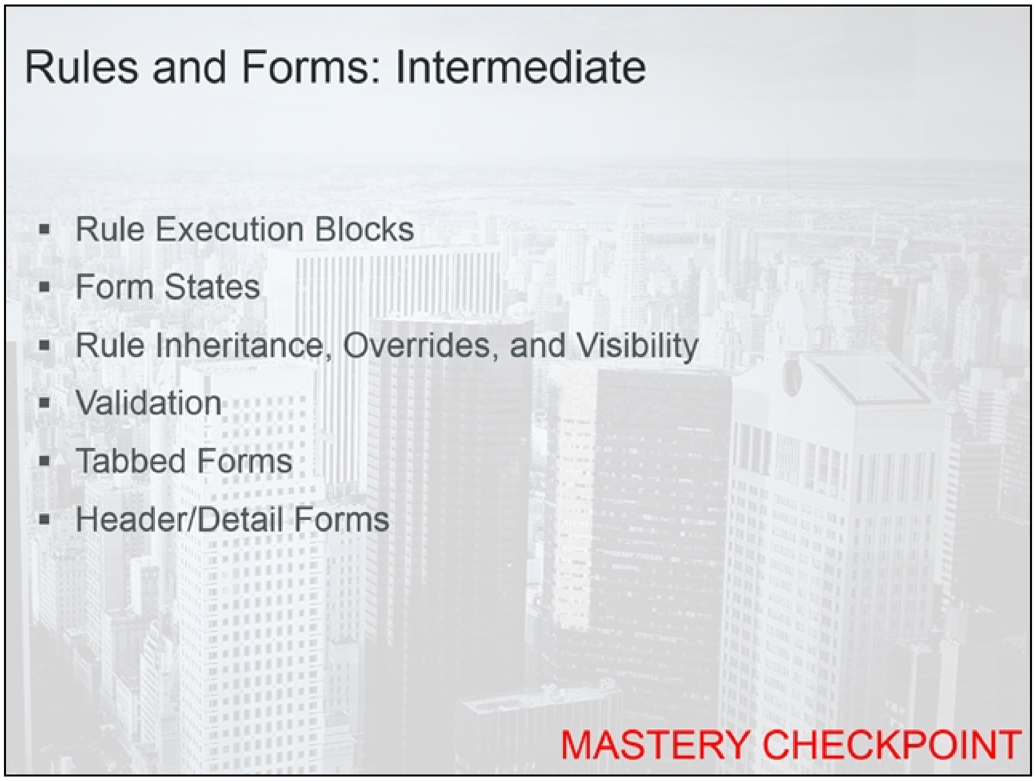
The difference between these two Execution Blocks is that "one after another" executes the first Action, returns to the Form and then executes the next action in a new "transaction" scope. The "in a batch" Execution Block executes all Actions sequentially but does not return to the Form between Actions.

The "in a batch" approach is slightly faster in execution than the "one after another" approach. You would normally use the "in a batch" approach for the header/detail style of processing, unless you had to save a property back to a Control on the Form before calling the next Action.

Summary

- When implementing header/detail Forms:
  1. You must use separate Views for the Header (master/parent) and the List Items (details/children).
  2. Use a batch/sequential action to create the header record first and save the returned ID somewhere. Then call the create method for each child and use the saved Header ID as an input parameter so the List Items are associated with the Header record.
  3. You must map the child/list item method's input properties to the SmartObject's properties, not the Controls on the List View.

MASTERY CHECKPOINT: Rules and Forms



Rules and Forms: Intermediate

- Rule Execution Blocks
- Form States
- Rule Inheritance, Overrides, and Visibility
- Validation
- Tabbed Forms
- Header/Detail Forms

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 3 of this module. If you are attending instructor-led training, use this as an opportunity to answer any questions around working with more advanced Forms and Rules.

After completing Part 3, you should know:

- How to configure Rule Actions based on Execution Blocks
- Form States and how States affect Rule Inheritance
- How Form Rules inherit View Rules, override View Rules and what is visible to Form-level Rules vs. View-level Rules
- How to implement more advanced validation
- How and when to implement Tabbed Forms
- How to implement header-detail Forms

Knowledge-check questions

Q: True or False: a View Rule can see Controls on the Form.
Reveal answer**A:** False. Form Rules can see View Controls, but View Rules cannot see Form Controls. (Rules work top-down: the parent can see the children, the children cannot see the parent.)

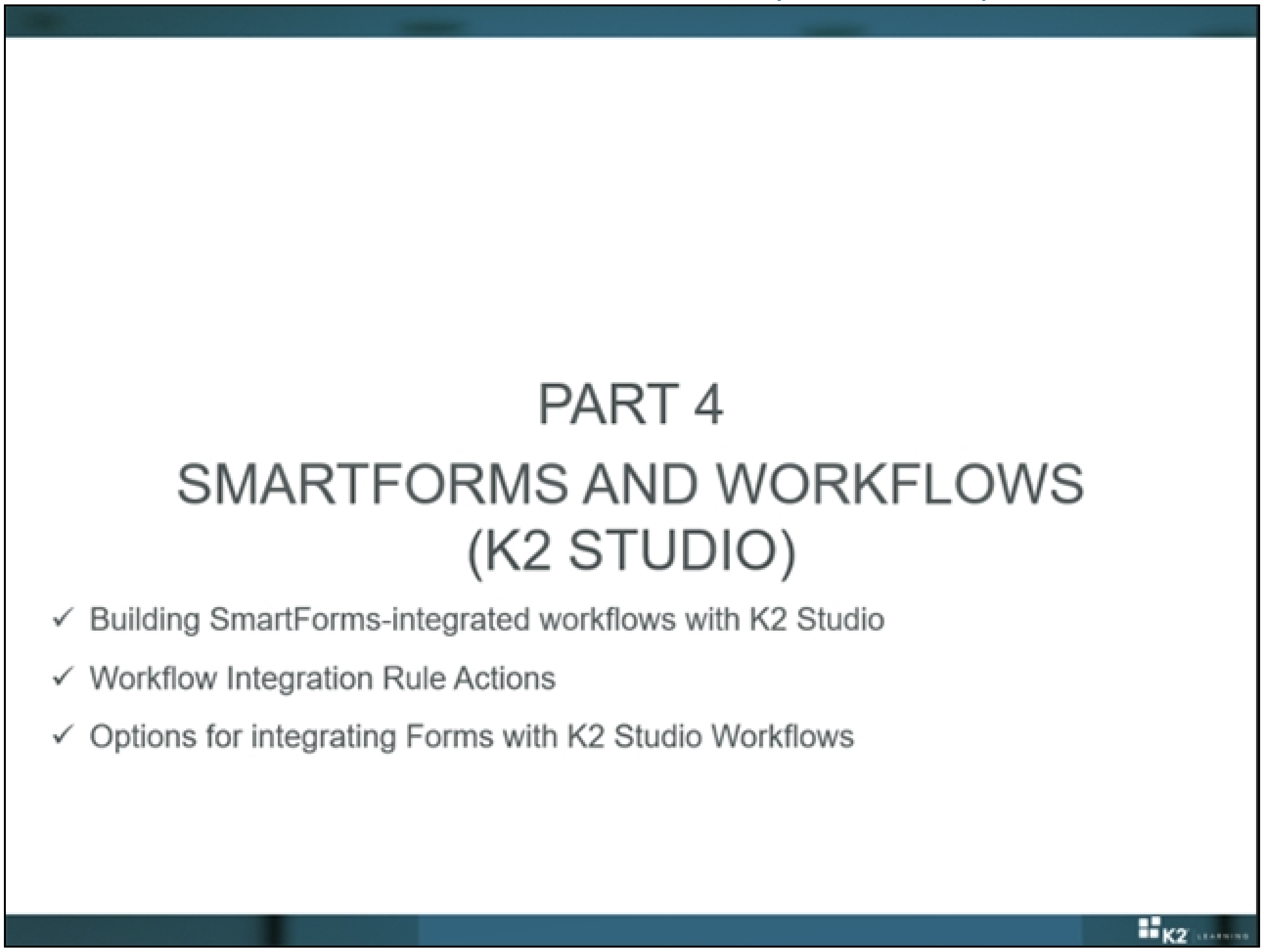
Q: When you define a custom validation expression, can you use that expression in other Forms?
Reveal answer**A:** Yes, custom validation expressions are available for re-use in any Form or View.

Q: True or False: a Form-level Rule can disable specific Actions in a View-level Rule.
Reveal answer**A:** True

Q: Why would one use Tabbed Forms?
Reveal answer**A:** To show multiple Views on the same Form.

Q: How can you use Tabs to improve the performance of a Form?
Reveal answer**A:** Place Views (especially List Views) on Tabs, then define Rules that only load the data on the View Tabs when that Tab is clicked.

PART 4: SmartForms and Workflows (K2 Studio)



PART 4

SMARTFORMS AND WORKFLOWS (K2 STUDIO)

- ✓ Building SmartForms-integrated workflows with K2 Studio
- ✓ Workflow Integration Rule Actions
- ✓ Options for integrating Forms with K2 Studio Workflows

Video

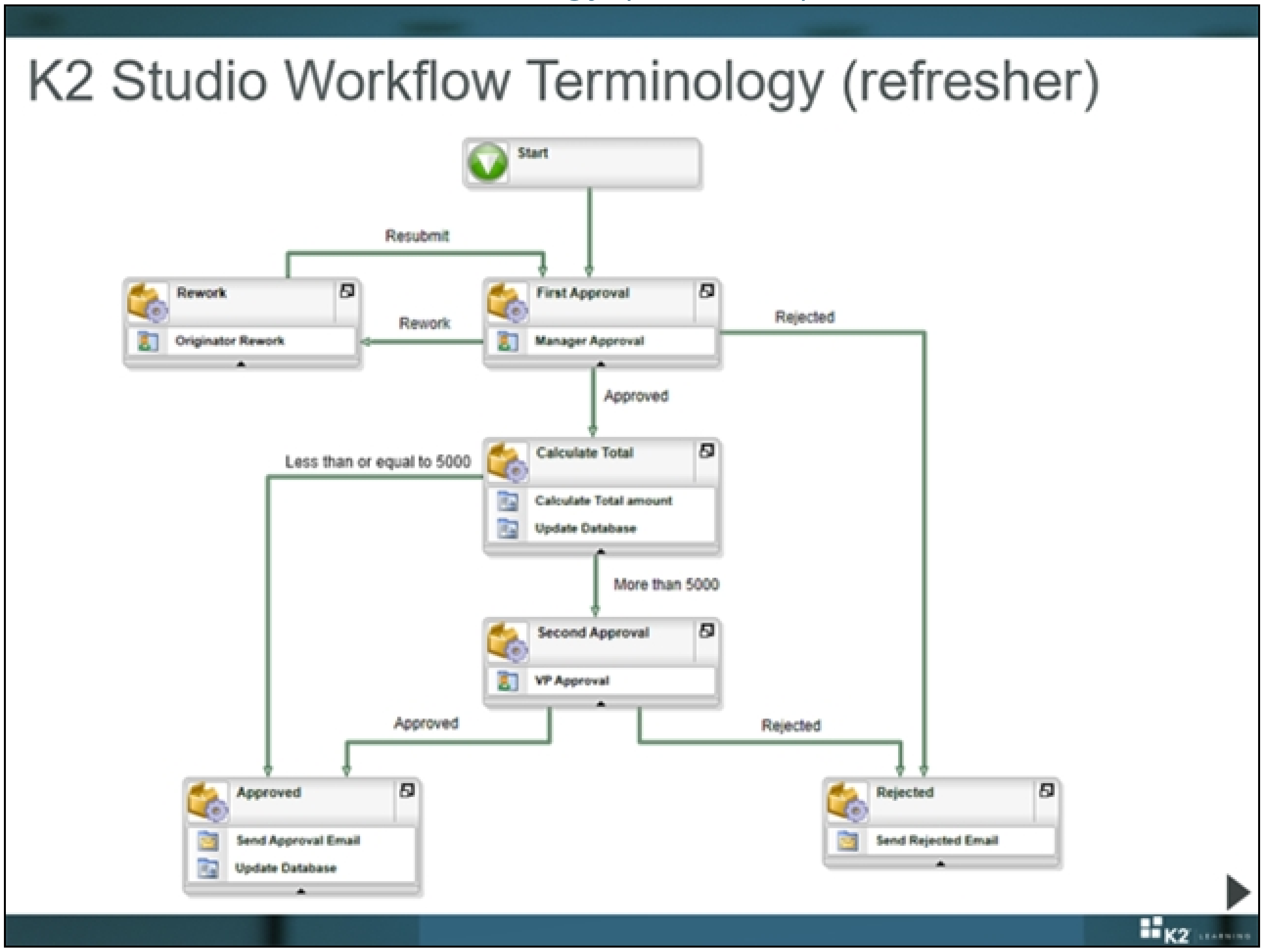
In Part 4 we will look at how to build SmartForms-integrated Workflows in K2 Studio. (And by inference, K2 for Visual Studio as well, since the integration with SmartForms is the same between K2 Studio and K2 for Visual Studio.)

Some of the concepts we will cover in this part include:

- The difference between Start Forms and Task Forms
- The Workflow integration Rule Actions used in SmartForms Rules
- Some of the options you have to integrate SmartForms with Workflows built in K2 Studio and K2 for Visual Studio

At the end of Part 4, we will build a very simple single-step approval Workflow that will only be started if the total order amount is over a certain limit.

K2 Studio Workflow Terminology (refresher)



Video

Note
This is a refresher of the K2 Studio Workflow terminology that was covered in the K2 Core training course. If you recently completed the Core training and are still familiar with K2 Studio Workflow terminology, you can move on to the next topic.

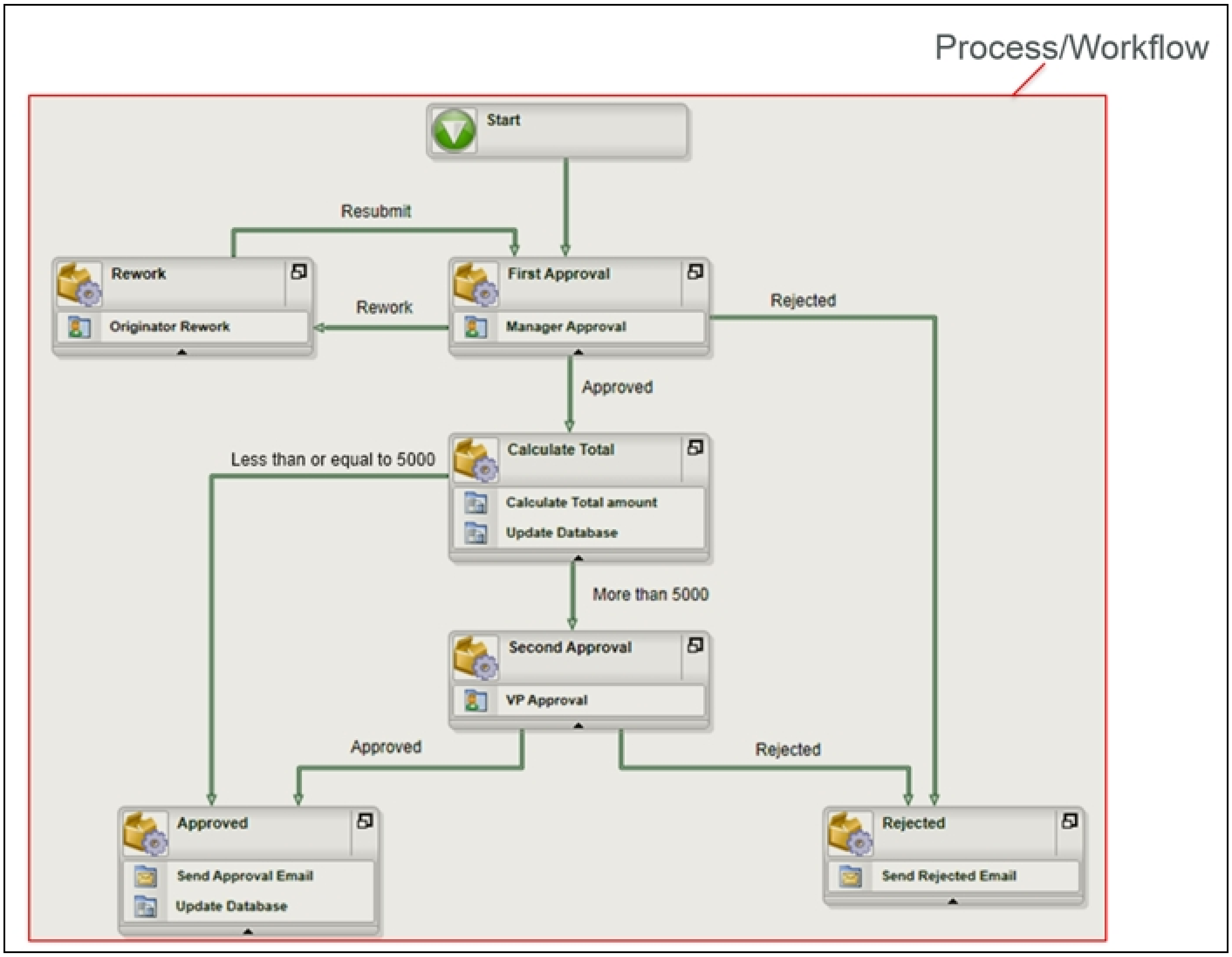
Before working with Workflows in K2 Studio, it is worthwhile to understand the terminology used to identify parts of the Workflow. Note that the terms used are slightly different when compared with the browser-based Workflow designer (K2 Designer). The K2 Studio Workflow designer offers features that are not available in the browser-based Workflow designer.

Process/Workflow

A Process (also known as a Workflow) refers to the entire Process designed in the K2 design tool. A Workflow will have at least a Start Activity and one other Activity (otherwise it wouldn't be a Workflow, right?). In K2, the terms "Process" and "Workflow" are used interchangeably, but they refer to the same thing: the entire Process with all of the Activities, Events, lines and properties that make up the Process. Similarly, the terms "Process Instance" and "Workflow Instance" also refer to the same concept: it is an instance of the Workflow design.

Processes usually contain multiple Activities, which are joined with lines. The Activities and lines generally define the flow of the Process. The Events inside the Activities define the work performed in the Process, and the Rules behind Activities define how the Activities are executed.

In the diagram below, all of the Activities, Events, lines (and the Rules behind each of these items) make-up the Process.

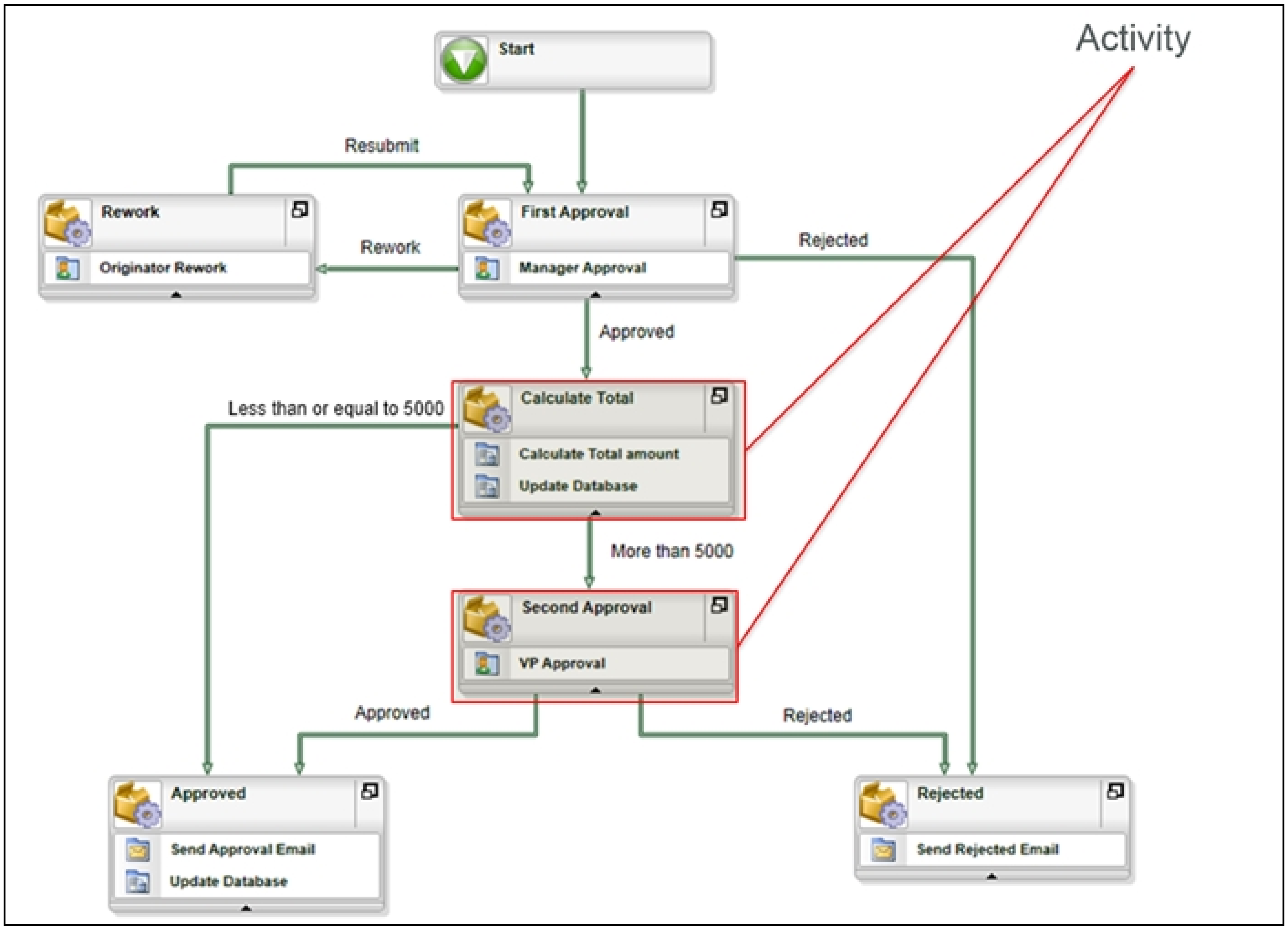


Activity

An Activity is a step in a Process, and acts as a container for work that will be performed by a user or a system. Whenever work moves from one person to another or to the K2 Server, a separate Activity will be created for each of these roles. (Strictly speaking, it IS possible to perform both user tasks and server tasks in the same Activity, but think of Activities as indicators representing that work moves from one human/system to another human/system.)

The actual work in the Activity is represented by Events in that Activity. Activities must always contain at least one Event but could contain more than one Event.

In the diagram below, two Activities are highlighted. In this case, the Activity **Second Approval** is performed by a user, and the Activity **Calculate Total** contains two server-side Events performed by the K2 Server.



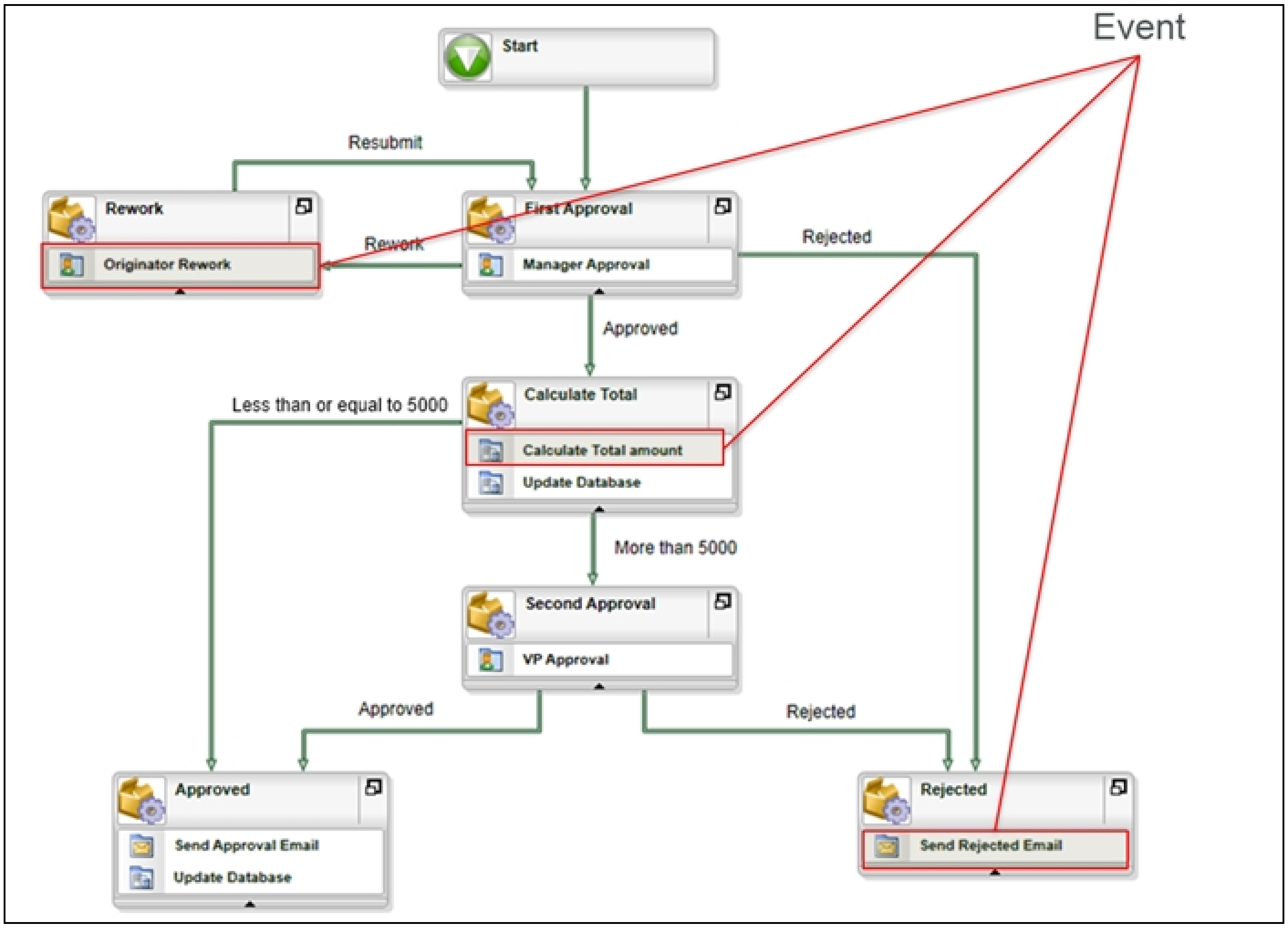
It is very common to define various Rules for an Activity that will control how that Activity should execute at runtime. Activities may contain Rules that define IF and WHEN that Activity should execute, WHO will perform the task if the Activity must be completed by a user and WHAT the possible outcomes for the Activity could be. A Process will always have at least one Activity that follows from the Start Activity.

Event

An Event is a unit of work in a Process, which is performed by a human or system. Work performed by a user is known as a Client Event and work performed by the K2 Server is known as a Server Event.

Events must always be contained inside Activities, and the Event will describe the work that is being performed. K2 provides many wizards to make the setup of these Events very easy; for example, the E-mail Event wizard which allows the designer to configure an E-mail that will be sent as part of the Process. This wizard gathers input from the designer and lets them drag and drop variables or type text into the various properties on an E-mail message.

In the diagram below, three Events are highlighted. In this case, **Originator Rework** is performed by a user, and the other Events **Calculate Total amount** and **Send Rejected Email** are performed by the K2 Server.



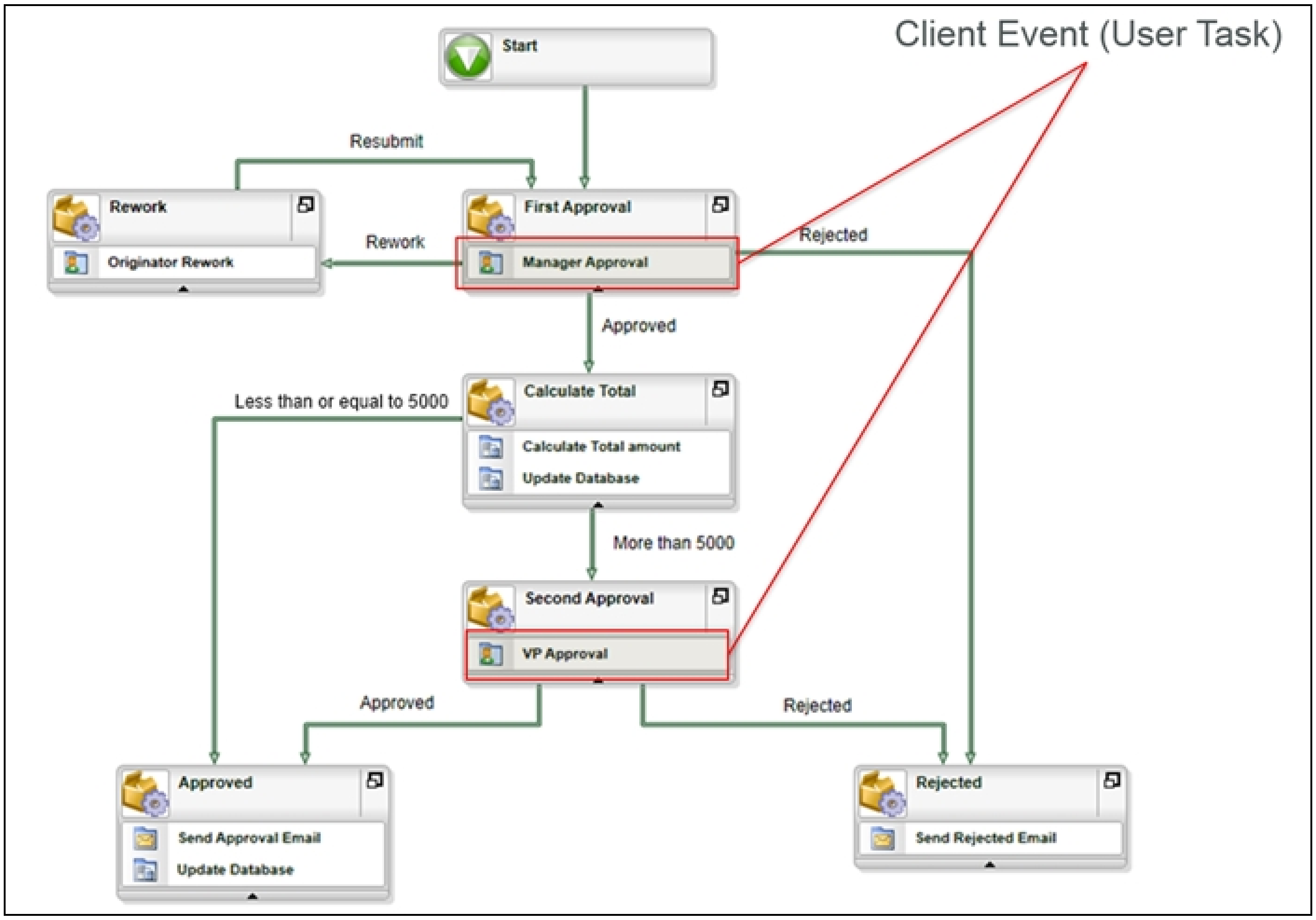
Notice that the **Calculate Total amount** and the **Approved** Activities each contain two Events. Activities can contain as many Events as needed to complete the step, and for server-side processing it is very common to have multiple Events in the same Activity.

Client Event

A Client Event is a task which is performed by a human. All tasks that are performed by humans must be allocated to a Destination (User or Users) which is defined on the Activity level. This Destination determines who will ultimately get the K2 task on their task list.

Most often, user tasks require the participant to make some decision: perhaps they need to approve or reject a request, or send it back for rework. Users do not always need to select a decision though. In some cases, the participant may only need to indicate that the task has been completed for the Workflow to continue. The key, however, is that a person had to tell the Workflow to continue. Any Process that is currently at a Client Event will remain at that step until the user completes the task, or an escalation causes the task to be expired. (A Process administrator may also override the Process and force the Workflow to go to another step. For more information, see the learning modules that deal with K2 Administration.)

In the diagram below, two user tasks have been highlighted: **Manager Approval** and **VP Approval**. Each of these tasks will be completed by the relevant users with one or more user interfaces.

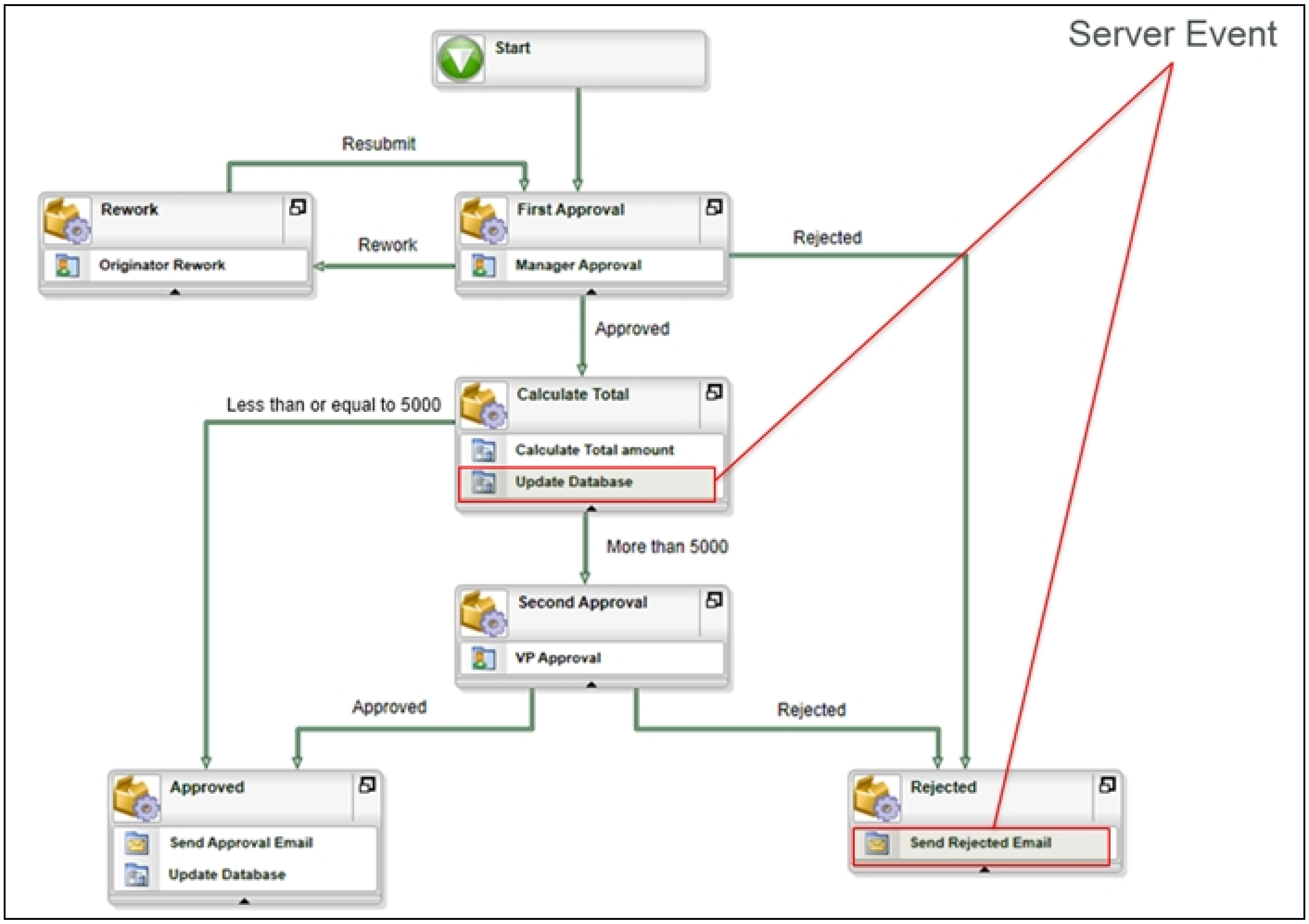


Note that all Client Events must have a Destination set on their containing Activity before the Process can be deployed. After all, if a task is assigned to a user, K2 must know who that user is. This Destination could be a hard-coded username, based on a group in Active Directory or SharePoint; a Role defined in K2, or could be looked up in some external system. The K2 Workflow doesn't really care HOW you determine who should do the work, it just cares that some username or group name is returned that tell K2 who to assign the work to.

Server Event

Server Events are tasks which are performed by the K2 Server. These tasks are usually not long-lasting and will complete in a few seconds, at most. Server Events could involve integration with some system, and the exact nature of that interaction depends on the wizard or SmartObject method being executed in the Event. Server Events are also used to perform some server-side processing such as performing calculations or manipulating data.

In the diagram below, two Server Events have been highlighted: in the **Update Database** Event, K2 is using some code to interact with a database, and in the **Send Rejected Email** Event, K2 is generating and sending an email.

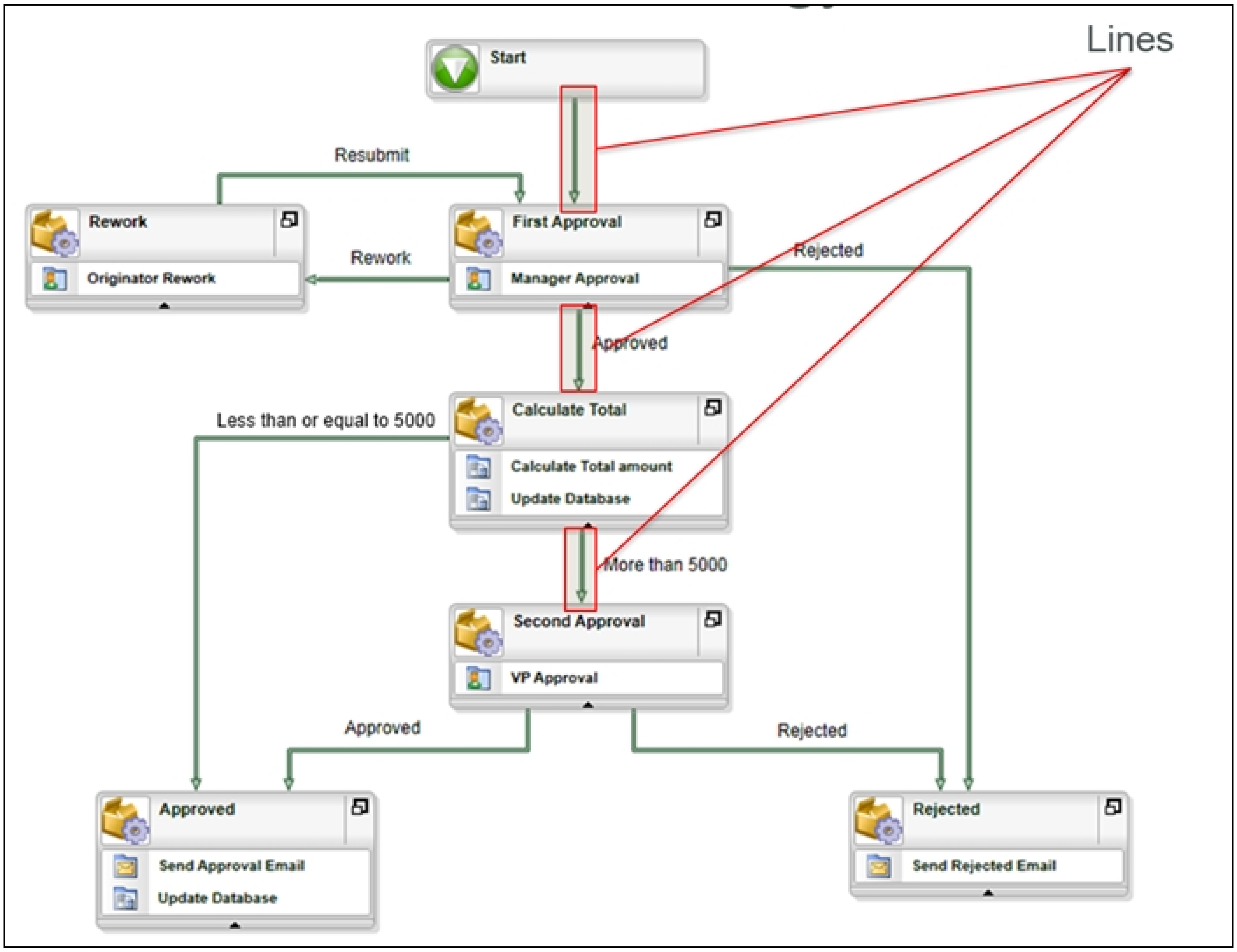


Lines

Lines are the possible paths in a Workflow. Lines are used to join Activities in a specific sequence, and effectively tell K2 how to move from one Activity to another.

Lines control the “flow” of a Process, and by using Line Rules or Outcomes, Workflow designers can tell K2 to follow a certain path in the Process based on some condition. Lines can even be used to split and merge a Process if parallel work is required.

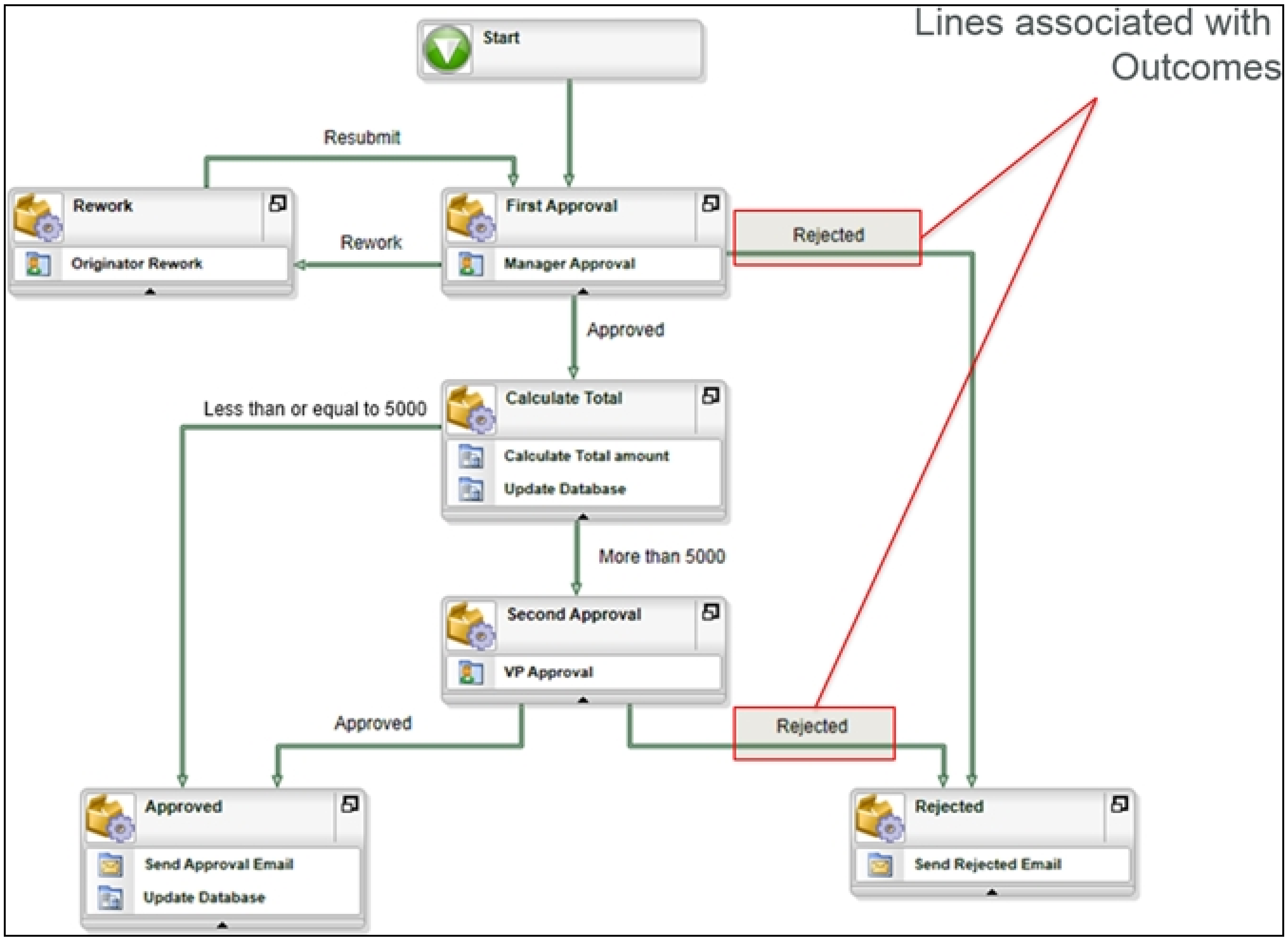
In the diagram below, three different types of lines have been highlighted. The first line joins the **Start** box to the **First Approval** Activity, and will always be followed since there are no Rules defined for that line. The **Approved** line is a special type of line called an Outcome, and is evaluating the decision made by the approver. The **More than 5000** is a Line with a Line Rule, which is evaluating a data value in order to make a decision about where to go next. In this example, if the amount is more than 5000, the Workflow must go to the **Second Approval**.



Outcomes

Outcomes are a special type of Succeeding Rule and usually surface on a Workflow as Lines. Outcomes are usually based on one or more user's decisions (Actions) or by evaluating user actions along with other values to establish a path to follow in the Process.

In the diagram below, two lines that follow certain Outcomes have been highlighted. The first **Rejected** line is one of three possible Outcomes resulting from the action taken in the **Manager Approval** Client Event. The second **Rejected** line is one of two possible outcomes resulting from the action taken in the **VP Approval** Client Event.

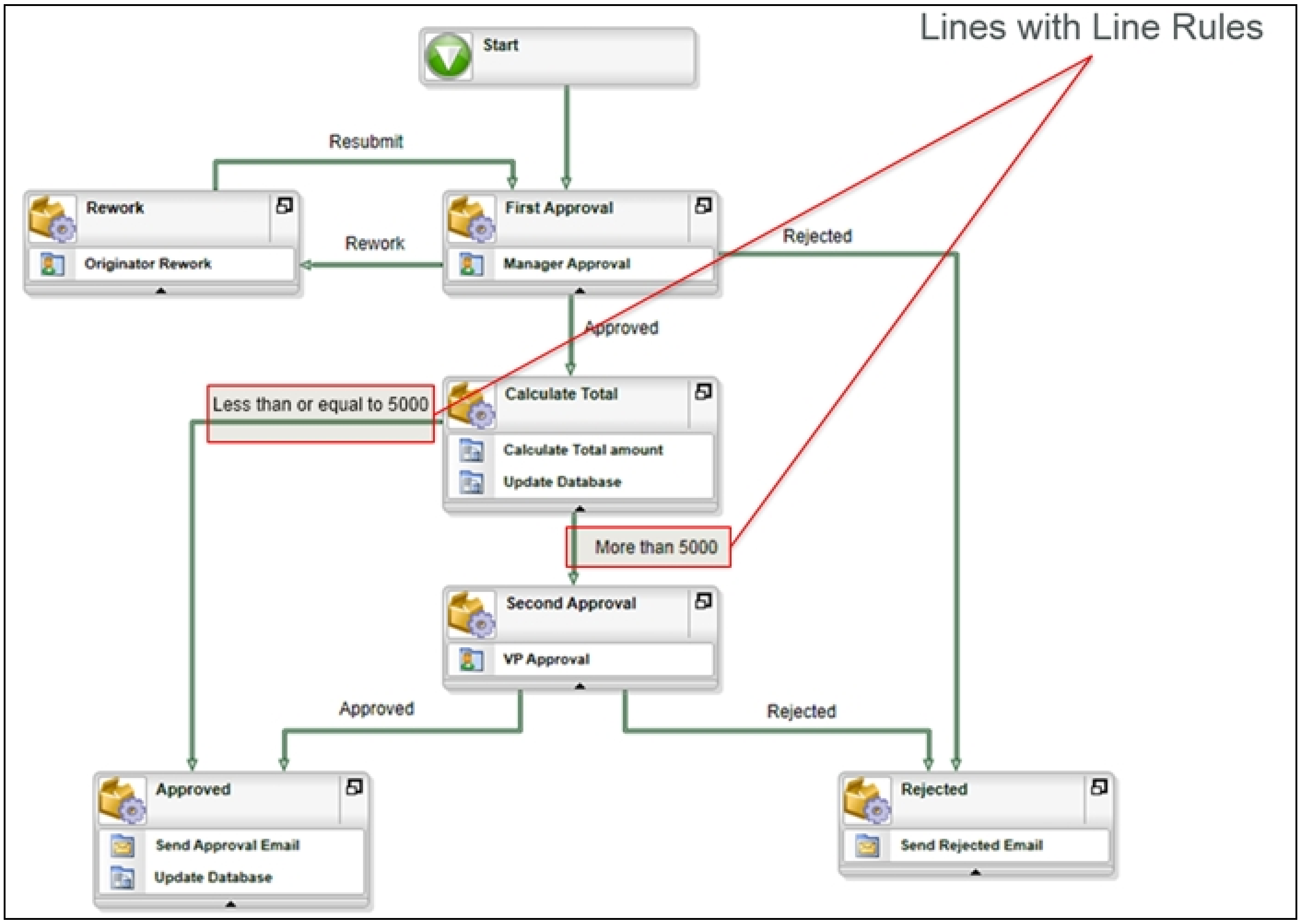


Behind the scenes, Outcomes are defined as Succeeding Rules for Workflow Activities, and the resulting Outcome is saved to an Activity-level datafield. The Lines that flow from the Activity evaluate this Activity datafield. Even though we have indicated "Outcomes" on the diagram below as lines, strictly speaking, Outcomes are actually defined on an Activity level.

Of course, Outcomes and Lines can be a lot more complex than the example above, such as multiple parallel lines, Outcomes that evaluate both user input and data values, and so on. For the purposes of this discussion, just remember that in most cases, Actions yield Outcomes which yield Lines.

Line Rules

Workflow designers can define additional Line Rules behind Lines to control the flow of a Process. Most often this is used to control the path of a Process based on the result of some comparison operation. In the diagram below, two Lines with Line Rules have been highlighted - each line checks the Total value of the request. If the value is less than or equal to 5000, the Process goes directly to the **Approved** Activity, but if the value is more than 5000, the Process goes to the **Second Approval** Activity.



Escalations

Escalations are tasks performed by the K2 Server in response to an expected time frame being exceeded. Escalations can be configured for Events, Activities and the Process as a whole, and will only fire if that Event, Activity or Process has not completed within the specified interval (time frame).

Events, Activities and the Process could have several escalations that can be repeated as many times as desired. This makes it possible to design escalating levels of escalations.

An example of an escalation is to send an E-mail to the originator of the Process if the first approver has not completed the task within two days of the Process being started. You could add another escalation to the first approval Activity to forward the task to another user if the original approver hasn’t completed the task after three days.

Notifications

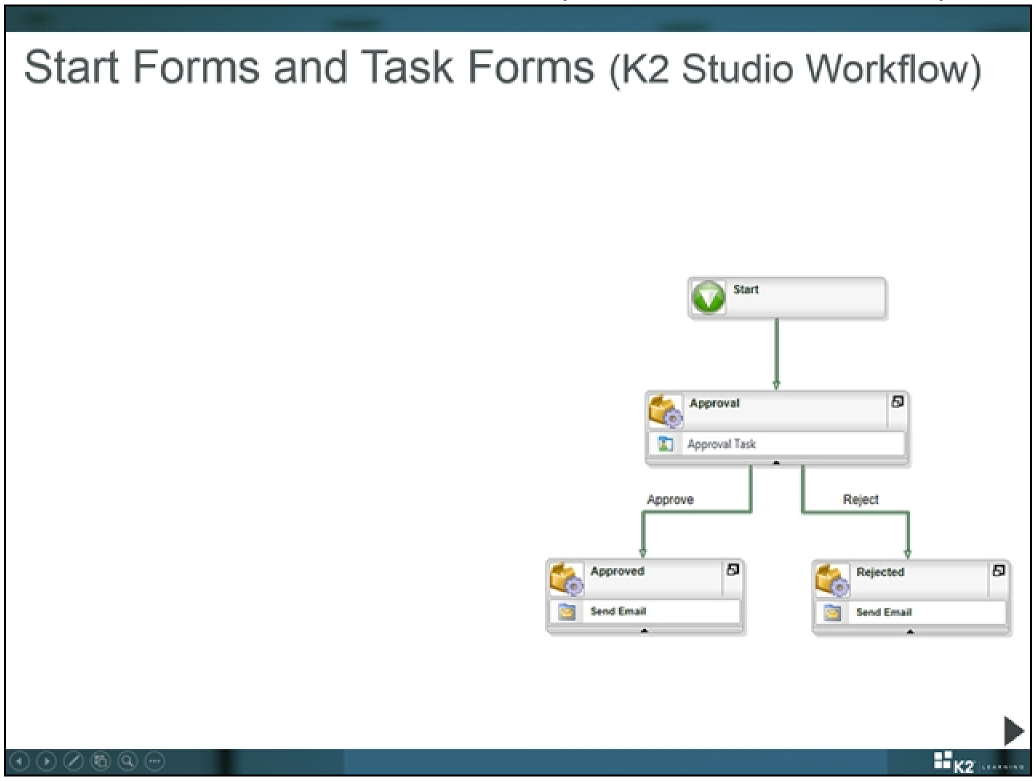
Notifications are E-mails that are sent to users during Process execution. Notifications are typically used to alert participants that they have new tasks, and are used in escalations as well. Both the styling (for example, applying custom headers and footers or fonts) and content (the message itself and the data included in the message) of the notification can be customized. *SmartActions* enable users to complete their tasks just by replying to the task notification E-mail that was sent to them. The Workflow designer can decide whether to allow users to complete tasks this way, or whether they must always open the Form to review the task before they can complete the task.

Summary

- Process/Workflow refers to the Activities, Events, Lines and Properties that make up the Process
- Activities are "steps" in the Workflow joined by Lines and contain one or more Events
- Events are the tasks/work that gets done in an Activity and could be Client Events (human tasks) or Server Events (system tasks).

- Lines connect Activities and define the flow of a Workflow
    - Lines can be associated with Outcomes (e.g. the result of the decisions/actions taken in a Client Event).
    - Lines can contain Line Rules that perform some logic to determine Workflow routing

Start Forms and Task Forms (K2 Studio Workflow)



Video

Note
The behavior of Start Forms and Task Forms for Workflows was discussed in the 100.YYZ SmartForms Fundamentals course, but using a K2 Designer Workflow. Here we will apply the same concepts to a Workflow built in K2 Studio.

The integration between Forms and Workflows boils down to using Rules and Actions to start Workflows and open/-complete Workflow tasks. When using SmartForms with Workflows, it is helpful to understand the relationship between the Workflow and the Form being used to interact with the Workflow. For example, you might use one Form to start a Workflow, but a completely different Form to open and complete a Workflow task. Or, you may want to use the same Form to both start the Workflow and complete a task in the Workflow, but use different States to determine whether the Form is starting the Workflow or completing a task.

There are separate Rule Actions to interact with Workflows, depending on what you want to do with the Rule. It is easiest to think in terms of Start Forms and Task Forms. Start Forms use a single Action to start a Workflow, while Task Forms use two Actions: one to open the worklist item based on a Serial Number when the Form is opened, and another Action to complete the task.

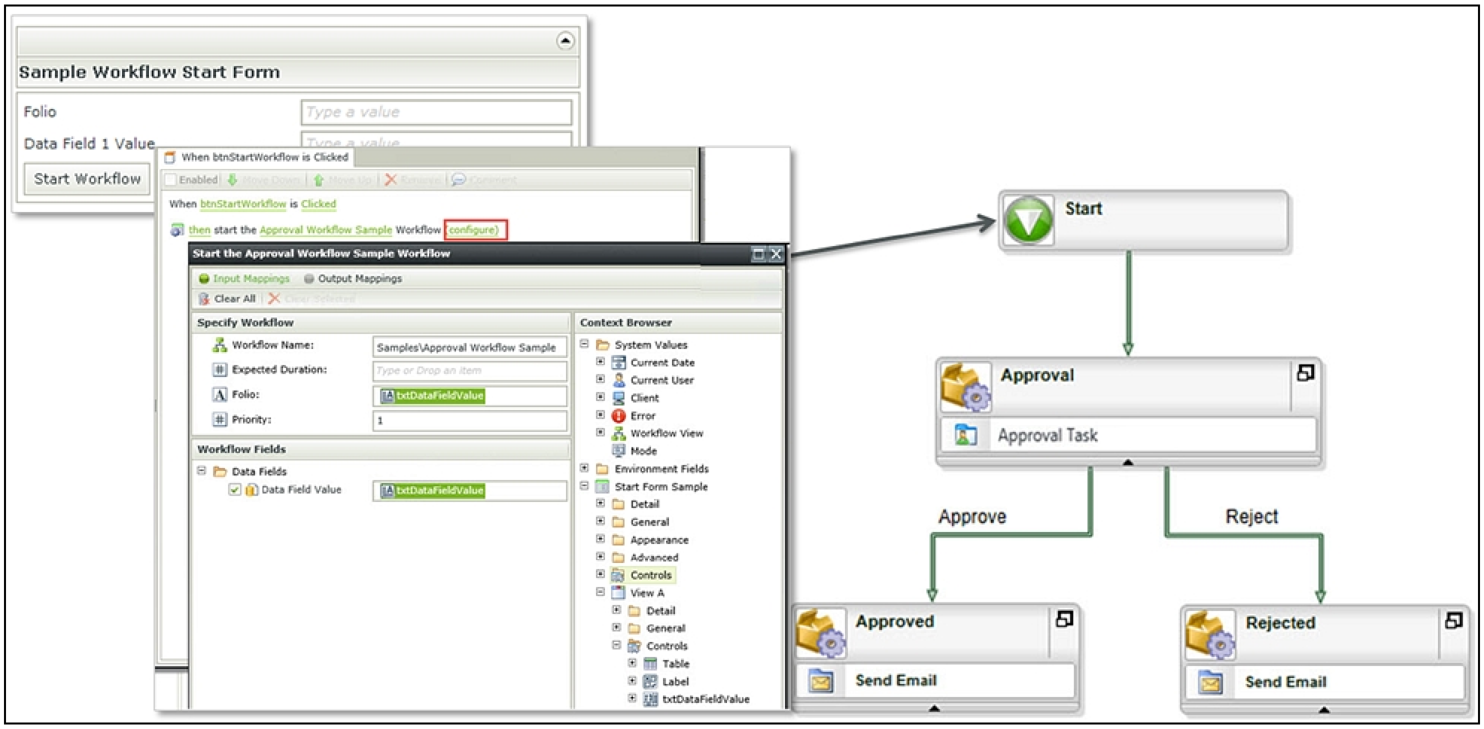
Start Forms

Start Forms use the Start a Workflow Action to start a Workflow. This Action is usually contained in a Rule for a button Click event or some other user-driven event. In the example below, the Start Workflow button’s Rule calls the Start Workflow Action to start the sample Workflow.

The Start Form is typically opened and completed by a user before the Workflow even exists: the Workflow is only started when the user takes some action on the Form. These Start Forms are usually “capture Forms” or “user input Forms” where the user will complete the Form with information before submitting the Form. In the majority of cases, the submit Action will save the data to some data store by calling a SmartObject method, and then start a Workflow in the same Rule. While these input Forms are normally purpose-built for Workflow, remember that you can re-use Views on the Form to save development time; you can also use Rule conditions to start Workflows based on some condition.

The Start Form would use a Rule calling the "Start a Workflow Action" to initiate the Workflow. As part of the configuration of this Action, you can set the Folio of the Process that will start, set any data fields in the Process, set the priority of the Process Instance, and the expected duration.

A Form Rule used to Start a Workflow



Sample Workflow Start Form

Folio	Type a value
Data Field 1 Value	Type a value

When btnStartWorkflow is Clicked

When btnStartWorkflow is Clicked

then start the Approval Workflow Sample Workflow (configure)

Start the Approval Workflow Sample Workflow

Specify Workflow

Workflow Name:	Samples\Approval Workflow Sample
Expected Duration:	Type or Drop an item
Folio:	txtDataFieldValue
Priority:	1

Workflow Fields

Data Fields	
✓ Data Field Value	txtDataFieldValue

Task Forms

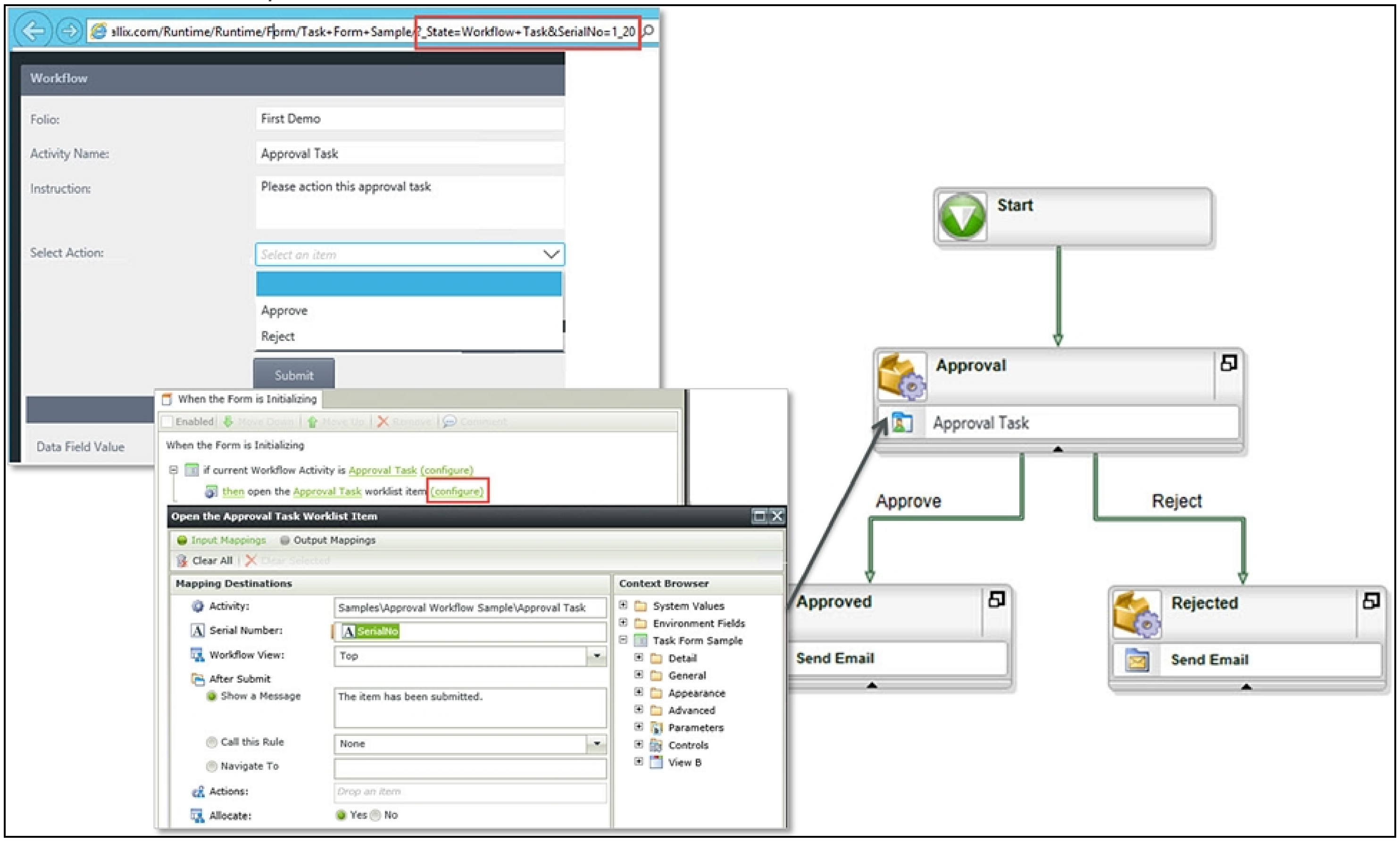
A Form used to complete a Workflow Task (also known as a worklist item) usually contains two Workflow-specific Actions: one Action opens the Workflow Task when the Form is initialized, and another Action completes the Workflow Task when the user takes an action on the Form. (Remember that Actions are contained in Rules, and these Rules would be configured to fire at the appropriate times.)

The first step is to open the associated task with the "Open a worklist item" Action, using the Serial Number to locate and open the Workflow Task. This Rule is usually executed when the Form is initialized, and would usually also read the available Actions for the task so that these can be presented to the user in some way, normally a drop-down menu. You can also read data from the Workflow Instance, like the Folio or data field values.

This next image shows that when the Form associated with the Approval step is opened, then open the worklist item for this task. When configuring this Rule, you will have to provide the Serial Number for the task: this Serial Number is a unique value that is usually passed as a Form parameter. We have highlighted the Serial Number in the screenshot below to show where it is usually found.

Note
By default, opening a worklist item allocates the item to this user and they either need to complete the task or release it.

A Form Rule used to open a User Task worklist item


allix.com/Runtime/Runtime/Form/Task+Form+Sample/?_State=Workflow+Task&SerialNo=1_20

Workflow

Folio:	First Demo
Activity Name:	Approval Task
Instruction:	Please action this approval task
Select Action:	Select an item

Approve
Reject

Data Field Value

When the Form is Initializing

if current Workflow Activity is Approval Task (configure)
then open the Approval Task worklist item (configure)

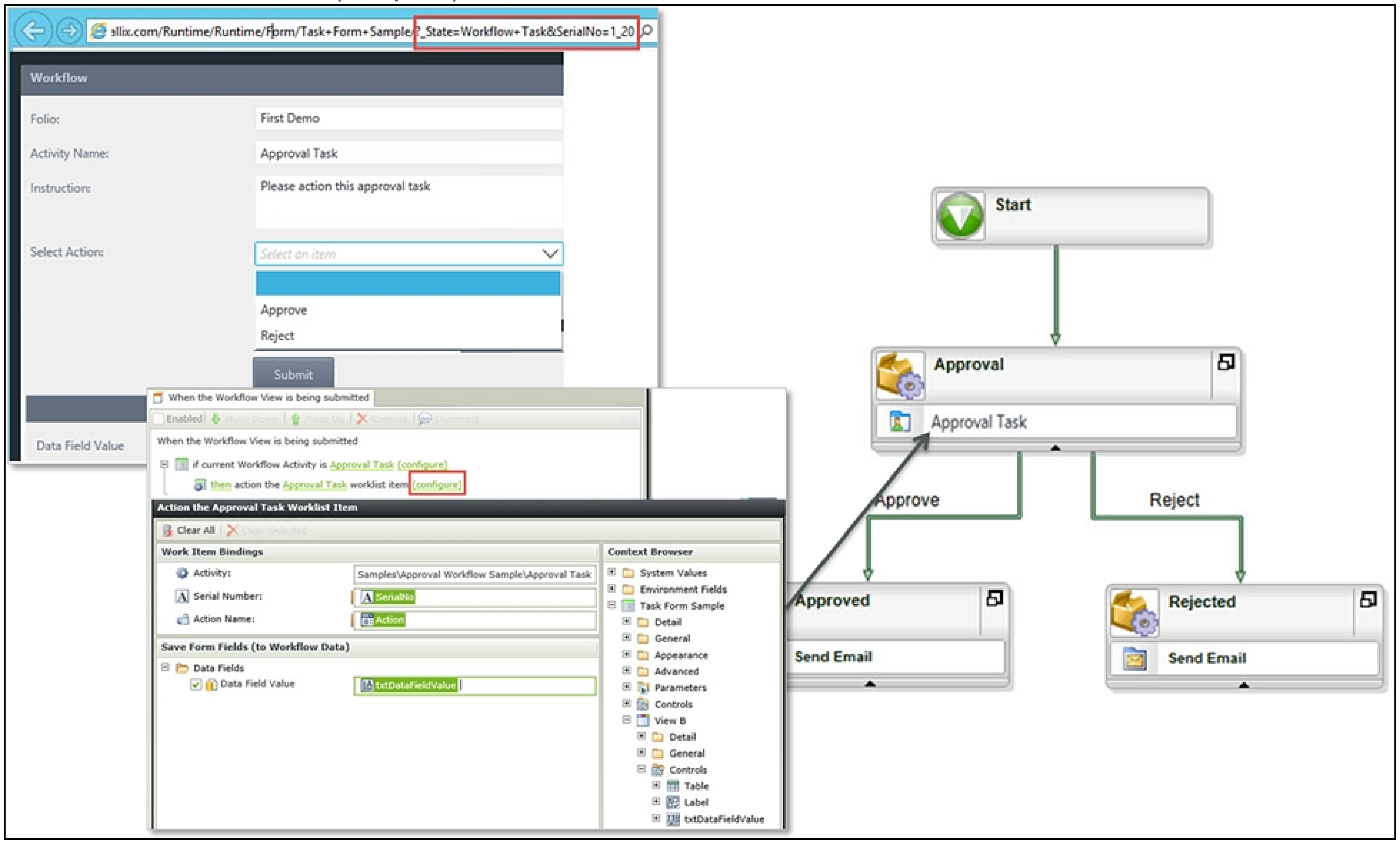
Open the Approval Task Worklist Item

Mapping Destinations	
Activity:	Samples\Approval Workflow Sample\Approval Task
Serial Number:	SerialNo
Workflow View:	Top
After Submit	
Show a Message	The item has been submitted.
Call this Rule	None
Navigate To	
Actions:	Drop an item
Allocate:	[✓ Yes] No

The second part is executed when the user wants to complete the task, normally with a button Click event. This part would use the "Action a Worklist Item" Action to complete the Workflow Task. You would normally pass the Action that the user selected through as part of the Action configuration so that the Workflow knows which way to go next. As part of the Action, you can also update the Process Folio and Process data field values.

Caution
The Action name used in the configuration for this step must exactly match the Action name defined in the Workflow, otherwise the Action will not complete successfully.

A Form Rule used to Action (complete) a User Task worklist item



Folio:	First Demo
Activity Name:	Approval Task
Instruction:	Please action this approval task
Select Action:	Select an item (Approve, Reject)

When the Workflow View is being submitted

When the Workflow View is being submitted
- if current Workflow Activity is Approval Task (configure)
  - then action the Approval Task worklist item (configure)

Action the Approval Task Worklist Item

Work Item Bindings

Activity:	Samples\Approval Workflow Sample\Approval Task
Serial Number:	SerialNo
Action Name:	Action

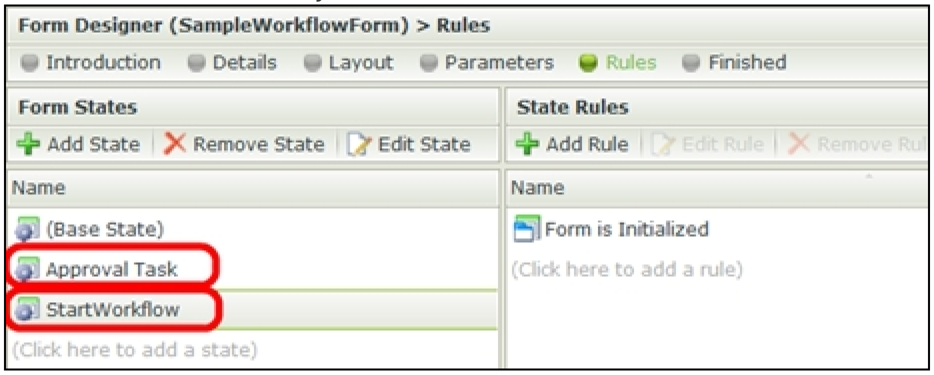
Save Form Fields (to Workflow Data)

Data Fields	
Data Field Value	txtDataFieldValue

Form States and Workflow integration

You should recall that Form States can be used to render the same Form in different ways depending on how the Form is being viewed. When you Workflow-enable a Form, Form States are used extensively, and usually the Form would have separate States for each step in the Workflow. For example, one State is used when the Form starts a Workflow, another State is used when the Form is used for a Client Event. Additional States are added for each of the Client Events that the Form is integrated with. This is very similar to the approach used in InfoPath Forms where different Views are used for different steps of the Workflow, but the difference is that with SmartForms, you do not have to define separate Views for each step. Instead, you will use different States for the same Form. K2 adds Rules for the States so that the Form can be rendered correctly based on the current activity.

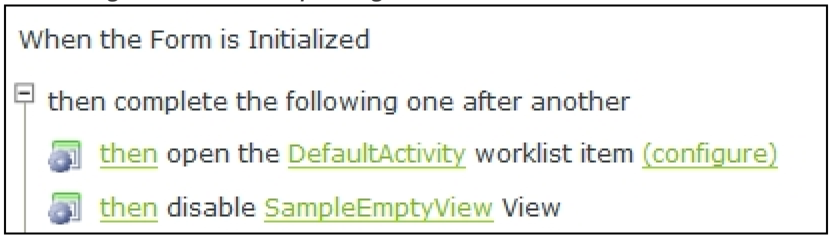
States added and used by the Workflow wizards



Name	Name
(Base State)	Form is Initialized
Approval Task	(Click here to add a rule)
StartWorkflow	
(Click here to add a state)	

You may want to define additional Rules on the Form and use the View Manipulation Actions we discussed earlier to control the Forms based on the current State. For example, you could use the “Disable a View” Action to prevent an approver from making changes to the data that was submitted for approval, but only if the Form State is equal to Approval. You will have to do this manually: K2 will not automatically disable the View for a user task because it does not know which fields you want to enable and which fields you want to disable.

Disabling a View after opening a Workflow Task


When the Form is Initialized

then complete the following one after another

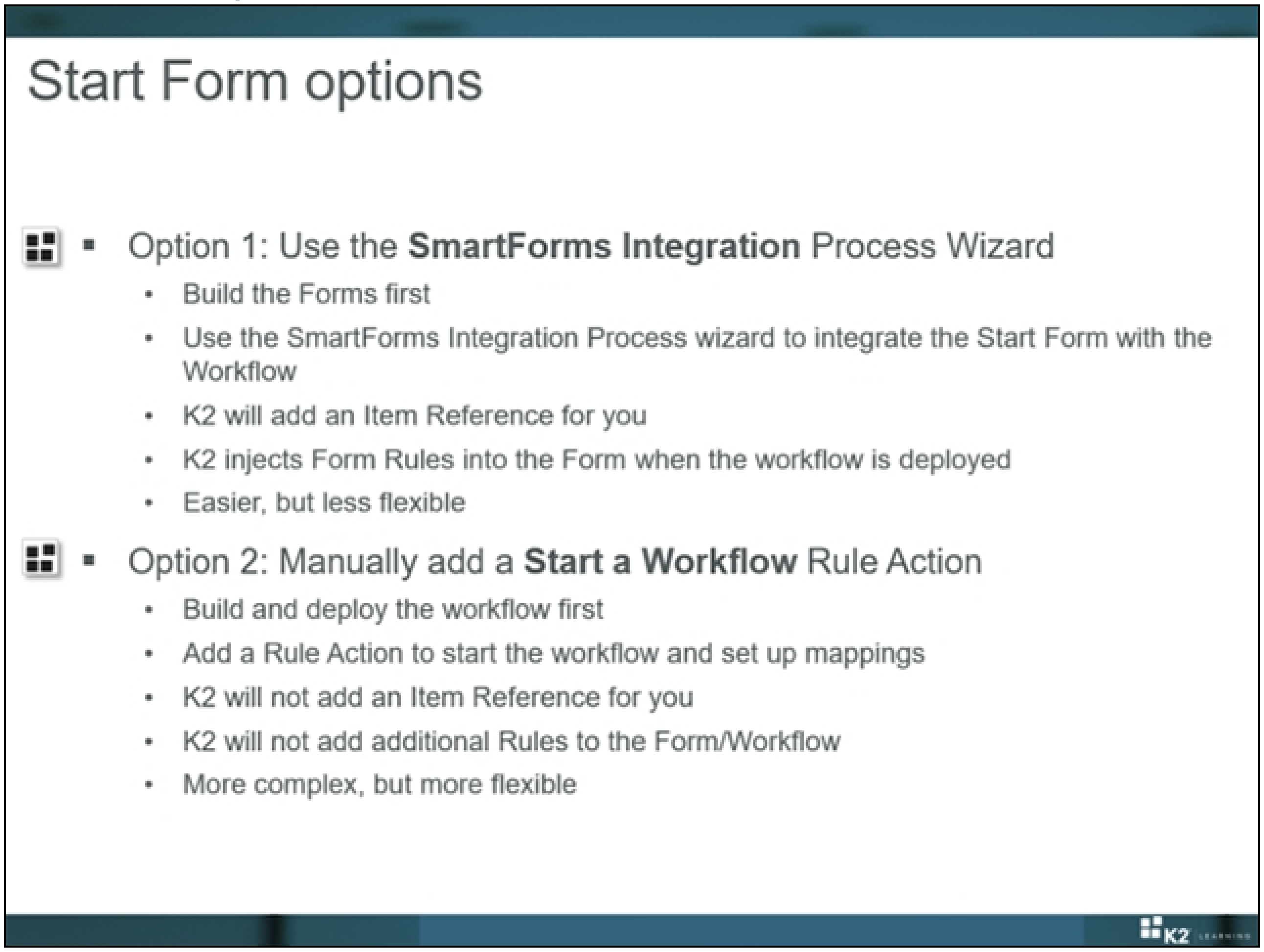
then open the DefaultActivity worklist item (configure)

then disable SampleEmptyView View

Summary

- There are different Rule Actions to interact with Workflows:
    - Start Workflow
    - Open a worklist item (requires the task Serial Number, usually a parameter in the query string)
    - Complete (Action) a worklist item (requires the Serial Number)
- You can use the same Form to both start a Workflow and complete a user task later in that Workflow
    - Usually this is achieved with States and changing the behavior (i.e. Rules) of the Form based on the State

Start Form options



Start Form options

- Option 1: Use the **SmartForms Integration** Process Wizard
  - Build the Forms first
  - Use the SmartForms Integration Process wizard to integrate the Start Form with the Workflow
  - K2 will add an Item Reference for you
  - K2 injects Form Rules into the Form when the workflow is deployed
  - Easier, but less flexible
- Option 2: Manually add a **Start a Workflow** Rule Action
  - Build and deploy the workflow first
  - Add a Rule Action to start the workflow and set up mappings
  - K2 will not add an Item Reference for you
  - K2 will not add additional Rules to the Form/Workflow
  - More complex, but more flexible

Video

You have two options when integrating a SmartForm to start a K2 Studio/K2 for Visual Studio Workflow:

1. Use the SmartForms Integration Process Wizard
2. Manually configure the Workflow Start Rule Action using Rule Actions

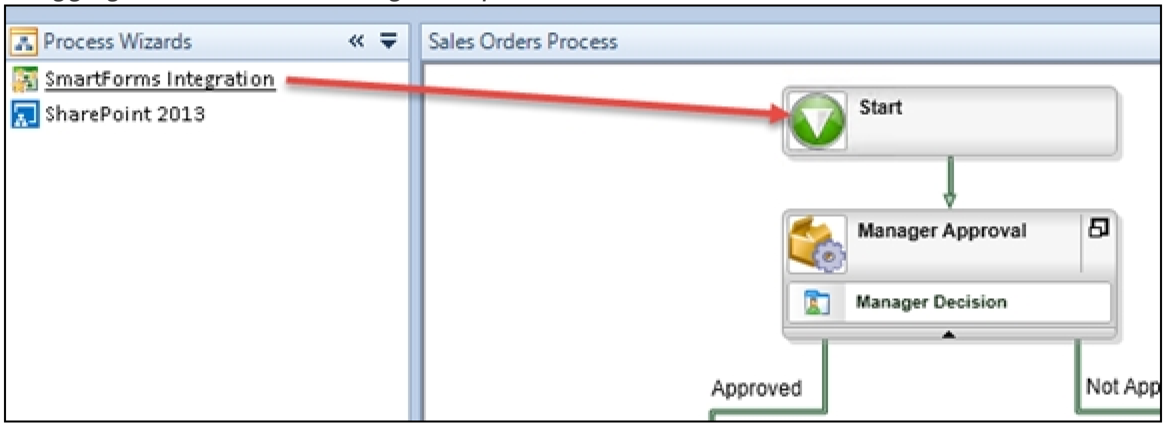
Which approach you should use depends on your requirements: the first is easier because most of the integration work is done automatically for you. The second approach is more flexible but requires more manual configuration to set up. Let's look at the options in more detail.

Using the SmartForms Integration Process Wizard

The easiest way to integrate a Form with a K2 Studio Workflow is to use the **SmartForms Integration** Process wizard. This wizard will add the necessary Rule Actions to start the Workflow and transfer data into the Workflow for you. The Form modifications will be applied to the Form when the Workflow is deployed. With this approach you have to build the Forms first and then the Workflow. It is the easier approach, but you don't have full control over the Form-Workflow integration.

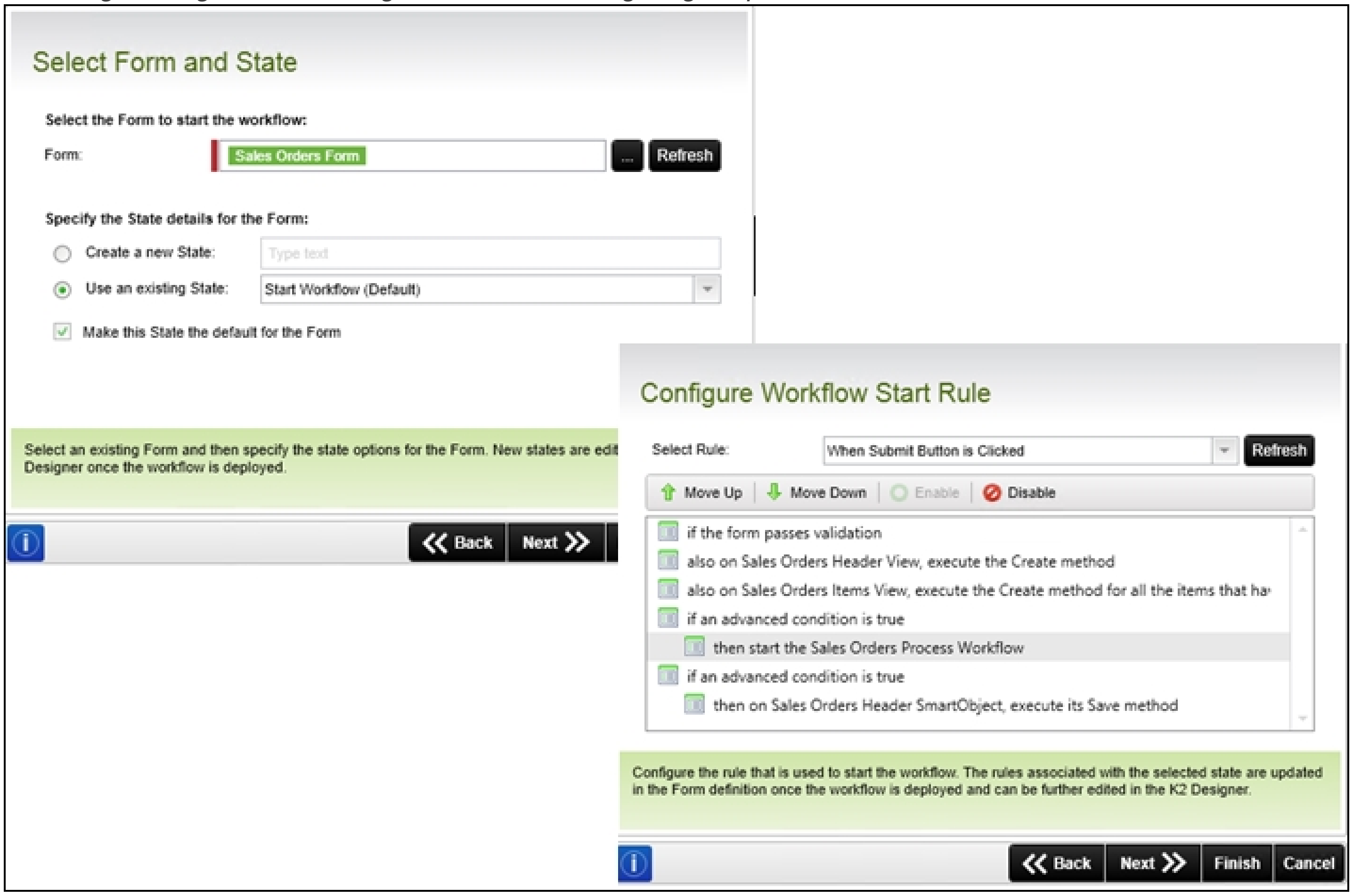
The Start Workflow Action will allow you select a Workflow to start, and set the Folio and Workflow data field mappings as part of the wizard.

Dragging the SmartForms Integration process wizard into the Workflow



Once you run the wizard, you can point it to the Form you want to use to start the Workflow. As part of the wizard, you can decide whether to inject the Start Workflow Action into an existing State in the Form or create a new State. You can also decide which existing Rule the Start Workflow Action should be added to.

Selecting the target Form, setting the State and configuring the placement of the Start Workflow Action



Select Form and State

Select the Form to start the workflow:

Form: Sales Orders Form

Specify the State details for the Form:

Create a new State: Type text

Use an existing State: Start Workflow (Default)

✓ Make this State the default for the Form

Select an existing Form and then specify the state options for the Form. New states are edit
Designer once the workflow is deployed.

Configure Workflow Start Rule

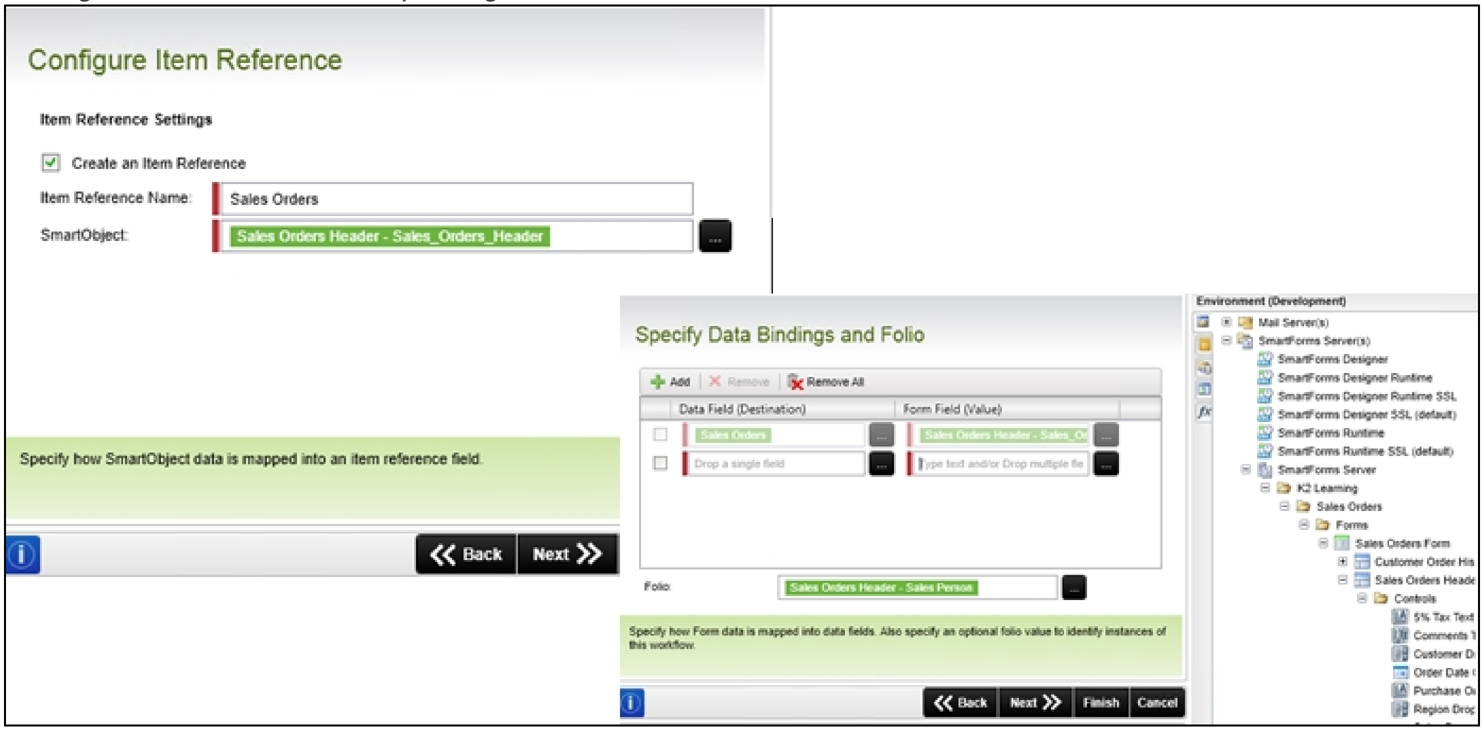
Select Rule: When Submit Button is Clicked

if the form passes validation
also on Sales Orders Header View, execute the Create method
also on Sales Orders Items View, execute the Create method for all the items that ha
if an advanced condition is true
then start the Sales Orders Process Workflow
if an advanced condition is true
then on Sales Orders Header SmartObject, execute its Save method

Configure the rule that is used to start the workflow. The rules associated with the selected state are updated in the Form definition once the workflow is deployed and can be further edited in the K2 Designer.

In subsequent screens of the wizard, you can set up an Item Reference and optionally pass additional data from the Form into the Workflow.

Adding the Item Reference and passing data from the Form to the Workflow



Configure Item Reference

Item Reference Settings

☑ Create an Item Reference

Item Reference Name:	Sales Orders
SmartObject:	Sales Orders Header - Sales_Orders_Header

Specify how SmartObject data is mapped into an item reference field.

Specify Data Bindings and Folio

Data Field (Destination)	Form Field (Value)
Sales Orders	Sales Orders Header - Sales_Or
Drop a single field	Type text and/or Drop multiple fie

Folio: Sales Orders Header - Sales Person

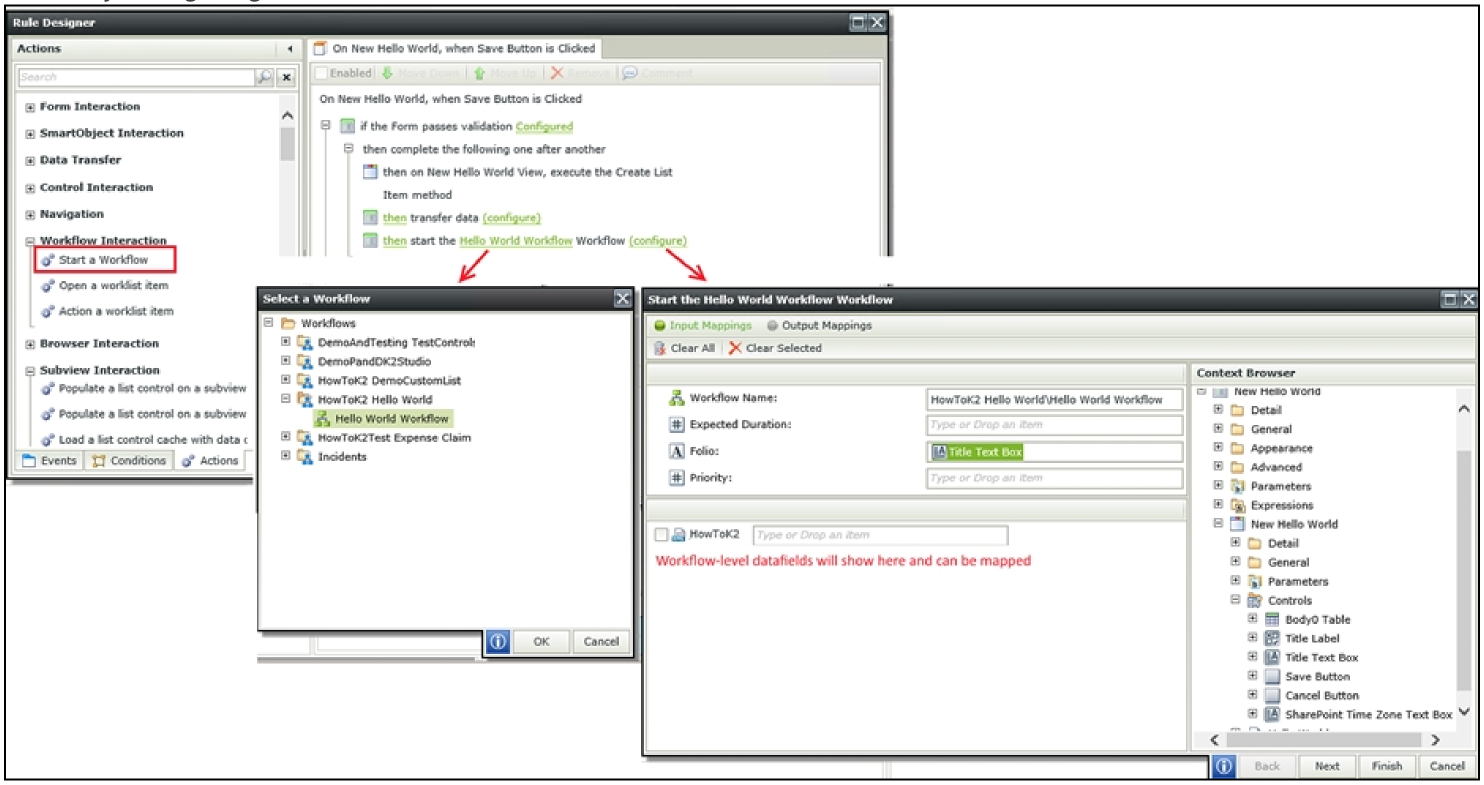
Specify how Form data is mapped into data fields. Also specify an optional folio value to identify instances of this workflow.

Manually adding the Start a Workflow Action

The other option is to manually set up the Form-Workflow integration by manually adding and configuring the **Start a Workflow** Action. With this approach, K2 will not inject or create additional integration (e.g. adding Item References) for you. When you deploy the Workflow, no changes are made to the Form because in essence, the Workflow does not know about the Form. With this approach you have to build and deploy the Workflow first, and then do the Form-Workflow Rule integration.

This is a more complex approach and requires additional work, but the Workflow and Form are less tightly bound together and you have full control over the integration. This approach may also require extra work when building a deployment package.

Manually configuring the Start Workflow Rule Action



On New Hello World, when Save Button is Clicked

if the Form passes validation Configured
then complete the following one after another
then on New Hello World View, execute the Create List Item method
then transfer data (configure)
then start the Hello World Workflow Workflow (configure)

Workflow Name: HowToK2 Hello World\Hello World Workflow
Expected Duration: Type or Drop an item
Folio: Title Text Box
Priority: Type or Drop an item

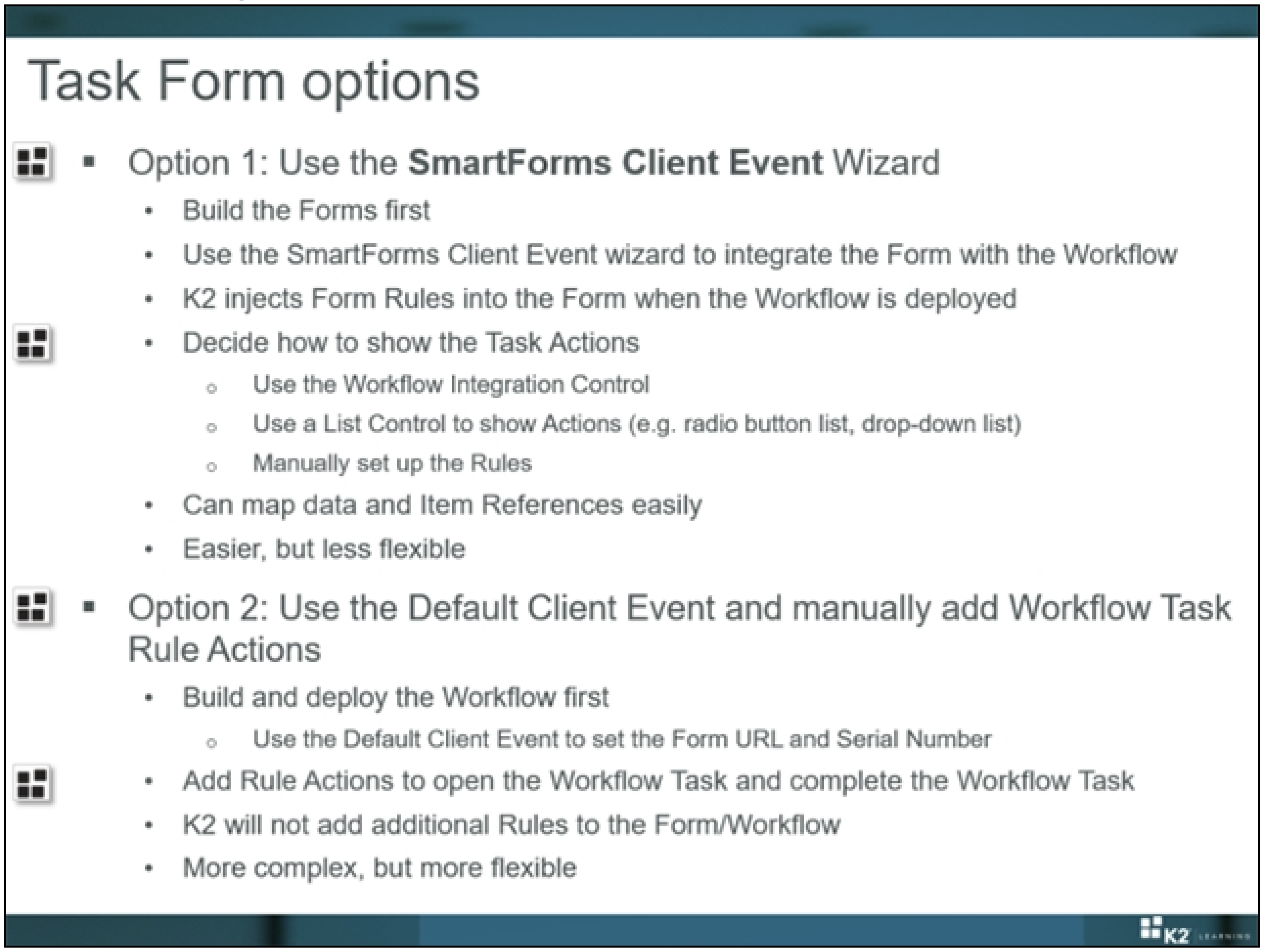
HowToK2 Type or Drop an item

Workflow-level datafields will show here and can be mapped

Summary

- There are two common approaches used to integrate Start Forms with a Workflow
- Use the SmartForms Integration Process Wizard (tightly coupled).
  - This will automatically add the Rules and mappings into the Form and set up the Item Reference for you.
  - The Form modifications will be applied to the Form when the Workflow is deployed.
  - With this approach you have to build the Forms first and then the Workflow.
  - It is the easier approach but you don’t have full control over the Form-Workflow integration.
- Manually configure the Workflow Start Rule Action using Rule Actions (loosely coupled).
  - You have to manually configure the Rule Action.
  - No Item Reference will be added to the Workflow, you will have to do so manually.
- The first approach is easier, the second approach is more flexible but requires more work.

Task Form options



Task Form options

- Option 1: Use the **SmartForms Client Event** Wizard
  - Build the Forms first
  - Use the SmartForms Client Event wizard to integrate the Form with the Workflow
  - K2 injects Form Rules into the Form when the Workflow is deployed
  - Decide how to show the Task Actions
    - Use the Workflow Integration Control
    - Use a List Control to show Actions (e.g. radio button list, drop-down list)
    - Manually set up the Rules
  - Can map data and Item References easily
  - Easier, but less flexible
- Option 2: Use the Default Client Event and manually add Workflow Task Rule Actions
  - Build and deploy the Workflow first
    - Use the Default Client Event to set the Form URL and Serial Number
  - Add Rule Actions to open the Workflow Task and complete the Workflow Task
  - K2 will not add additional Rules to the Form/Workflow
  - More complex, but more flexible

There are two common approaches used to integrate Forms with Workflows as Task Forms:

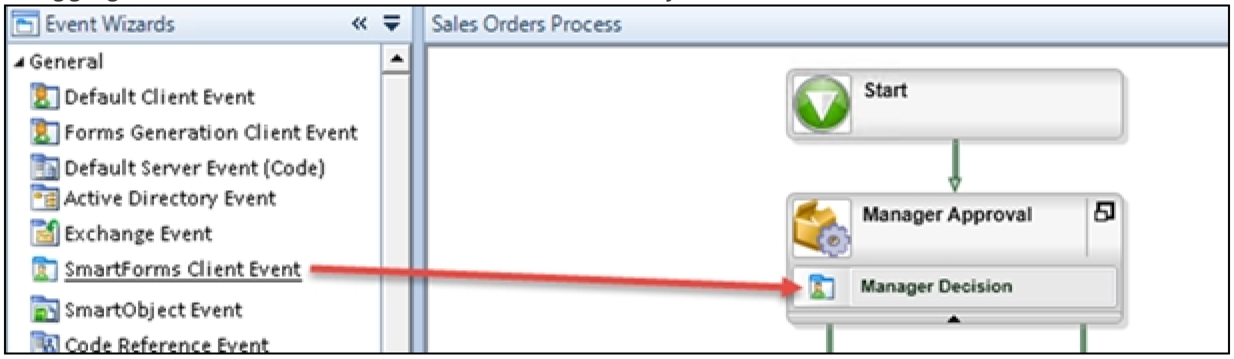
1. Use the **SmartForms Client Event** event wizard
2. Use the **Default Client Event** and manually add Workflow Task Rule Actions

Which approach you should use depends on your requirements: the first is easier because most of the integration work is done automatically for you. The second approach is more flexible but requires more manual configuration to set up. Let's look at the options in more detail.

Using the **SmartForms Client Event** event wizard

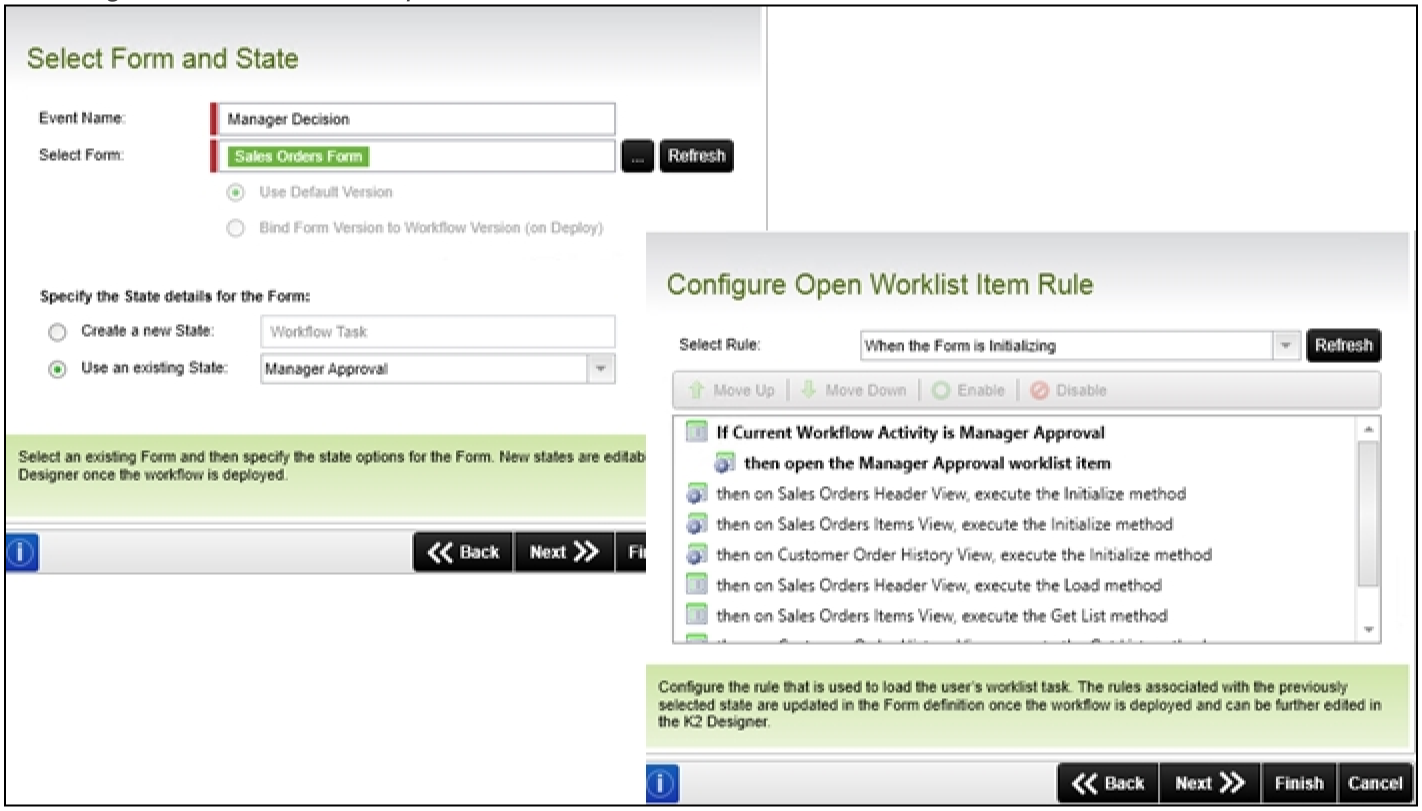
The easiest way to integrate a Task Form with a K2 Studio Workflow is to use the **SmartForms Client Event** wizard. This wizard will add the necessary Rule Actions to open and complete the Workflow Task and transfer data to and from the Form. The Form modifications will be applied to the Form when the Workflow is deployed. With this approach you have to build the Forms first and then the Workflow. It is the easier approach, but you don’t have full control over the Form-Workflow integration.

Dragging the SmartForms Client Event into an Activity



Once the wizard launches, you can select which Form and State to use for the Workflow Task integration. You can also select into which existing Rule the Action to open the task should be added.

Selecting the Form, State and Open Worklist Item Rule



Select Form and State

Event Name: Manager Decision

Select Form: Sales Orders Form

Use Default Version

Bind Form Version to Workflow Version (on Deploy)

Specify the State details for the Form:

Create a new State: Workflow Task

[✓ Use an existing State:] Manager Approval

Select an existing Form and then specify the state options for the Form. New states are editab Designer once the workflow is deployed.

Configure Open Worklist Item Rule

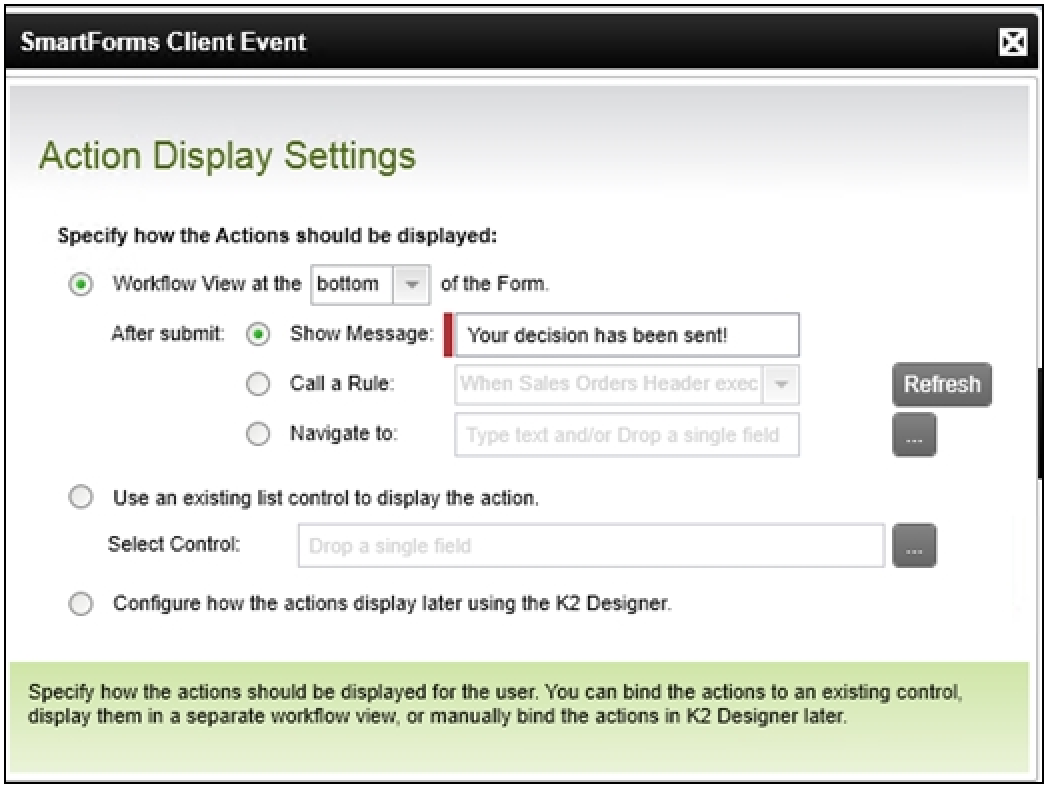
Select Rule: When the Form is Initializing

If Current Workflow Activity is Manager Approval
then open the Manager Approval worklist item
then on Sales Orders Header View, execute the Initialize method
then on Sales Orders Items View, execute the Initialize method
then on Customer Order History View, execute the Initialize method
then on Sales Orders Header View, execute the Load method
then on Sales Orders Items View, execute the Get List method

Configure the rule that is used to load the user's worklist task. The rules associated with the previously selected state are updated in the Form definition once the workflow is deployed and can be further edited in the K2 Designer.

As part of the wizard, you need to tell K2 how you want to expose the Actions to the end user. You have three options:

a. Workflow View: Let K2 automatically add a Workflow Task panel into the Form for you. You can select the placement of the task panel and what should happen after the Workflow Action is applied.
b. Use an existing list Control: K2 can populate a Control (e.g. a radio button list or drop-down list) with the available actions for the task.
c. Configure later: with this option, you need to set up the task Rule integration yourself.


SmartForms Client Event

Action Display Settings

Specify how the Actions should be displayed:

[✓ ●] Workflow View at the bottom of the Form.

After submit: [✓ ●] Show Message: Your decision has been sent!

○ Call a Rule: When Sales Orders Header exec

○ Navigate to: Type text and/or Drop a single field

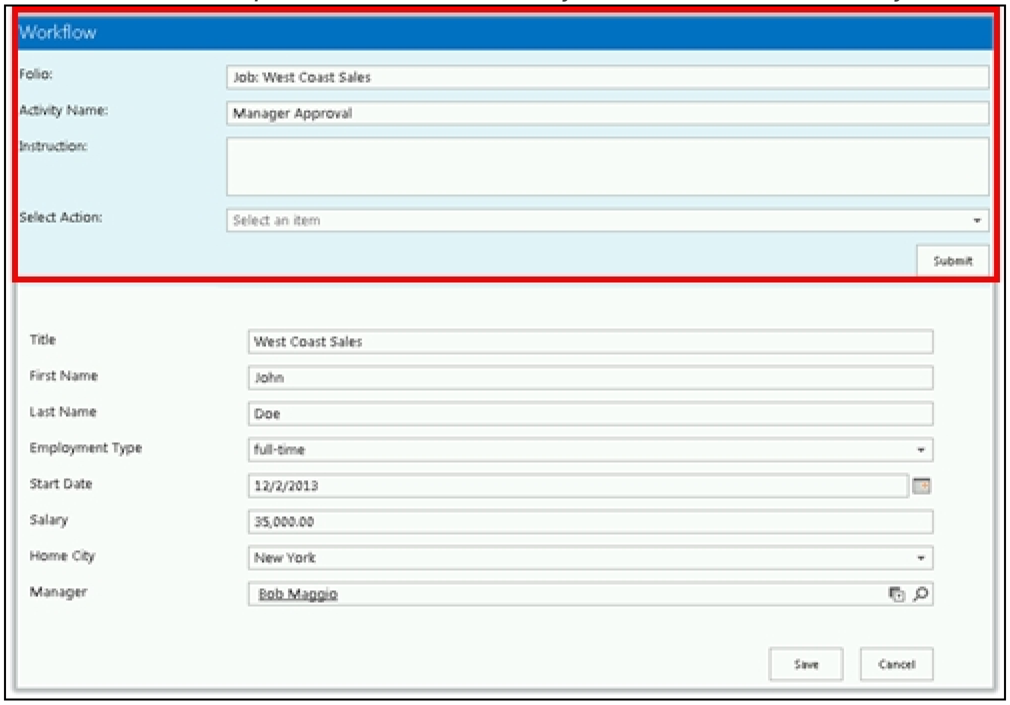
○ Use an existing list control to display the action.

Select Control: Drop a single field

○ Configure how the actions display later using the K2 Designer.

Specify how the actions should be displayed for the user. You can bind the actions to an existing control, display them in a separate workflow view, or manually bind the actions in K2 Designer later.

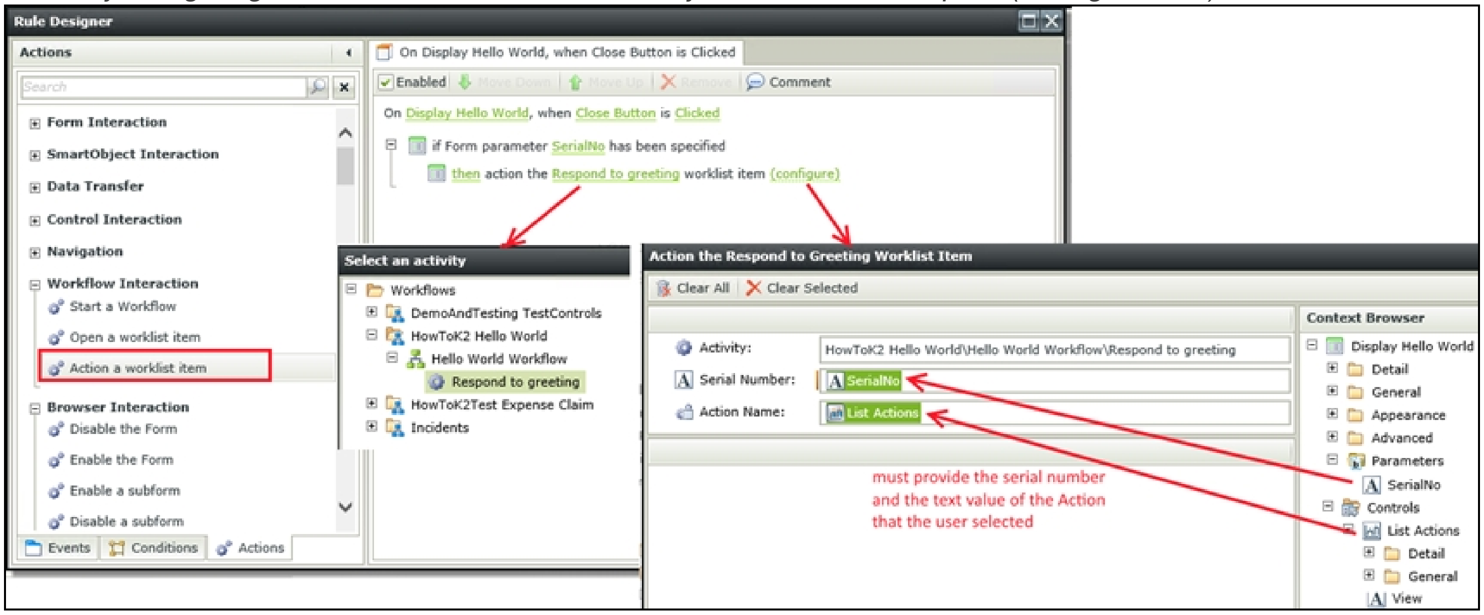
The Workflow Task panel that is automatically added to the Form when you select the first option (Workflow View)


Workflow

Folio:	Job: West Coast Sales
Activity Name:	Manager Approval
Instruction:	
Select Action:	Select an item

Title	West Coast Sales
First Name	John
Last Name	Doe
Employment Type	full-time
Start Date	12/2/2013
Salary	35,000.00
Home City	New York
Manager	Bob Maggio

Manually configuring the Workflow Task Actions when you select the third option (Configure later)



On Display Hello World, when Close Button is Clicked

if Form parameter SerialNo has been specified

then action the Respond to greeting worklist item (configure)

Activity: HowToK2 Hello World\Hello World Workflow\Respond to greeting

Serial Number: SerialNo

Action Name: List Actions

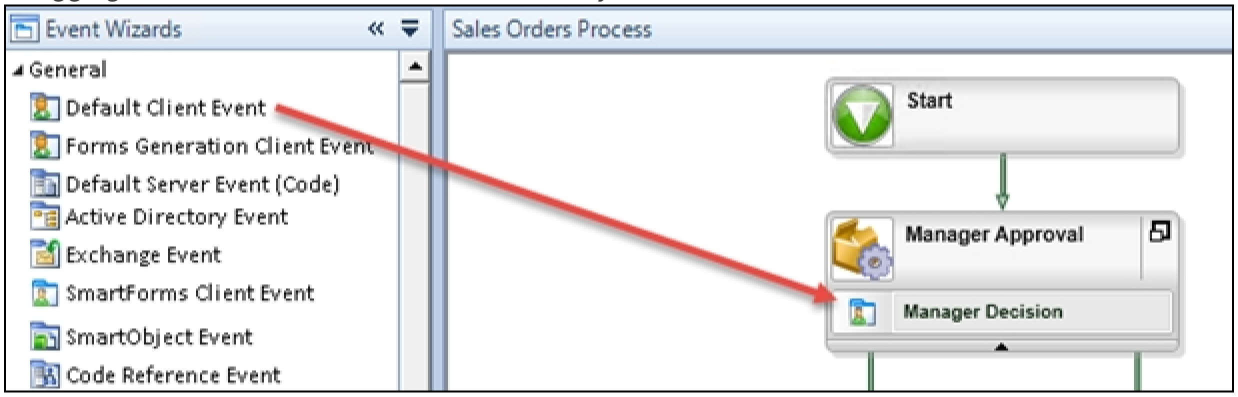
must provide the serial number and the text value of the Action that the user selected

Using the **Default Client Event** and manually add Workflow Task Rule Actions

Another approach is to use the **Default Client Event** wizard. This wizard will not add any Actions to the Form, and you will need to set up the Task Form-Workflow integration manually. With this approach you have to build and deploy the Workflow first and then the Forms, then go back to the Workflow to set the Form URL in the Client Event. This is a more complex approach and requires manual configuration, but you have full control over the Form-Workflow integration and the Form and Workflow are less tightly bound together.

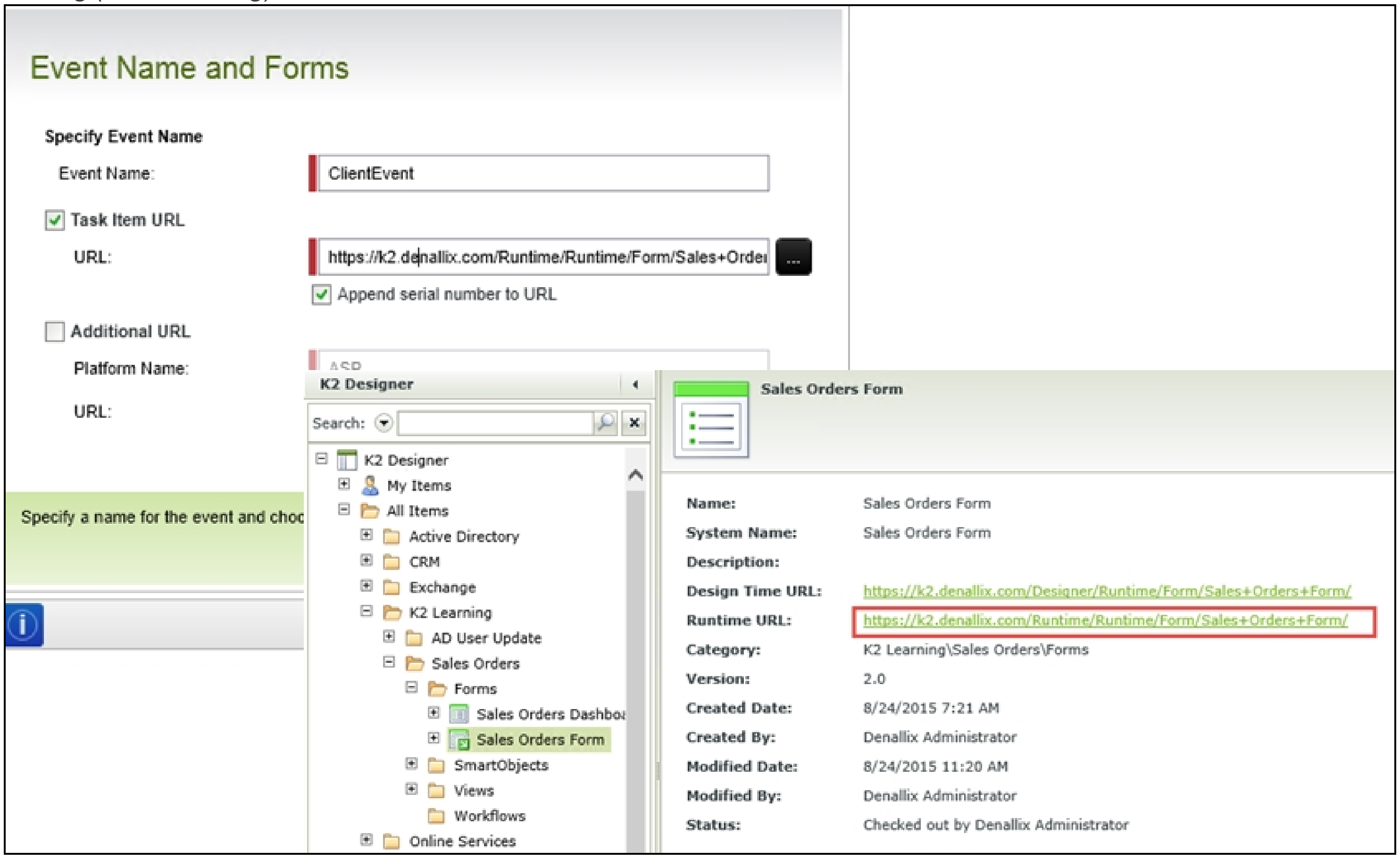
Tip
This is also an option to consider when you want to use a completely different Form for the same Client Event, because you can set the Form URL using values from the Context Browser, as opposed to targeting one specific Form.

Dragging the Default Client Event into an Activity



When you configure the Default Client Event, you have to provide the URL to the Form that will be used to complete that task. You can obtain the Form's Runtime URL from the Form Properties screen, as shown below. Remember that you have to include the Task Serial Number (usually as a query string parameter) because the Rule Actions will require the Serial Number when opening and completing the task.

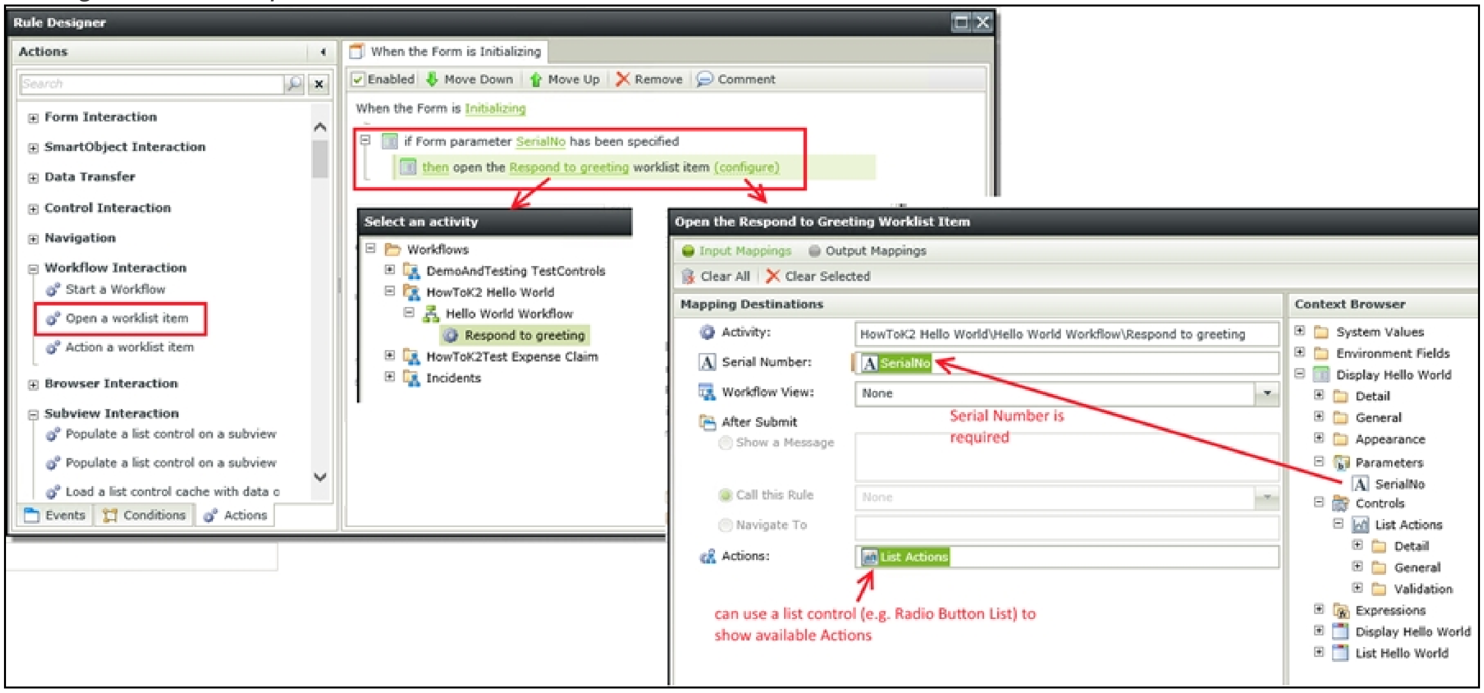
Setting (and obtaining) the URL for the Default Client Event



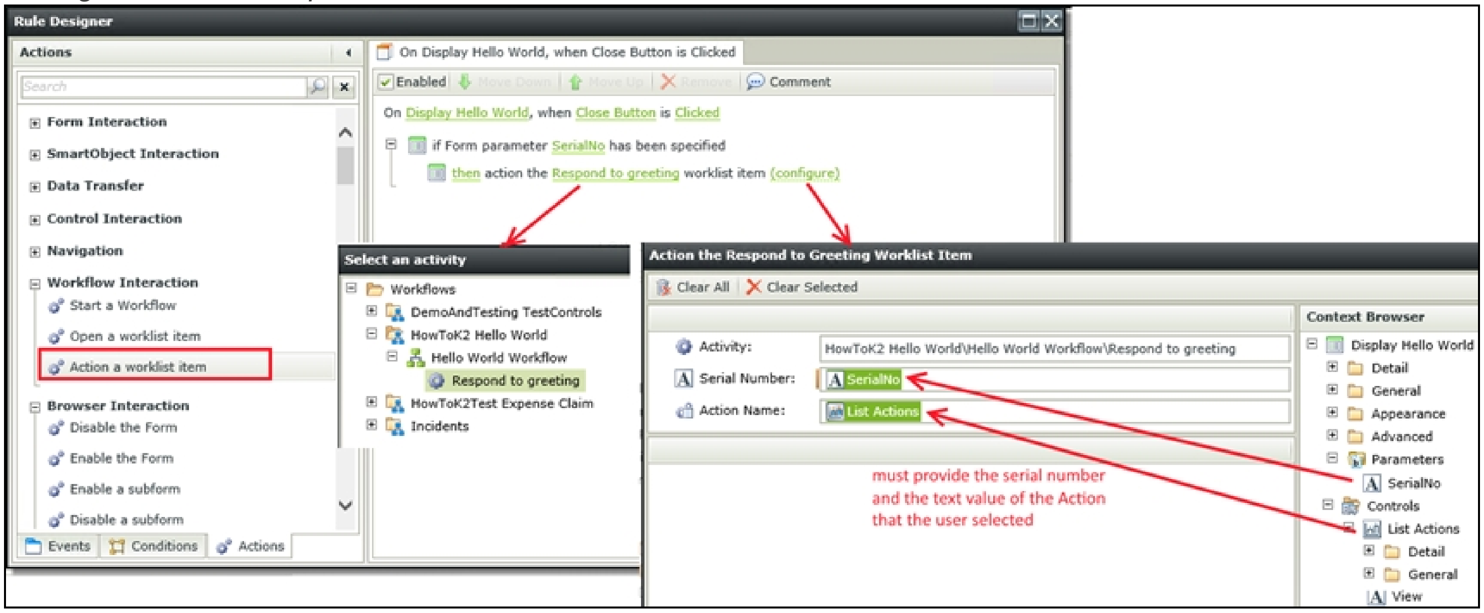
Name:	Sales Orders Form
System Name:	Sales Orders Form
Description:	
Design Time URL:	https://k2.denallix.com/Designer/Runtime/Form/Sales+Orders+Form/
Runtime URL:	https://k2.denallix.com/Runtime/Runtime/Form/Sales+Orders+Form/
Category:	K2 Learning\Sales Orders\Forms
Version:	2.0
Created Date:	8/24/2015 7:21 AM
Created By:	Denallix Administrator
Modified Date:	8/24/2015 11:20 AM
Modified By:	Denallix Administrator
Status:	Checked out by Denallix Administrator

Once the Workflow is deployed, you need to set up the Rule Actions to open and complete the Workflow Task manu-ally.

Adding an Action to open the Workflow Task



Adding an Action to complete the Workflow Task



On Display Hello World, when Close Button is Clicked

On Display Hello World, when Close Button is Clicked
if Form parameter SerialNo has been specified
then action the Respond to greeting worklist item (configure)

Action the Respond to Greeting Worklist Item

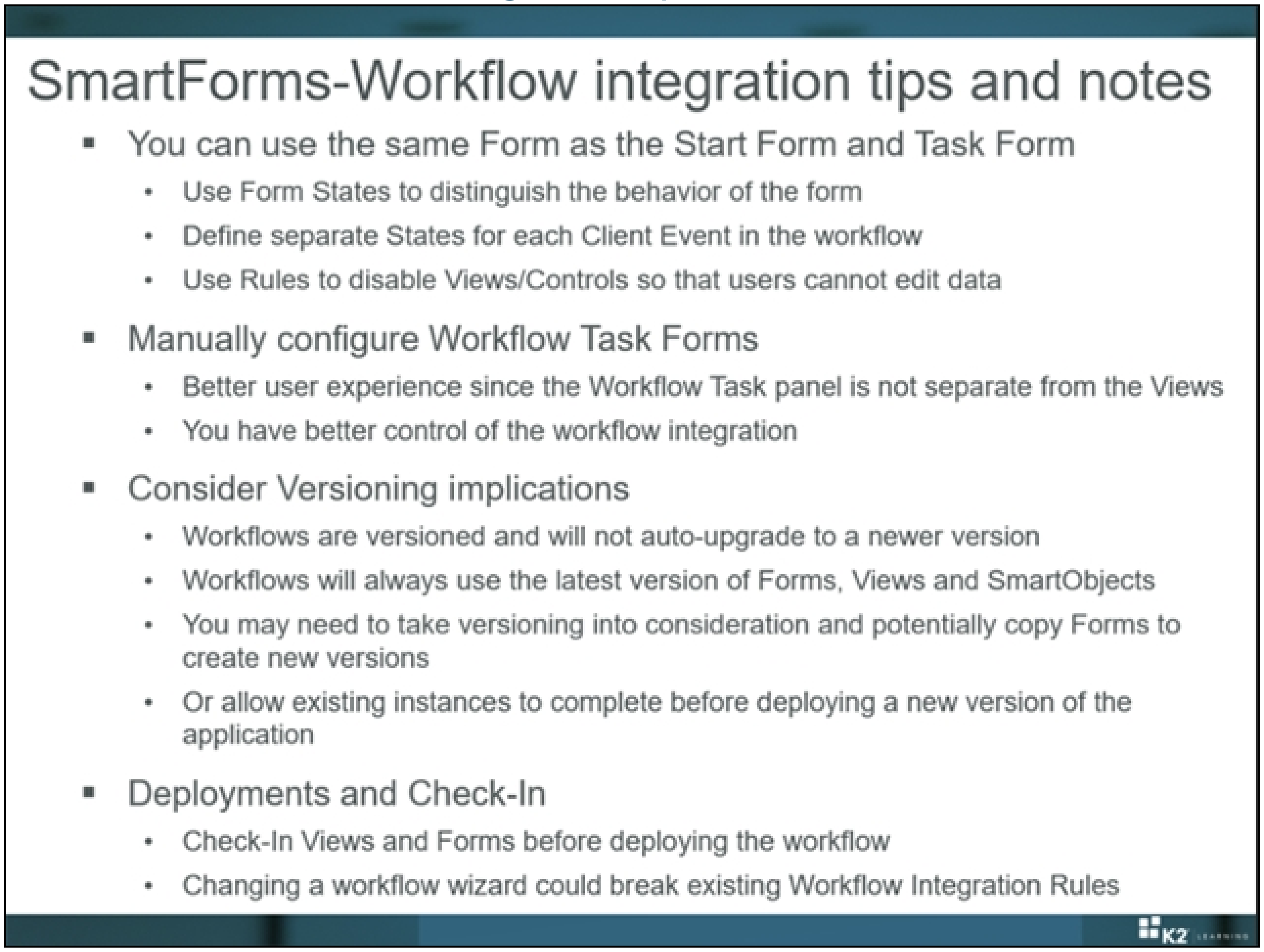
Activity: HowToK2 Hello World\Hello World Workflow\Respond to greeting
Serial Number: SerialNo
Action Name: List Actions

must provide the serial number and the text value of the Action that the user selected

Summary

- You have two options when integrating Task Forms with Client Events:
  - Use the **SmartForms Client Event** wizard (tightly coupled)
    - Build the Forms first
    - Use the **SmartForms Client Event** wizard to integrate the Form with the Workflow
    - K2 injects Form Rules into the Form when the Workflow is deployed
    - Rules to open the Workflow Task
    - Rules to complete/action the Workflow Task
    - This approach is easier, but less flexible because the Workflow and Forms become tightly bound together, and the Workflow makes changes to the Form when it is deployed.
  - Use the **Default Client Event** and manually add Workflow Task Rule Actions (loosely coupled)
    - With this approach you need to set up all the SmartForms-Workflow integration manually
    - Build and deploy the Workflow first
    - Use the Default Client Event to set the Form URL and Serial Number. You can obtain the Form URL from the Form Properties page.
    - Add Rule Actions to open the Workflow Task and complete the Workflow Task
    - K2 will not add additional Rules to the Form/Workflow, you need to set up all the integration and data transfer actions yourself.
    - This approach is more complex to set up, but more flexible because you have full control over the Workflow integration.
- The first approach is easier, the second approach is more flexible but requires more work to set up.

SmartForms-Workflow integration tips and notes



SmartForms-Workflow integration tips and notes

- You can use the same Form as the Start Form and Task Form
  - Use Form States to distinguish the behavior of the form
  - Define separate States for each Client Event in the workflow
  - Use Rules to disable Views/Controls so that users cannot edit data
- Manually configure Workflow Task Forms
  - Better user experience since the Workflow Task panel is not separate from the Views
  - You have better control of the workflow integration
- Consider Versioning implications
  - Workflows are versioned and will not auto-upgrade to a newer version
  - Workflows will always use the latest version of Forms, Views and SmartObjects
  - You may need to take versioning into consideration and potentially copy Forms to create new versions
  - Or allow existing instances to complete before deploying a new version of the application
- Deployments and Check-In
  - Check-In Views and Forms before deploying the workflow
  - Changing a workflow wizard could break existing Workflow Integration Rules

Video

When considering how SmartForms integrate with Workflows, there are a number of tips and notes to bear in mind.

You can use the same Form as the Start Form and Task Form

Remember that you can use Form States to distinguish the behavior of the Form and define Rule Actions accordingly. If you have a complex Form you may want to use the same Form for all tasks since it can reduce the number of separate Forms you need to maintain. It is also a good idea to define separate States for each Client Event in the Workflow so that you can control the Form's behavior depending on where in the Workflow it is being used. Note that too many States can negatively impact the performance of a Form. Remember also, you can use Rules to disable Views/Controls so that users cannot edit data.

Manually configure Workflow Task Forms

If users need to edit data on the Forms, do not use the Workflow Task panel but instead, manually configure Workflow integration Actions and embed them into the Form/View Rules. This results in a better user experience because the Workflow Task panel is not separate from the Views, and you can take multiple Actions (e.g. update the data and complete the workflow Action) on the same event (such as when the user clicks the Submit button). While this approach does take a little more work to build, you have better control of the Workflow integration and can put Controls where you want them, as opposed to K2 adding on a Workflow Task panel to your Form.

Consider Versioning implications

It is important to keep in mind the effect of Workflow versioning when you integrate Forms with Workflows. Remember that Workflow Instances are bound to specific versions of the Workflow definition, and will not auto-upgrade to a newer version if you deploy a new version of the Workflow. Workflow Instances are bound to the version of the Workflow they were started on. (It is possible to migrate Workflow versions using Live Instance Management APIs, but this is not recommended as a life cycle management approach.)

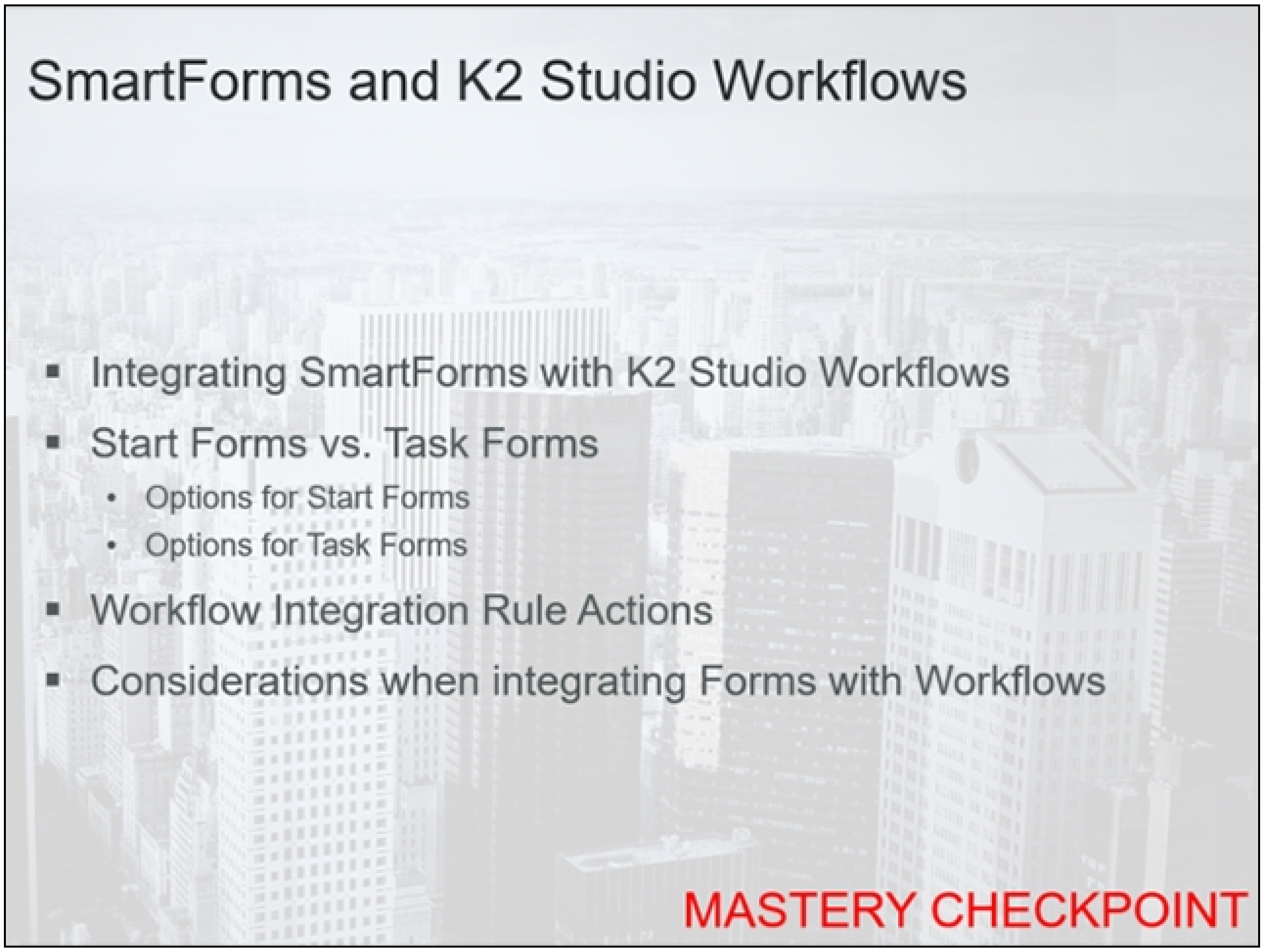
When it comes to Forms, Views and SmartObjects, all of these artifacts will use the latest deployed version, so it is quite possible that an older-version Workflow Instance might try to run against a newer version of a Form or SmartObject. (Workflows will always use the latest version of Forms, Views and SmartObjects.)

You may need to take versioning into consideration and potentially copy Forms to create new versions (i.e. Leave Request V1 joined to Workflow V1, Leave Request V2 joined to Workflow V2), or allow existing instances to complete before deploying a new version of the application so that existing Workflows are not broken by new SmartObjects or new Forms/Views.

Deployments and check-in

Check-in Views and Forms before deploying the Workflow, so that there are no surprises when K2 adds Rules to your Views and Forms. (Remember that K2 may add Rules and Actions to Forms and Views depending on how you integrated the Form and Workflow.) Changing a Workflow wizard could impact existing Workflow integration Rules (e.g. deleting a Client Event and recreating it). Try to avoid this as much as possible.

MASTERY CHECKPOINT: SmartForms and K2 Studio Workflows



SmartForms and K2 Studio Workflows

- Integrating SmartForms with K2 Studio Workflows
- Start Forms vs. Task Forms
  - Options for Start Forms
  - Options for Task Forms
- Workflow Integration Rule Actions
- Considerations when integrating Forms with Workflows

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 4 of this module. If you are attending instructor-led training, use this as an opportunity to answer any questions around integrating Forms with Workflows built in K2 Studio or K2 for Visual Studio.

After completing Part 4, you should know:

- How to integrate SmartForms with Workflows built in K2 Studio or K2 for Visual Studio
- The difference between Start Forms and Task Forms
- Start Form options
  - Using the Process wizard
  - Manually configuring the Workflow Start Action
- Task Form options
  - Using the SmartForms Client Event wizard
  - How the Workflow actions are displayed depending on the choices you make in the wizard
  - Using the Default Client Event wizard
- How to manually set up Workflow task integration to open and action the task
- Some considerations and tips when integrating Forms with Workflows

Knowledge-check questions

Q: Can you use the same Form as the Start Form and as a Task Form in a Workflow?
Reveal answer**A:** Yes, you would normally use States to achieve this.

Q: What is the difference between the 'Open a worklist item' action and the 'Action a worklist item' action?
Reveal answer**A:** Open a worklist item – is used to open the task, usually when the Form is opened by the assigned task user. Action a worklist item – used to complete the task when the user is ready to submit the form.

Q: True or False: Workflow Instances always run on the latest definition that was published to the server.
Reveal answer**A:** False. Workflows start on the Default version of the Workflow and are NOT automatically upgraded when a new version is deployed.

Q: True or False: Workflows always use the latest checked-in version of a Form.
Reveal answer**A:** True (well, mostly true: the exception is that users who currently have the Form checked out will always see the version they have checked out, while other users won't see the changes until the Form is checked in).

Q: Would you prefer to manually integrate SmartForms with Workflows, or use the wizards to do the integration? Why?
A: (discussion question)

PART 5: SmartForms-based Reports

PART 5

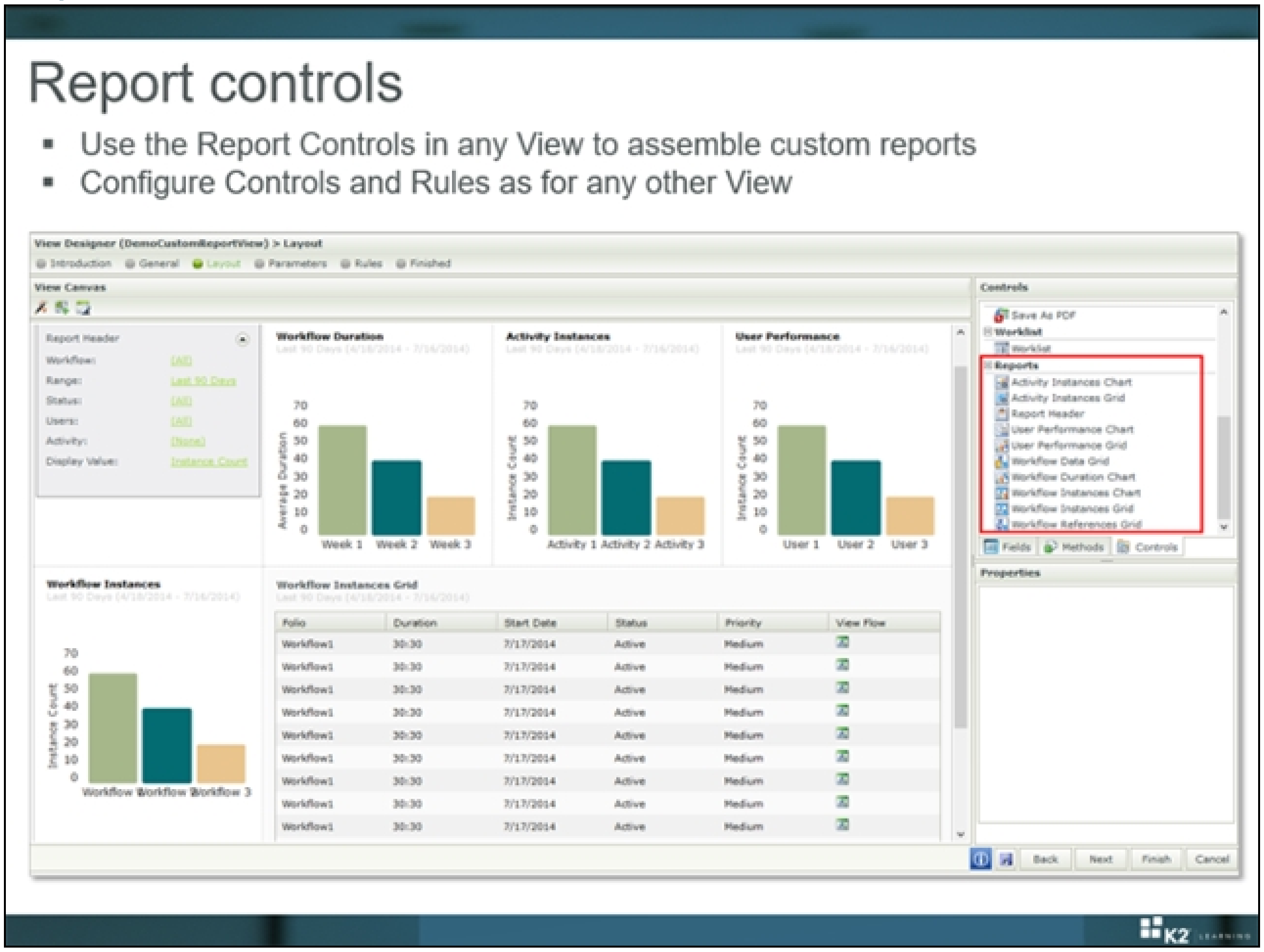
SMARTFORMS-BASED REPORTS

✓ Reporting controls in SmartForms

Video

In Part 5 we will look at using the available reporting Controls in SmartForms to build SmartForm-based reports. At the end of Part 5, we will create a dashboard-style report for the Sales Orders application.

Report Controls



Report controls

- Use the Report Controls in any View to assemble custom reports
- Configure Controls and Rules as for any other View

View Designer (DemoCustomReportView) > Layout

Introduction · General · Layout · Parameters · Rules · Finished

View Canvas

Report Header

Workflow:	(All)
Range:	Last 90 Days
Status:	(All)
Users:	(All)
Activity:	(None)
Display Value:	Instance Count

Workflow Instances Grid

Last 90 Days (4/18/2014 - 7/16/2014)

Folio	Duration	Start Date	Status	Priority	View Flow
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	
Workflow1	30:30	7/17/2014	Active	Medium	

Controls

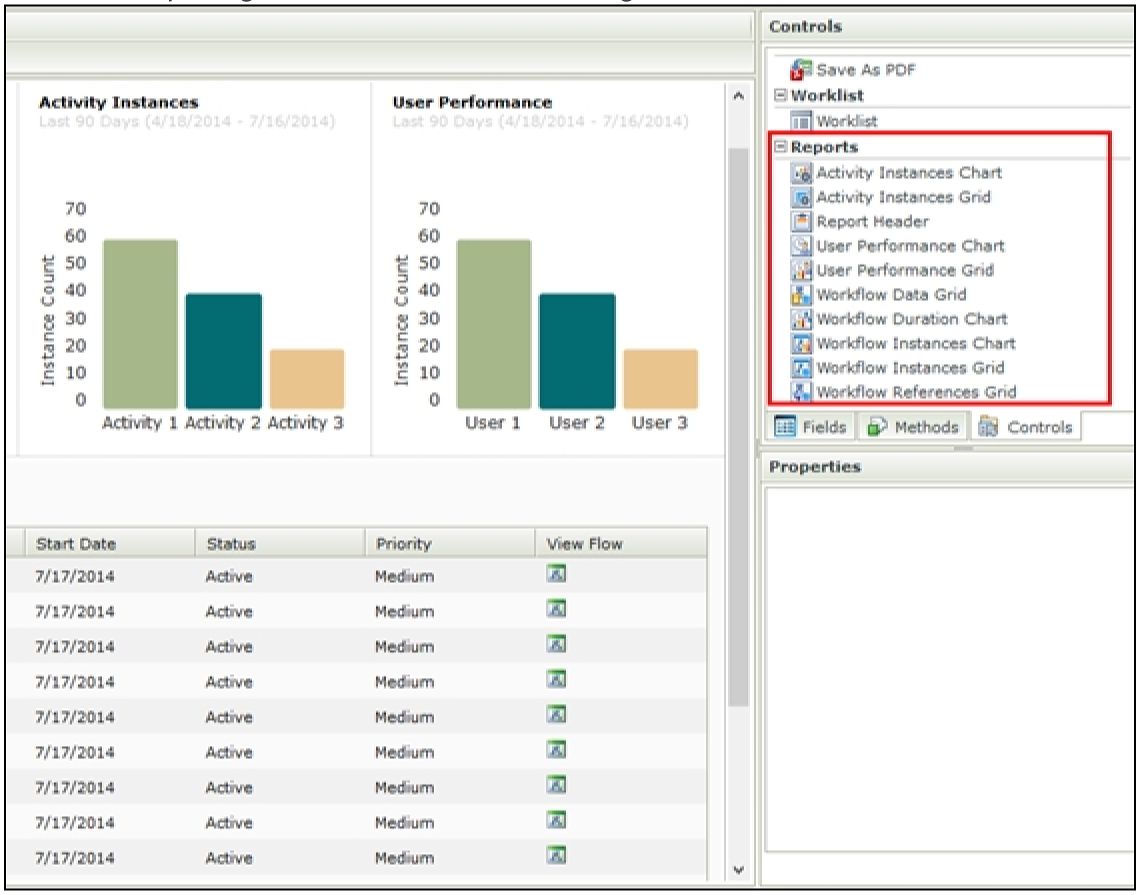
- Save As PDF
- Worklist
  - Worklist
- Reports
  - Activity Instances Chart
  - Activity Instances Grid
  - Report Header
  - User Performance Chart
  - User Performance Grid
  - Workflow Data Grid
  - Workflow Duration Chart
  - Workflow Instances Chart
  - Workflow Instances Grid
  - Workflow References Grid

You can create custom reports by incorporating reporting Controls with other Controls and Views to assemble custom report SmartForms. You can then add these Forms to SharePoint using the K2 Forms Viewer App Part, or just by exposing the Forms through their Form (Runtime) URL.

Reporting controls are found in the same location as the other View and Form Controls. Under the Reports heading, you will find Controls for adding spreadsheet-formatted lists, along with charts and graphs. Simply drag and drop the report Control you need onto the View canvas, then configure it using the options in the Properties Pane.

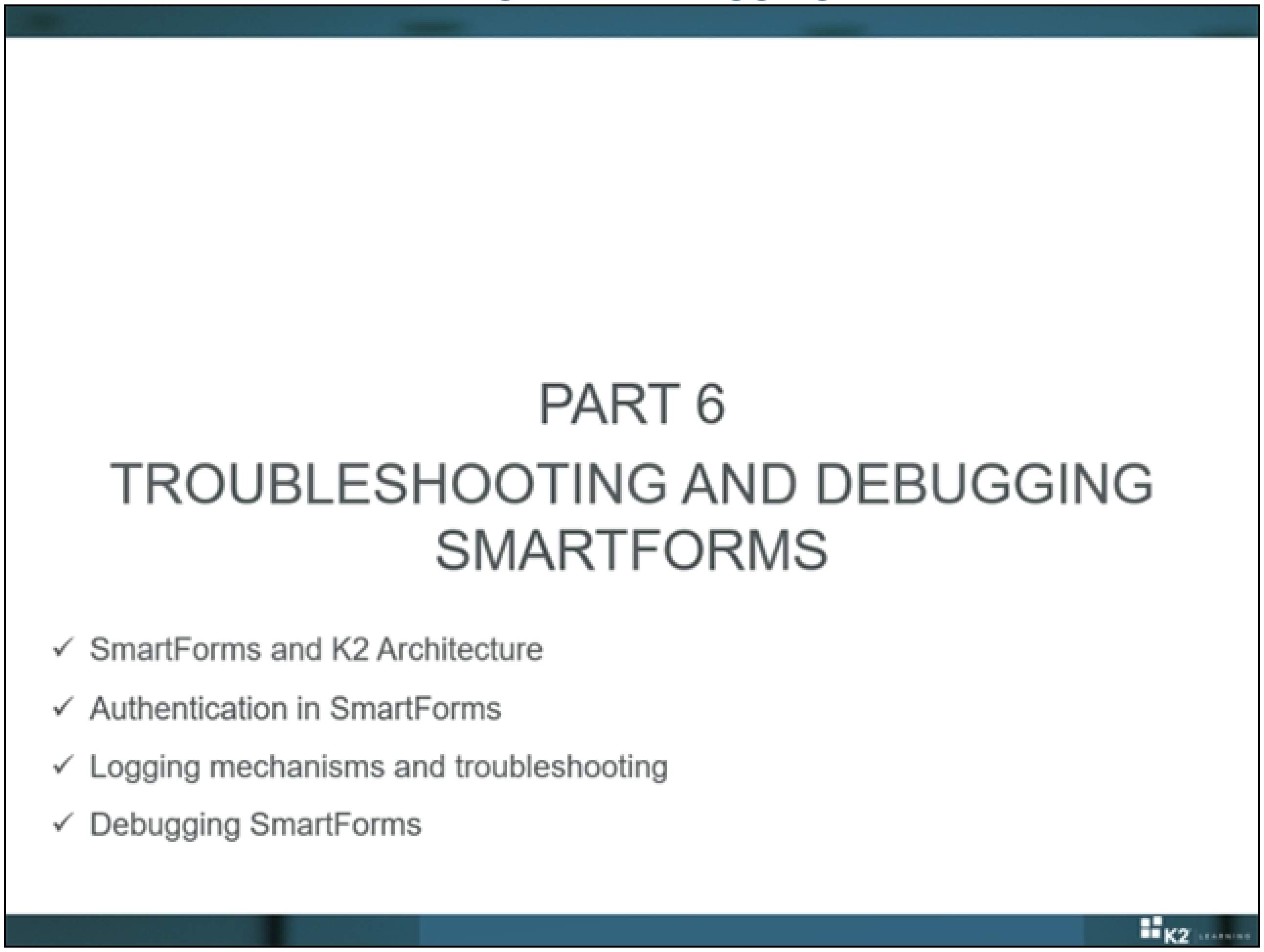
The benefit of creating a custom report Form is that you can assemble dashboard-style reports that combine Workflow reporting data with SmartObject business data. At the time of writing (August 2015), the reporting Controls only support workflow data, but check your version of K2 since additional reporting Controls may have been added in the meantime.

Workflow Reporting Controls in the K2 View Designer



Start Date	Status	Priority	View Flow
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	
7/17/2014	Active	Medium	

PART 6: Troubleshooting and Debugging SmartForms



PART 6

TROUBLESHOOTING AND DEBUGGING SMARTFORMS

- ✓ SmartForms and K2 Architecture
- ✓ Authentication in SmartForms
- ✓ Logging mechanisms and troubleshooting
- ✓ Debugging SmartForms

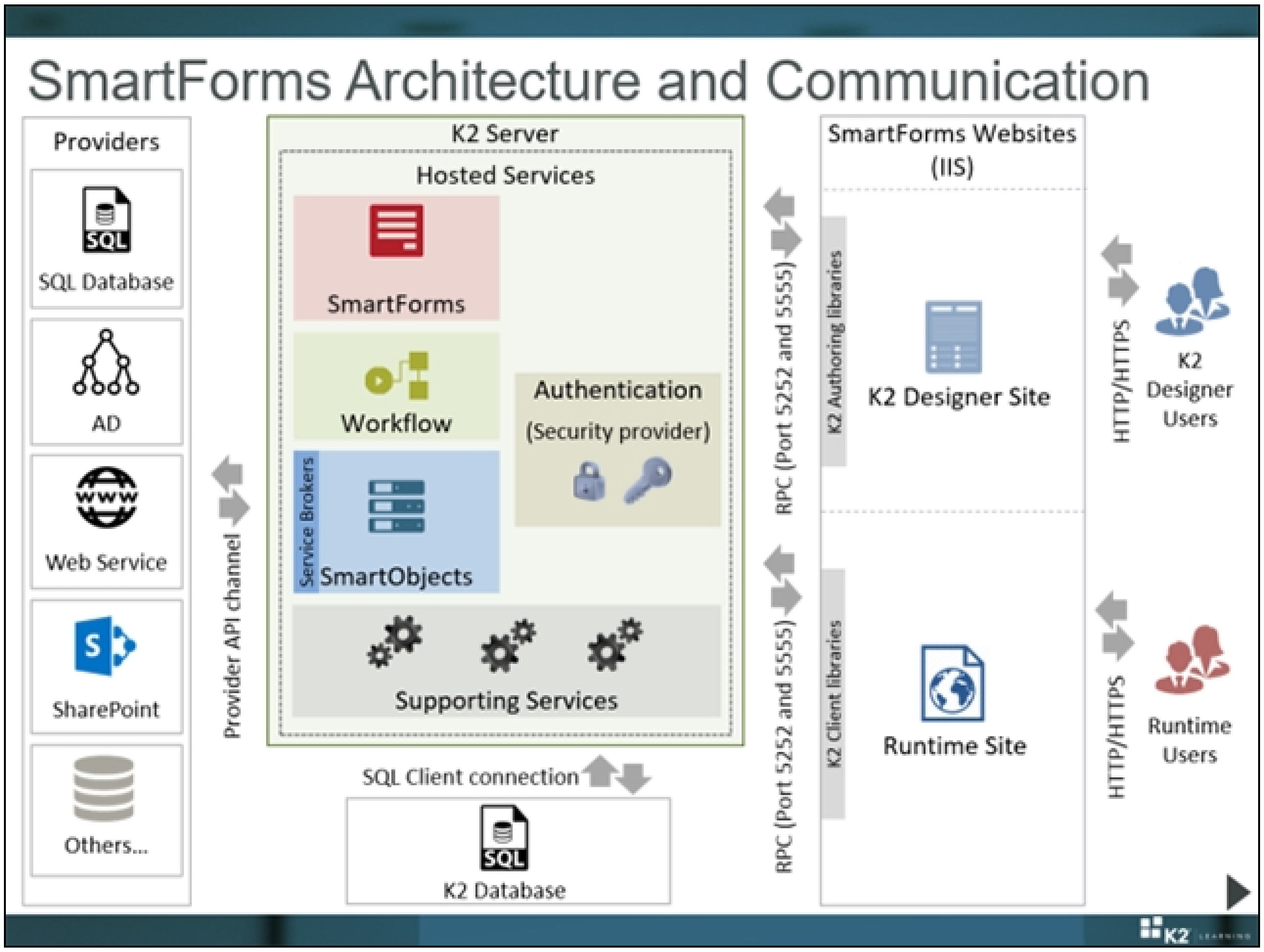
Video

In Part 6 we will look at how to troubleshoot and debug SmartForms, including the available logging options that you can use to troubleshoot SmartForms. We will also be looking at the architecture and authentication of SmartForms in more detail, since it is important to understand the architecture to troubleshoot effectively.

At the end of Part 6, we will do a simple exercise to debug a SmartForm.

Note
Given the subject matter, this section of the training course will be more technical in nature. If you are not familiar with web authentication or architecture of web applications, this might be challenging content, but knowing the basics of troubleshooting will help you build SmartForms.

SmartForms Architecture and Communication



Video

It is important to understand the architecture and communication of SmartForms because when you need to troubleshoot, it helps to know how communication flows and what the components are. The diagram in the slide for this topic represents the logical communication architecture of SmartForms.

There are two websites that are installed along with SmartForms. End users mostly interact with these sites through web browsers on their machines or devices, using standard web-based HTTP or HTTPS protocols and methods like HTTP gets, jQuery and AJAX calls. (The specific ports in use will depend on the environment and IIS configuration, but normally HTTP is over port 80 and HTTPS is over port 443.)

- K2 Designer Site: this is where users build SmartForms, SmartObjects and Workflows with the K2 Designer tool
- Runtime Site: this is where end users will run Forms at runtime

You can create multiple copies of the runtime site in different IIS websites to serve different users, as we will explore further in the next topic.

The SmartForms websites communicate with K2 using the available K2 authoring and client libraries, for example, the workflow client library (.dll), the SmartForms authoring library (.dll) and other libraries. These libraries communicate with the K2 Servers using Remote Procedure Calls (RPC). By default, the ports used are 5252 for workflow client communication and 5555 for all other communication, although customers may have changed their port numbers during installation.

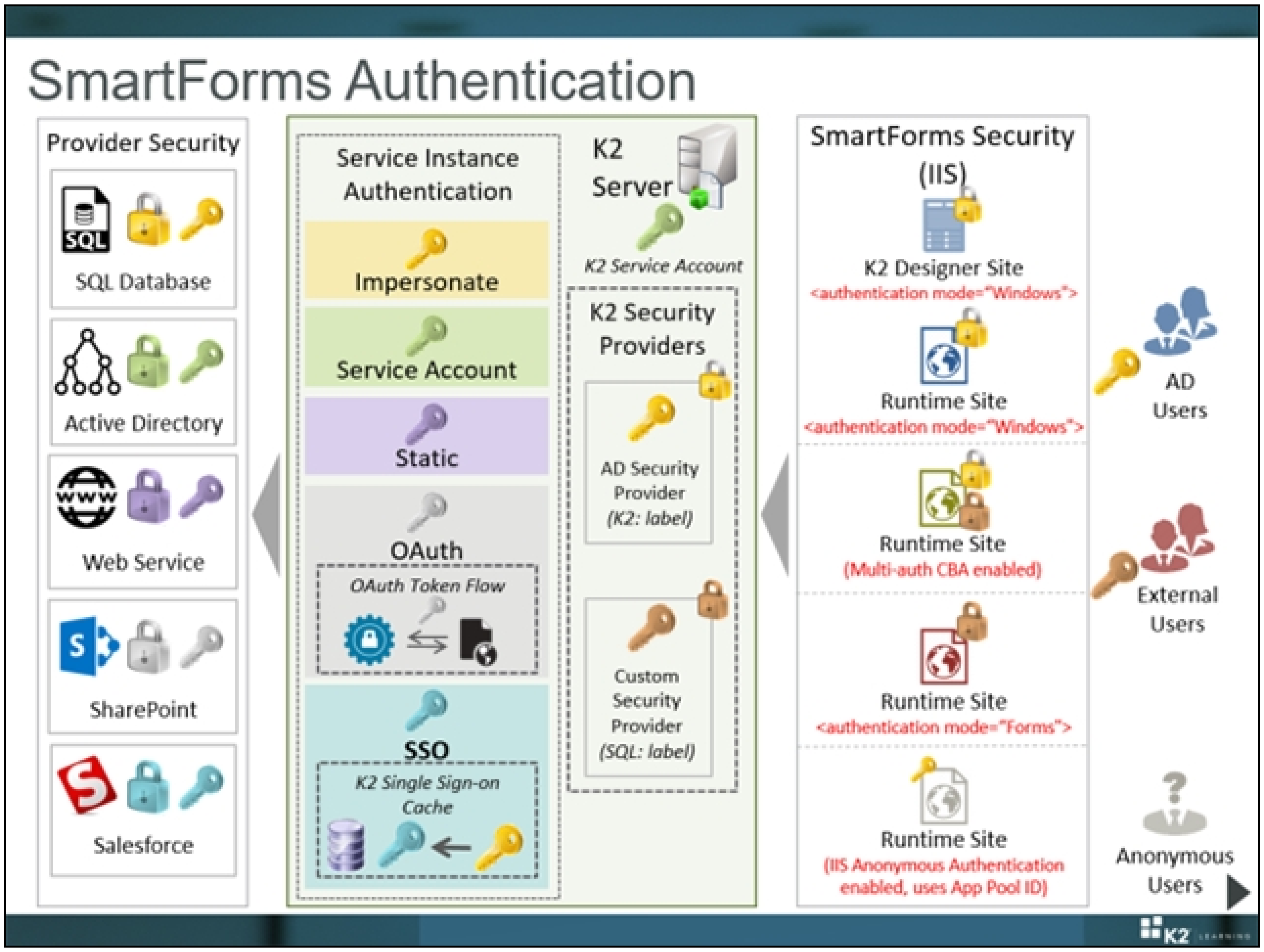
K2 internally has a number of hosted services (such as SmartForms, Workflow and SmartObjects) along with supporting services like authentication, logging and others. Once the libraries connect to K2, the subsequent processing happens within these hosted services inside the K2 Host Server process. As part of this processing, K2 will communicate with its own database (Microsoft SQL server) using standard .NET SQL client calls (normally port 1433).

When K2 communicates with other providers via Service Brokers, the communication channels depend on that provider: sometimes it might be HTTP calls and sometimes native client calls (it depends on what communication channel the provider API uses).

Summary

- There are two websites that are installed along with SmartForms:
  - K2 Designer: this is where users build SmartForms, SmartObjects and Workflows with the K2 Designer tool
  - Runtime site: this is where end users run Forms (runtime)
- The SmartForms websites communicate with K2 using the available K2 authoring and client libraries
  - These libraries communicate with the K2 Servers using Remote Procedure Calls (RPC) over ports 5252 and 5555
- K2 has a number of internal services that perform the majority of the processing work
- When K2 communicates with other providers via Service Brokers, the communication channels depend on that provider

SmartForms Authentication



Video

K2 smartforms are built to leverage the existing authentication mechanisms ("Security Providers") available in K2. By default, K2 smartforms are installed and configured to work seamlessly with Active Directory authentication. If you are only using Active Directory for authentication in your SmartForms/K2 environment, you do not need to spend too much time delving into the details around authentication in K2 smartforms. Normally, a K2 and SmartForms installation in an Active Directory-only environment is easy to set up and does not usually require any additional configuration.

The key concept in K2 security is that of a *Security Label*. A Security Label is effectively an instruction that tells K2 which security provider to use when authenticating a user. Active Directory is always identified with the *K2:* security label. When a user connects to K2 with the K2: security label, K2 will authenticate that user against Active Directory.

Consider the diagram in the slide that accompanies this topic, and specifically look at the Security Providers section under K2 Server Security. In this example, we have two security providers: the standard *K2:* Active Directory security provider, and a custom security provider with a *SQL:* security provider label. When users connect to K2, K2 will look at the security label prepended to the username so that it knows which security provider to use when authenticating a user.

Now, consider the SmartForms Security section of the diagram. Here, we have two different approaches: Windows Authentication (blue sites) and Forms Authentication (maroon sites). The authentication setting is stored in the web.-config file for each of the websites that make up a SmartForms installation: the K2 Designer website and the Runtime website. Let’s assume you have only Active Directory users. In this case, the websites are automatically configured with the Windows Authentication setting, and the websites will automatically take the current user’s Windows credentials and pass them through to K2 with the K2: security label. K2, in turn, authenticates the user against Active Directory because of the K2: label in the username. All of this happens transparently and the user does not need to provide their login credentials: K2 will automatically use the connected user’s credentials.

In a more complex environment, you may need to expose SmartForms to external users as well. In this case, there are two important pieces to the puzzle: first, you need a security provider that can authenticate the external users. The second part is to configure the Designer and/or Runtime websites to use the Forms Authentication mechanism. When you use the Forms Authentication, users will need to log into the website before they will gain access to the Runtime or Designer websites.

Remember that there are two separate websites used in a SmartForms implementation: the Designer website (which hosts the K2 Designer components) and the Runtime website (which is used by end-users at runtime to interact with Forms). You are free to change the security settings for each website independently or create copies of each website and configure them differently for different audiences. Remember that the Forms themselves are served up by the K2 Server, so end users would see the same Form regardless of which website they have used to connect to K2.

In a typical internal/external-user environment, you would normally have a single installation of the Designer website, configured for Windows Authentication and used exclusively by internal users to design Forms and Workflows. For the end-user interaction, you would normally create two installations of the Runtime website: one site would be configured for Windows Authentication (so that internal users can interact with Forms without providing additional log in credentials) while the external website would be configured for Forms Authentication. External users would log into the Forms Authenticated website with a specific set of credentials and a specific security label. Ultimately, both websites look at the same collection of Forms, so it is not necessary to implement separate copies of the same Form for the different users. Once you have changed the authentication mode to Forms, end-users will need to log in with their username, password and a security label before they will be able to access any Forms.

When the user clicks a sign-in button, SmartForms will combine these security credentials and then pass them through to the K2 Server for authentication. Depending on the security label used to log in, K2 will use the appropriate security provider to authenticate the user. This means that when you change a website to use Forms Authentication, both external and Active Directory users can use the same website. The Active Directory users will use the K2: label and log in with their normal Active Directory username and password, while the external users would use the label of the custom security provider.

Multi-Auth

You may want to use a single runtime site for all users, regardless of how they are authenticated. SmartForms 1.0.6 introduced a concept called Multi-Auth, where you can configure a single SmartForm runtime site to allow multiple different authentication mechanisms. With this setup, as long as the necessary authentication providers and claims-based authentication setup has been configured, you can use a single website to authenticate users from different authentication mechanisms (although you will still require separate authentication providers for each authentication mechanism). See http://help.k2.com/onlinehelp/K2smartforms/ICG/current/default.htm#IntroductiontoMulti-Auth/K2MultiAuthIntroduction.htm for more information on using Multi-Auth.

Anonymous access

You may want to expose Forms to non-authenticated users (for example a public survey application). You can set up a SmartForms runtime site to enable anonymous access. In this case, K2 will use the Application Pool Identity to authenticate against K2 and perform actions (of course you need to be very careful of the security implications of doing this). See http://help.k2.com/onlinehelp/K2smartforms/ICG/4.6.10/default.htm#Anonymous_Access.html for more information on configuring anonymous access.

When SmartForms use SmartObjects to interact with providers, the security context depends on the configuration of the Service Instance. Depending on this configuration and whether it is possible to use or translate user credentials, the user credentials may flow all the way through to the provider (through Impersonate or OAuth) or perhaps be converted to another set of credentials (SSO). They may even stop at the K2 Server (when the Service Account or static credentials are used).

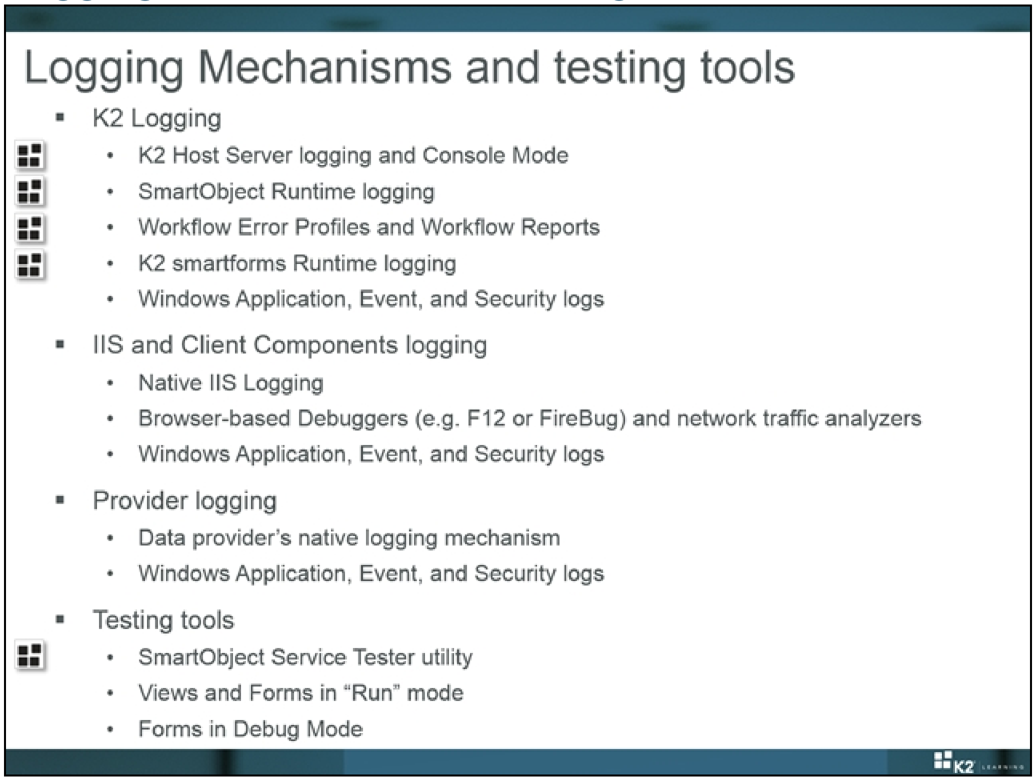
Impact of authentication on troubleshooting

If you are using multiple authentication mechanisms and security providers, you should try to keep in mind the “flow” of a user’s credentials. Very often, you will be faced with unexpected authorization or authentication errors when external users attempt to access certain resources. The easiest way of troubleshooting these issues is to use the available logging mechanisms (described in the next topic) and to draw out a flow diagram so that you understand where the user’s credentials are being passed, and the point at which the credentials stop and are no longer passed. You can also run the K2 Server in console mode and watch the output for security or authentication error messages.

Summary

- SmartForms authentication is tightly bound to IIS authentication along with the Security Providers/Security Labels concept in K2.
- In most standard situations, SmartForms leverage Active Directory and Windows credentials.
- In some cases you may want to allow external users to be authenticated through some other mechanism to work with SmartForms.
  - In this case you would usually set up a custom security provider in K2, then create a separate SmartForms runtime website and set it up to use Forms Authentication (pointing the configuration to the custom security provider).
- You can enable Multi-Auth to expose a single runtime site for all users, regardless of how they are authenticated.
- You can expose Forms to non-authenticated anonymous users by setting up a SmartForms runtime website to enable anonymous access.
- When SmartForms use SmartObjects to interact with providers, the security context depends on the configuration of the Service Instance.

Logging Mechanisms and testing tools



Logging Mechanisms and testing tools

- K2 Logging
  - K2 Host Server logging and Console Mode
  - SmartObject Runtime logging
  - Workflow Error Profiles and Workflow Reports
  - K2 smartforms Runtime logging
  - Windows Application, Event, and Security logs
- IIS and Client Components logging
  - Native IIS Logging
  - Browser-based Debuggers (e.g. F12 or FireBug) and network traffic analyzers
  - Windows Application, Event, and Security logs
- Provider logging
  - Data provider's native logging mechanism
  - Windows Application, Event, and Security logs
- Testing tools
  - SmartObject Service Tester utility
  - Views and Forms in "Run" mode
  - Forms in Debug Mode

Video

Consider again how K2 smartforms fit into the overall K2 architecture. When troubleshooting SmartForms and investigating logging, it is important to understand that each layer is normally responsible for a certain function and each layer normally has its own logging mechanism. There are different logging outputs available in K2 depending on the component you want to troubleshoot. Knowing which logging output to review will help when troubleshooting SmartForms. You should understand the communication path of the application (as discussed in the architecture topic earlier) to determine what logging output you should review.

Tip
When troubleshooting, start with the basics before adding the additional layers. In other words, if a View fails to load data, verify with the SmartObject Service Tester utility that the data is actually being returned before you try to troubleshoot the View. If a View fails to open at all, verify that the K2 Server is actually running and that you can open other Views. By testing components individually you can more readily identify where the problem might be occurring.

Native K2 Logging Output

K2 provides a number of internal logging mechanisms for various components and services.

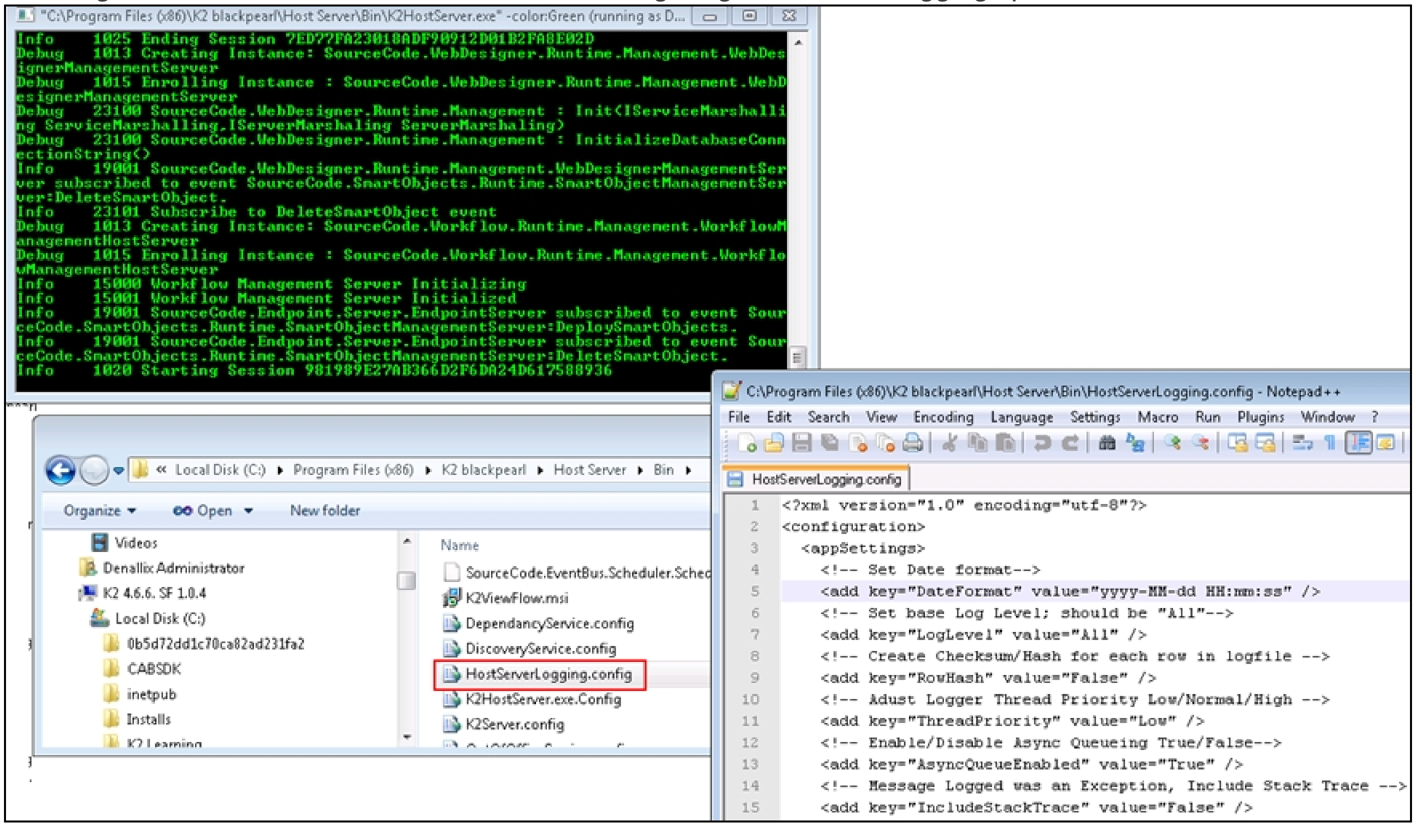
K2 Host Server Logging and Console Mode

To troubleshoot issues in the hosted services inside the K2 service, you can run the Host Server in console mode and enable logging output with various targets and levels of verbosity. You can run the K2 Host Server in console mode to get real-time output of logging on the server, which can be useful when troubleshooting authentication issues or intermittent issues.

This logging mechanism is useful for troubleshooting

- Authentication failures
- Communication failures
- Verifying that Actions are being sent to the K2 Server (and the correct K2 Server)
- Verifying that Workflows are being processed

Running the Host Server in console mode, and configuring Host Server logging options.


"C:\Program Files (x86)\K2 blackpearl\Host Server\Bin\K2HostServer.exe" -color:Green (running as D...
Local Disk (C:) ▸ Program Files (x86) ▸ K2 blackpearl ▸ Host Server ▸ Bin ▸
Organize ▾
Open ▾
New folder
Videos
Denallix Administrator
K2 4.6.6. SF 1.0.4
Local Disk (C:)
0b5d72dd1c70ca82ad231fa2
CABSDK
inetpub
Installs
Name
SourceCode.EventBus.Scheduler.Sched
K2ViewFlow.msi
DependancyService.config
DiscoveryService.config
HostServerLogging.config
K2HostServer.exe.Config
K2Server.config
C:\Program Files (x86)\K2 blackpearl\Host Server\Bin\HostServerLogging.config - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
HostServerLogging.config

```
Info    1025 Ending Session 7ED77FA23018ADF90912D01B2FA8E02D
Debug   1013 Creating Instance: SourceCode.WebDesigner.Runtime.Management.WebDes
ignerManagementServer
Debug   1015 Enrolling Instance : SourceCode.WebDesigner.Runtime.Management.WebD
esignerManagementServer
Debug   23100 SourceCode.WebDesigner.Runtime.Management : Init(IServiceMarshalli
ng ServiceMarshalling,IServerMarshaling ServerMarshaling)
Debug   23100 SourceCode.WebDesigner.Runtime.Management : InitializeDatabaseConn
ectionString()
Info    19001 SourceCode.WebDesigner.Runtime.Management.WebDesignerManagementSer
ver subscribed to event SourceCode.SmartObjects.Runtime.SmartObjectManagementSer
ver:DeleteSmartObject.
Info    23101 Subscribe to DeleteSmartObject event
Debug   1013 Creating Instance: SourceCode.Workflow.Runtime.Management.WorkflowM
anagementHostServer
Debug   1015 Enrolling Instance : SourceCode.Workflow.Runtime.Management.Workflo
wManagementHostServer
Info    15000 Workflow Management Server Initializing
Info    15001 Workflow Management Server Initialized
Info    19001 SourceCode.Endpoint.Server.EndpointServer subscribed to event Sour
ceCode.SmartObjects.Runtime.SmartObjectManagementServer:DeploySmartObjects.
Info    19001 SourceCode.Endpoint.Server.EndpointServer subscribed to event Sour
ceCode.SmartObjects.Runtime.SmartObjectManagementServer:DeleteSmartObject.
Info    1020 Starting Session 981989E27AB366D2F6DA24D617588936
```

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <!-- Set Date format-->
    <add key="DateFormat" value="yyyy-MM-dd HH:mm:ss" />
    <!-- Set base Log Level; should be "All"-->
    <add key="LogLevel" value="All" />
    <!-- Create Checksum/Hash for each row in logfile -->
    <add key="RowHash" value="False" />
    <!-- Adust Logger Thread Priority Low/Normal/High -->
    <add key="ThreadPriority" value="Low" />
    <!-- Enable/Disable Async Queueing True/False-->
    <add key="AsyncQueueEnabled" value="True" />
    <!-- Message Logged was an Exception, Include Stack Trace -->
    <add key="IncludeStackTrace" value="False" />
```

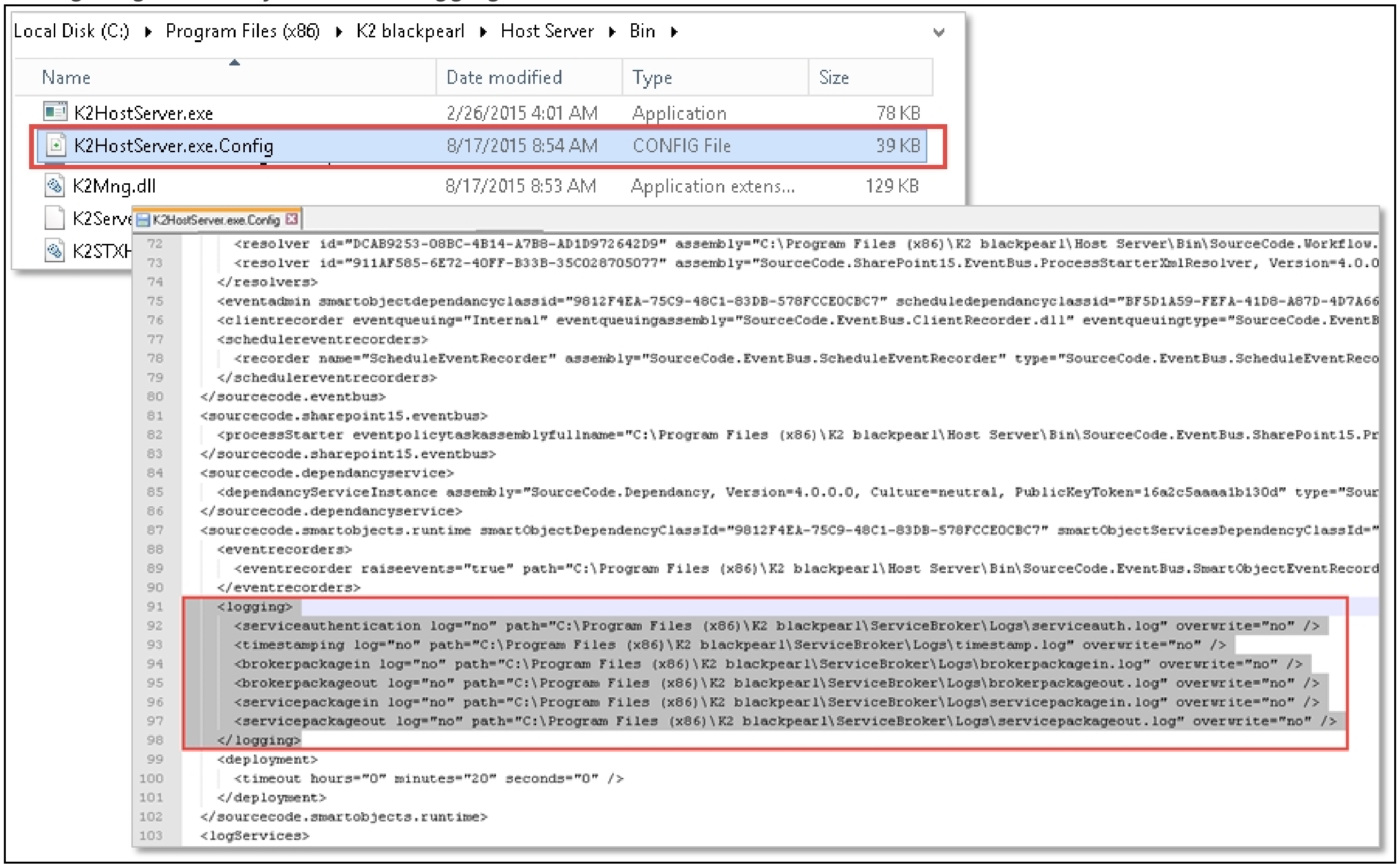
SmartObject Runtime Logging

This logging mechanism will output runtime activity for SmartObject methods, and will log the authentication, input values and return values when SmartObject methods are executed.

This logging mechanism is useful for troubleshooting:

- SmartObject execution errors
- Unexpected results when executing SmartObjects
- Authentication failures

Configuring SmartObject runtime logging



Local Disk (C:) ▸ Program Files (x86) ▸ K2 blackpearl ▸ Host Server ▸ Bin ▸
K2HostServer.exe.Config

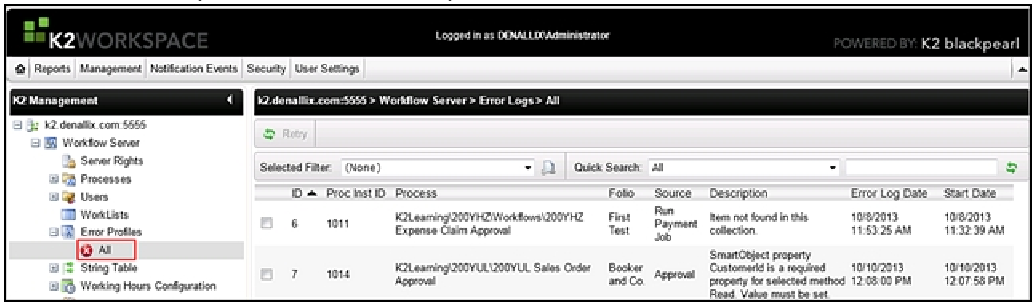
Name	Date modified	Type	Size
K2HostServer.exe	2/26/2015 4:01 AM	Application	78 KB
K2HostServer.exe.Config	8/17/2015 8:54 AM	CONFIG File	39 KB
K2Mng.dll	8/17/2015 8:53 AM	Application extens...	129 KB
K2Serv			
K2STXH			

```
      <resolver id="DCAB9253-08BC-4B14-A7B8-AD1D972642D9" assembly="C:\Program Files (x86)\K2 blackpearl\Host Server\Bin\SourceCode.Workflow.
      <resolver id="911AF585-6E72-40FF-B33B-35C028705077" assembly="SourceCode.SharePoint15.EventBus.ProcessStarterXmlResolver, Version=4.0.0
    </resolvers>
    <eventadmin smartobjectdependancyclassid="9812F4EA-75C9-48C1-83DB-578FCCE0CBC7" scheduledependancyclassid="BF5D1A59-FEFA-41D8-A87D-4D7A66
    <clientrecorder eventqueuing="Internal" eventqueuingassembly="SourceCode.EventBus.ClientRecorder.dll" eventqueuingtype="SourceCode.EventB
    <schedulereventrecorders>
      <recorder name="ScheduleEventRecorder" assembly="SourceCode.EventBus.ScheduleEventRecorder" type="SourceCode.EventBus.ScheduleEventReco
    </schedulereventrecorders>
  </sourcecode.eventbus>
  <sourcecode.sharepoint15.eventbus>
    <processStarter eventpolicytaskassemblyfullname="C:\Program Files (x86)\K2 blackpearl\Host Server\Bin\SourceCode.EventBus.SharePoint15.Pr
  </sourcecode.sharepoint15.eventbus>
  <sourcecode.dependancyservice>
    <dependancyServiceInstance assembly="SourceCode.Dependancy, Version=4.0.0.0, Culture=neutral, PublicKeyToken=16a2c5aaaa1b130d" type="Sour
  </sourcecode.dependancyservice>
  <sourcecode.smartobjects.runtime smartObjectDependencyClassId="9812F4EA-75C9-48C1-83DB-578FCCE0CBC7" smartObjectServicesDependencyClassId="
    <eventrecorders>
      <eventrecorder raiseevents="true" path="C:\Program Files (x86)\K2 blackpearl\Host Server\Bin\SourceCode.EventBus.SmartObjectEventRecord
    </eventrecorders>
    <logging>
      <serviceauthentication log="no" path="C:\Program Files (x86)\K2 blackpearl\ServiceBroker\Logs\serviceauth.log" overwrite="no" />
      <timestamping log="no" path="C:\Program Files (x86)\K2 blackpearl\ServiceBroker\Logs\timestamp.log" overwrite="no" />
      <brokerpackagein log="no" path="C:\Program Files (x86)\K2 blackpearl\ServiceBroker\Logs\brokerpackagein.log" overwrite="no" />
      <brokerpackageout log="no" path="C:\Program Files (x86)\K2 blackpearl\ServiceBroker\Logs\brokerpackageout.log" overwrite="no" />
      <servicepackagein log="no" path="C:\Program Files (x86)\K2 blackpearl\ServiceBroker\Logs\servicepackagein.log" overwrite="no" />
      <servicepackageout log="no" path="C:\Program Files (x86)\K2 blackpearl\ServiceBroker\Logs\servicepackageout.log" overwrite="no" />
    </logging>
    <deployment>
      <timeout hours="0" minutes="20" seconds="0" />
    </deployment>
  </sourcecode.smartobjects.runtime>
  <logServices>
```

Workflow Error Profiles and Workflow Reports

Workflows that fail with an error State are reported in the Error logs, which you can review using K2 Workspace or other K2 administration tools. Workflows that complete unexpectedly may not appear here (because they completed and did not fail), so the Workflow reports (especially the Process Overview and View Flow reports) are very useful when troubleshooting Workflows.

Workflow error profiles in K2 Workspace



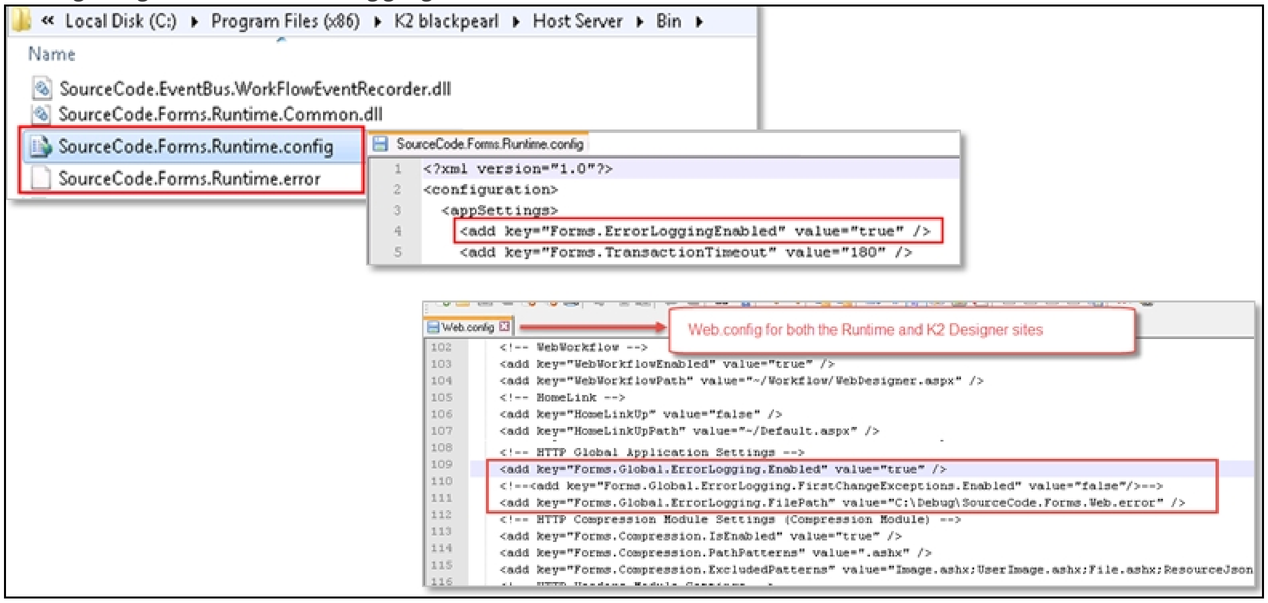
ID	Proc Inst ID	Process	Folio	Source	Description	Error Log Date	Start Date
6	1011	K2Learning\200YHZ\Workflows\200YHZ Expense Claim Approval	First Test	Run Payment Job	Item not found in this collection.	10/8/2013 11:53:25 AM	10/8/2013 11:32:39 AM
7	1014	K2Learning\200YUL\200YUL Sales Order Approval	Booker and Co.	Approval	SmartObject property CustomerId is a required property for selected method Read. Value must be set.	10/10/2013 12:08:00 PM	10/10/2013 12:07:58 PM

K2 smartforms Runtime Logging

You can configure runtime error output for SmartForms which will output errors to a specified text file. This option is useful for troubleshooting errors that are reported when trying to work with Forms on the runtime site or working with K2 Designer, such as:

- SmartForm runtime errors
- Unexpected behavior in Forms, Rules and Views
- Rules not executing correctly or not at all
- Incorrect values being displayed in the Form
- Design-time errors

Configuring Runtime Error logging for SmartForms sites

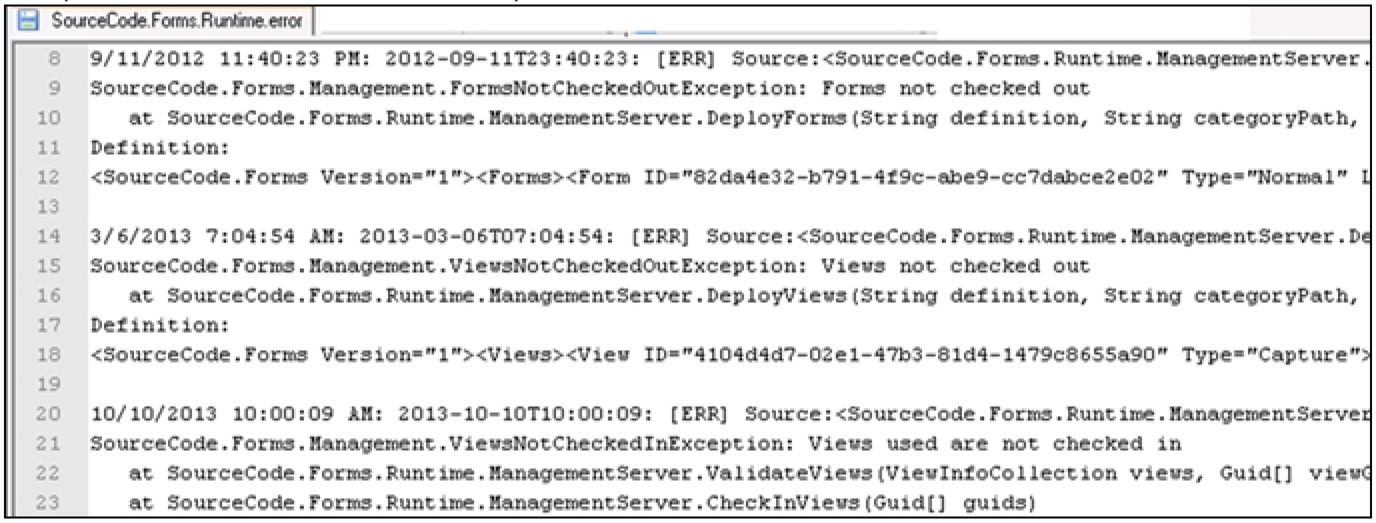


« Local Disk (C:) ▸ Program Files (x86) ▸ K2 blackpearl ▸ Host Server ▸ Bin ▸
Name
SourceCode.EventBus.WorkFlowEventRecorder.dll
SourceCode.Forms.Runtime.Common.dll
SourceCode.Forms.Runtime.config
SourceCode.Forms.Runtime.error
SourceCode.Forms.Runtime.config
Web.config
Web.config for both the Runtime and K2 Designer sites

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="Forms.ErrorLoggingEnabled" value="true" />
    <add key="Forms.TransactionTimeout" value="180" />
```

```
<!-- WebWorkflow -->
<add key="WebWorkflowEnabled" value="true" />
<add key="WebWorkflowPath" value="~/Workflow/WebDesigner.aspx" />
<!-- HomeLink -->
<add key="HomeLinkUp" value="false" />
<add key="HomeLinkUpPath" value="~/Default.aspx" />
<!-- HTTP Global Application Settings -->
<add key="Forms.Global.ErrorLogging.Enabled" value="true" />
<!--<add key="Forms.Global.ErrorLogging.FirstChangeExceptions.Enabled" value="false"/>-->
<add key="Forms.Global.ErrorLogging.FilePath" value="C:\Debug\SourceCode.Forms.Web.error" />
<!-- HTTP Compression Module Settings (Compression Module) -->
<add key="Forms.Compression.IsEnabled" value="true" />
<add key="Forms.Compression.PathPatterns" value=".ashx" />
<add key="Forms.Compression.ExcludedPatterns" value="Image.ashx;UserImage.ashx;File.ashx;ResourceJson
```

Sample of SmartForms runtime error output



```
9/11/2012 11:40:23 PM: 2012-09-11T23:40:23: [ERR] Source:<SourceCode.Forms.Runtime.ManagementServer.
SourceCode.Forms.Management.FormsNotCheckedOutException: Forms not checked out
   at SourceCode.Forms.Runtime.ManagementServer.DeployForms(String definition, String categoryPath,
Definition:
<SourceCode.Forms Version="1"><Forms><Form ID="82da4e32-b791-4f9c-abe9-cc7dabce2e02" Type="Normal" I

3/6/2013 7:04:54 AM: 2013-03-06T07:04:54: [ERR] Source:<SourceCode.Forms.Runtime.ManagementServer.De
SourceCode.Forms.Management.ViewsNotCheckedOutException: Views not checked out
   at SourceCode.Forms.Runtime.ManagementServer.DeployViews(String definition, String categoryPath,
Definition:
<SourceCode.Forms Version="1"><Views><View ID="4104d4d7-02e1-47b3-81d4-1479c8655a90" Type="Capture">

10/10/2013 10:00:09 AM: 2013-10-10T10:00:09: [ERR] Source:<SourceCode.Forms.Runtime.ManagementServer
SourceCode.Forms.Management.ViewsNotCheckedInException: Views used are not checked in
   at SourceCode.Forms.Runtime.ManagementServer.ValidateViews(ViewInfoCollection views, Guid[] viewC
   at SourceCode.Forms.Runtime.ManagementServer.CheckInViews(Guid[] guids)
```

Windows Application, Event and Security Logs

The Windows event logs on the K2, IIS and SQL servers used in a K2 environment sometimes contain useful information. Checking the Windows event logs should always be included in troubleshooting steps.

IIS and Client Components Logging

You may need to use logging on the IIS web servers that host the K2 Designer or Runtime websites. Review the native IIS log files for logging output related to IIS authentication and configuration issues. You should also review the Windows Application, Event and Security logs on the IIS server as these logs may contain useful information as well.

To troubleshoot issues reported on client machines, you can use browser-based debuggers (such as F12 in Internet Explorer or FireBug in FireFox) and network traffic analyzers (such as WireShark). These tools are useful when you have verified that the SmartObject data is coming through correctly with the SmartObject Service Tester utility, but is being rendered incorrectly in the Form, or when HTML or JavaScript errors are reported.

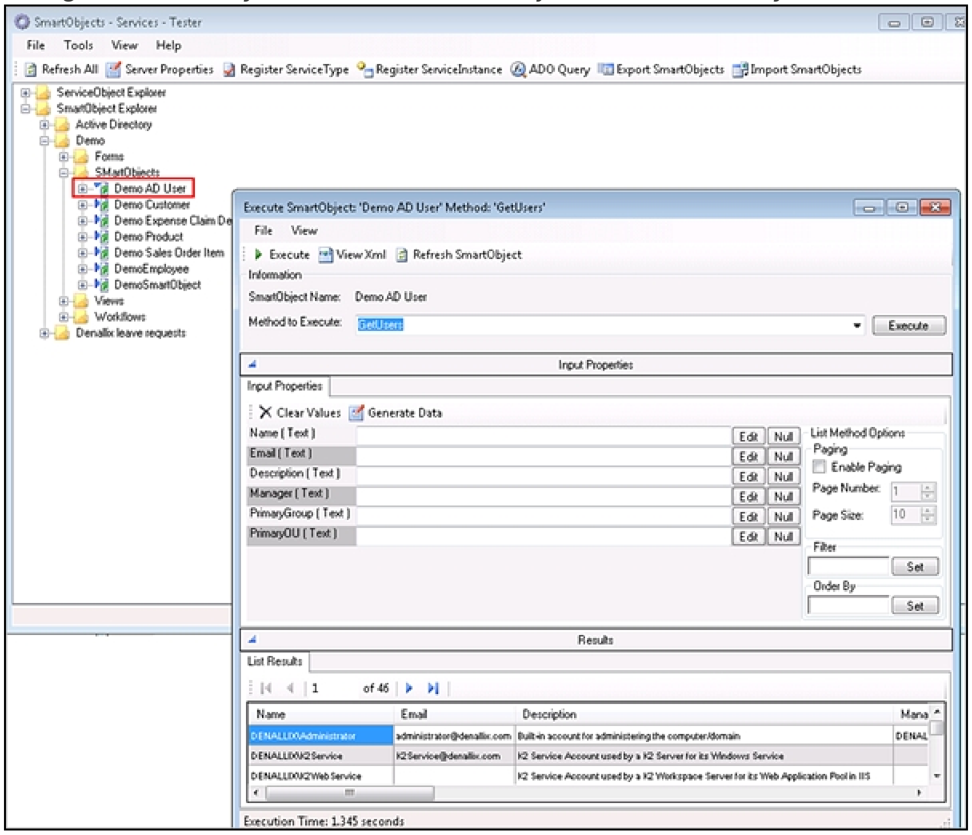
Provider Logging

If the data from SmartObjects is not appearing as expected when you use tools like the SmartObject Service Tester utility, you may want to investigate the logging output from the data provider. The logging output depends on the provider's available logging output, for example, SQL traces. As usual, always check the Windows event logs to see if any useful information is output there as well.

Testing tools

The most useful testing tool when troubleshooting SmartForms is the K2 SmartObject Service Tester utility. This should be your first test tool to turn to when troubleshooting SmartObject issues, since it will remove the SmartForms layer completely which will help you determine if the issue is on the SmartForm side or somewhere between the SmartObject and the data provider. You have worked with this utility in the K2 blackpearl Core training course, so we won't go into too much more detail here. You can also run K2 SmartForms in debug mode to output logging information, which we will look at in the next topic.

Using the SmartObject Service Tester utility to test a SmartObject



Execute SmartObject: 'Demo AD User' Method: 'GetUsers'

Information

SmartObject Name: Demo AD User

Method to Execute: GetUsers

Input Properties: Name (Text), Email (Text), Description (Text), Manager (Text), PrimaryGroup (Text), PrimaryOU (Text)

Page Number: 1
Page Size: 10
Filter
Order By

1 of 46

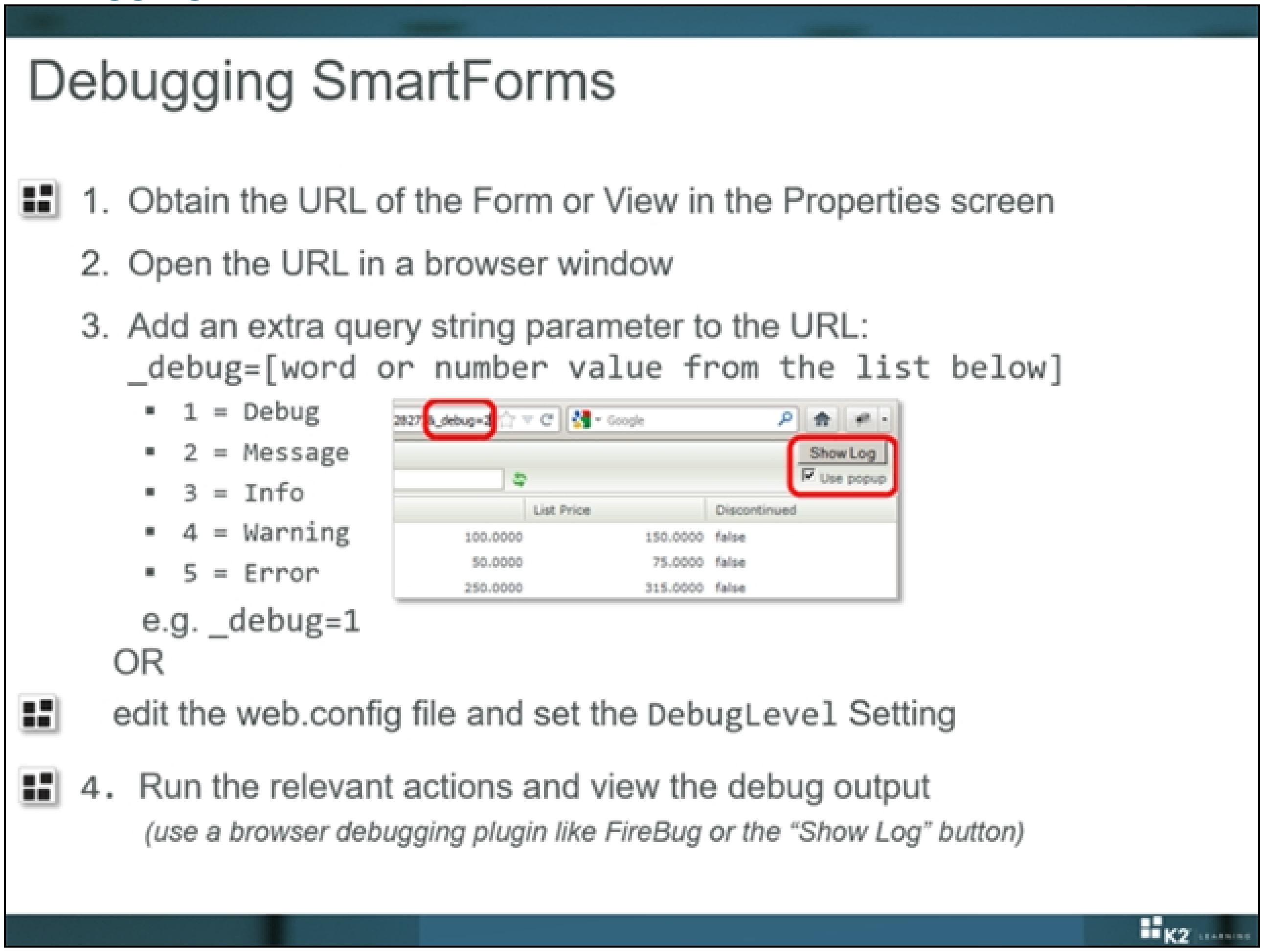
Name	Email	Description	Mana
DENALLIX\Administrator	administrator@denallix.com	Built-in account for administering the computer/domain	DENAL
DENALLIX\K2Service	K2Service@denallix.com	K2 Service Account used by a K2 Server for its Windows Service	
DENALLIX\K2Web Service		K2 Service Account used by a K2 Workspace Server for its Web Application Pool in IIS	

Execution Time: 1.345 seconds

Summary

- K2 native logging
  - K2 Host Server logging and Console Mode (host server operations)
  - SmartObject Runtime logging (SmartObject operations, authentication and data packets)
  - Workflow Error Profiles and Workflow Reports (Workflows that fail with error State or complete unexpectedly)
  - K2 smartforms Runtime logging (runtime error output for SmartForms)
  - Windows Application, Event and Security logs
- IIS and Client Components logging
  - Native IIS logging (especially IIS authentication and configuration issues)
  - Browser-based debuggers (e.g. F12 or FireBug) and network traffic analyzers
  - Windows Application, Event and Security logs on the IIS server
- Provider logging
  - Data provider's native logging mechanism (depends on the providers available output)
  - Views and Forms in "Run" mode – run Views and Forms directly to test them
  - Forms in Debug Mode – you can add the debug parameter to Forms to output debug messages. (See next topic.)
- SmartObject Service Tester utility is very useful to verify that the Service Broker and SmartObject are working as expected.

Debugging SmartForms



Debugging SmartForms

1. Obtain the URL of the Form or View in the Properties screen
2. Open the URL in a browser window
3. Add an extra query string parameter to the URL:
   `_debug=[word or number value from the list below]`
   - `1 = Debug`
   - `2 = Message`
   - `3 = Info`
   - `4 = Warning`
   - `5 = Error`

   e.g. `_debug=1`

   OR

   edit the web.config file and set the `DebugLevel` Setting
4. Run the relevant actions and view the debug output

   (use a browser debugging plugin like FireBug or the "Show Log" button)

	List Price	Discontinued
100.0000	150.0000	false
50.0000	75.0000	false
250.0000	315.0000	false

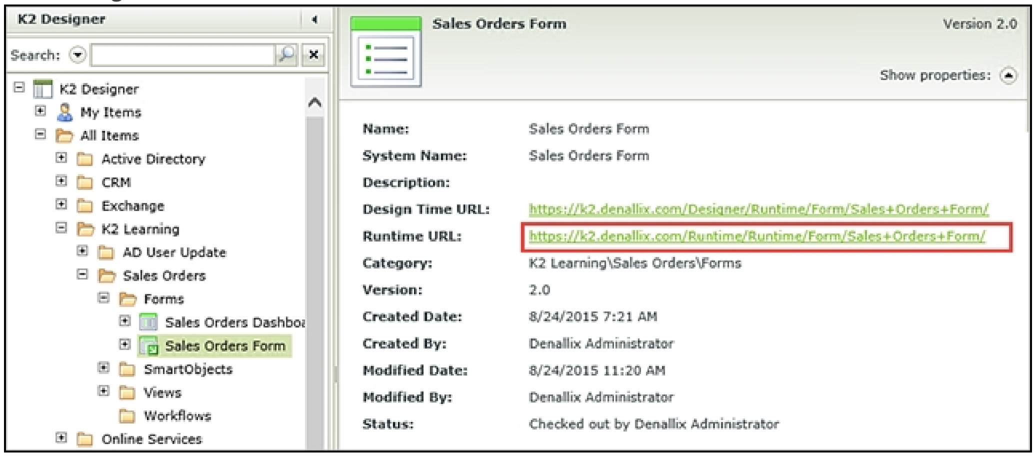
Video

From time to time, you may need to debug SmartForms to troubleshoot runtime errors or unexpected behavior. To run a Form or View with debugging enabled, simply open the Form or View's URL with the debug parameter (**_debug=[n]**) appended to the URL (or you can edit the web.config file to enable debugging).

The [n] part is a number value from 1 to 5 that you can change to output more or fewer details in the debug output. To run a Form with the most detailed level of output, add the query string parameter *_debug=1* to the Form URL. To only show errors in the debug output, add the query string parameter *_debug=5* to the Form URL.

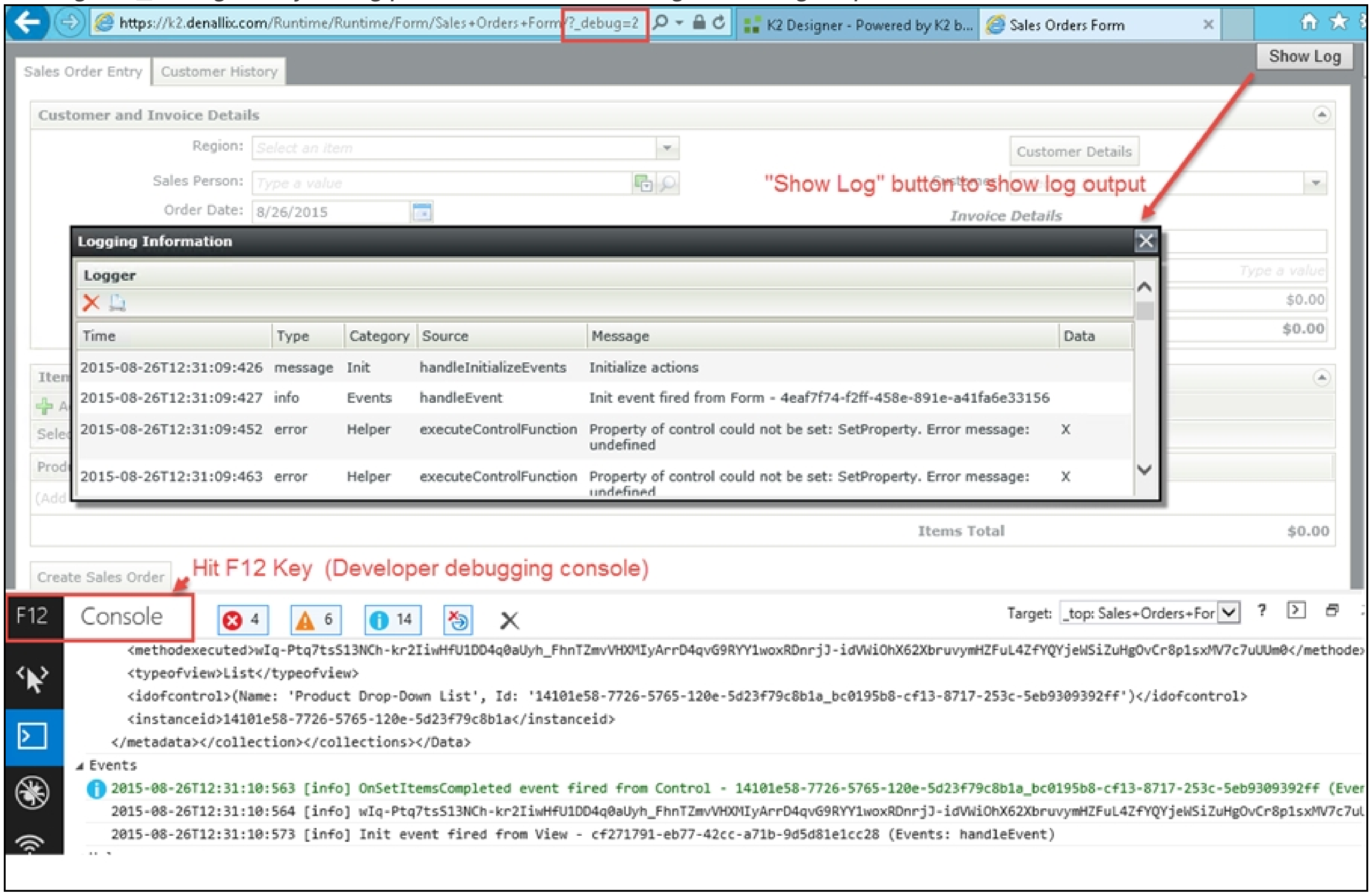
- Debug = 1
- Message = 2
- Info = 3
- Warning = 4
- Error = 5

Obtaining a Form's runtime URL



Name:	Sales Orders Form
System Name:	Sales Orders Form
Description:	
Design Time URL:	https://k2.denallix.com/Designer/Runtime/Form/Sales+Orders+Form/
Runtime URL:	https://k2.denallix.com/Runtime/Runtime/Form/Sales+Orders+Form/
Category:	K2 Learning\Sales Orders\Forms
Version:	2.0
Created Date:	8/24/2015 7:21 AM
Created By:	Denallix Administrator
Modified Date:	8/24/2015 11:20 AM
Modified By:	Denallix Administrator
Status:	Checked out by Denallix Administrator

Adding the _debug Query String parameter and reviewing the debug output



Logging Information

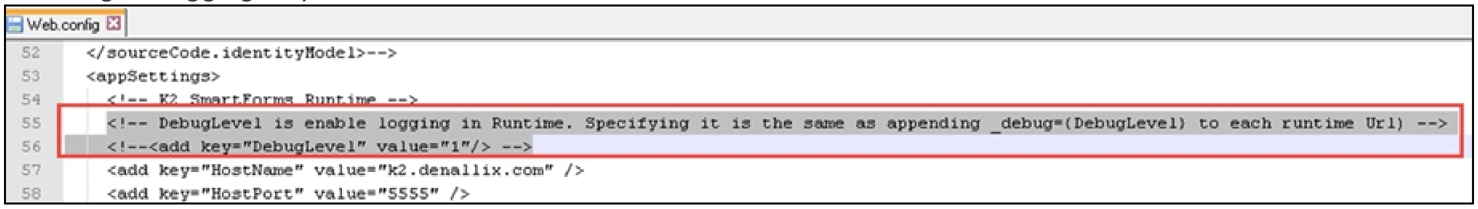
Logger

Time	Type	Category	Source	Message	Data
2015-08-26T12:31:09:426	message	Init	handleInitializeEvents	Initialize actions	
2015-08-26T12:31:09:427	info	Events	handleEvent	Init event fired from Form - 4eaf7f74-f2ff-458e-891e-a41fa6e33156	
2015-08-26T12:31:09:452	error	Helper	executeControlFunction	Property of control could not be set: SetProperty. Error message: undefined	X
2015-08-26T12:31:09:463	error	Helper	executeControlFunction	Property of control could not be set: SetProperty. Error message: undefined	X

```
    <methodexecuted>wIq-Ptq7tsS13NCh-kr2IiwHfU1DD4q0aUyh_FhnTZmvVHXMIyArrD4qvG9RYY1woxRDnrjJ-idVWiOhX62XbruvymHZFuL4ZfYQYjeWSiZuHgOvCr8p1sxMV7c7uUUm0</methode
    <typeofview>List</typeofview>
    <idofcontrol>(Name: 'Product Drop-Down List', Id: '14101e58-7726-5765-120e-5d23f79c8b1a_bc0195b8-cf13-8717-253c-5eb9309392ff')</idofcontrol>
    <instanceid>14101e58-7726-5765-120e-5d23f79c8b1a</instanceid>
  </metadata></collection></collections></Data>
◢ Events
  2015-08-26T12:31:10:563 [info] OnSetItemsCompleted event fired from Control - 14101e58-7726-5765-120e-5d23f79c8b1a_bc0195b8-cf13-8717-253c-5eb9309392ff (Ever
  2015-08-26T12:31:10:564 [info] wIq-Ptq7tsS13NCh-kr2IiwHfU1DD4q0aUyh_FhnTZmvVHXMIyArrD4qvG9RYY1woxRDnrjJ-idVWiOhX62XbruvymHZFuL4ZfYQYjeWSiZuHgOvCr8p1sxMV7c7uL
  2015-08-26T12:31:10:573 [info] Init event fired from View - cf271791-eb77-42cc-a71b-9d5d81e1cc28 (Events: handleEvent)
```

Alternatively, you can enable debugging output for all Forms in a site by editing the web.config file for that site, as shown below. Just remember to disable the debug output when you are done debugging, since the output can affect the performance of the Form at runtime.

Enabling debugging output fr a Site



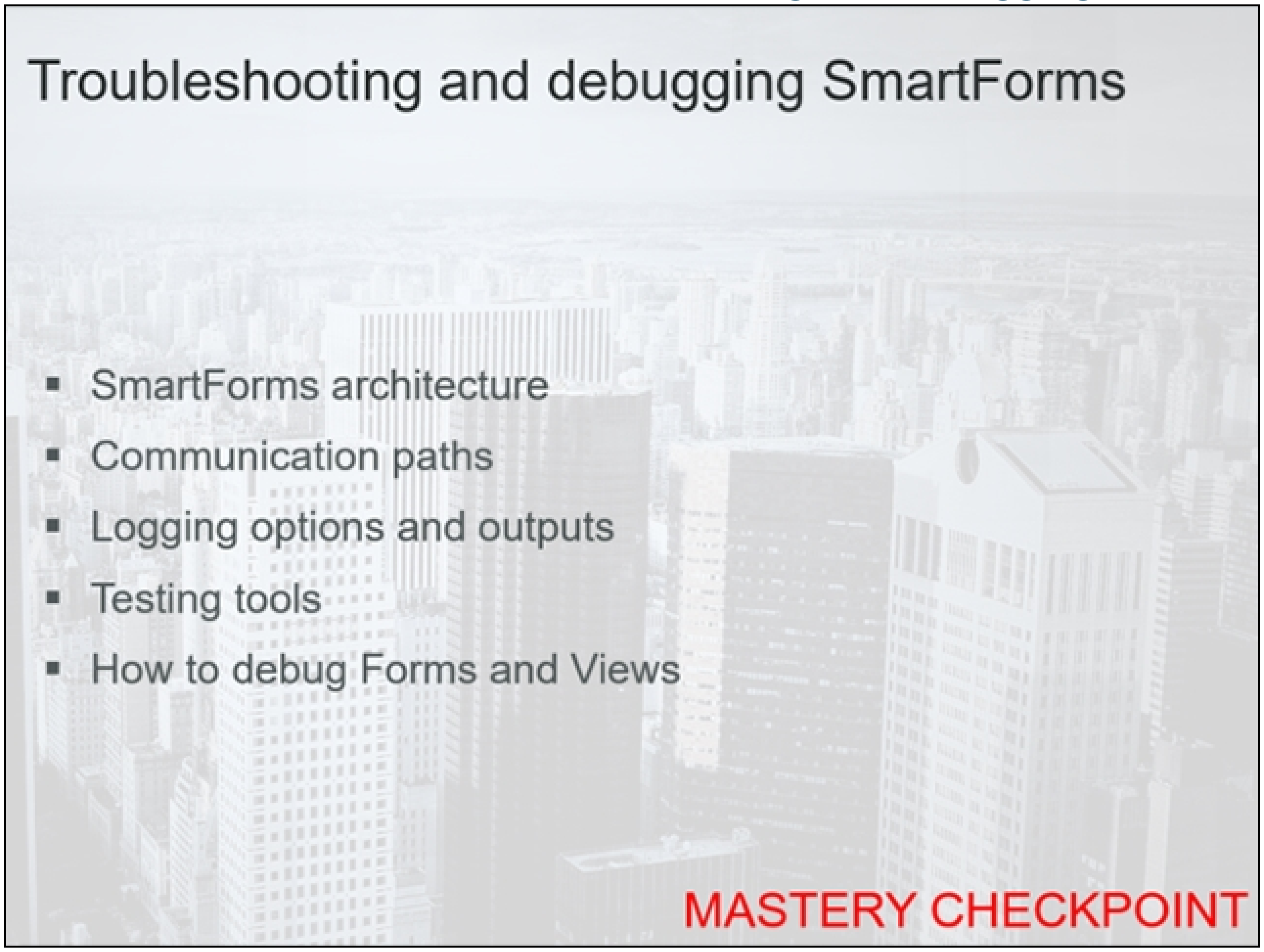
```
  </sourceCode.identityModel>-->
  <appSettings>
    <!-- K2 SmartForms Runtime -->
    <!-- DebugLevel is enable logging in Runtime. Specifying it is the same as appending _debug=(DebugLevel) to each runtime Url) -->
    <!--<add key="DebugLevel" value="1"/> -->
    <add key="HostName" value="k2.denallix.com" />
    <add key="HostPort" value="5555" />
```

You may need to play around with the level of logging (in other words, the number value added to the end of the query string) to see the log entries that are relevant for troubleshooting your issue. Remember also to use the standard K2 logging described in previous topics along with SmartForms logging to track down K2-relevant issues. You may need to review multiple log files before you discover the cause of unexpected behavior or errors.

Summary

- Append a query string parameter to the Form URL (_*debug=[level]*), or edit the web.config file to enable debugging.
- View the log output with the “View Log” button or by running browser debugging tools like F12 in IE or FireBug in Firefox, and review the console output.
- Limit the amount of logging output returned by selecting a different level (Level 1 is most verbose, Level 5 is least verbose).

MASTERY CHECKPOINT: Troubleshooting and debugging SmartForms



Troubleshooting and debugging SmartForms

- SmartForms architecture
- Communication paths
- Logging options and outputs
- Testing tools
- How to debug Forms and Views

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 6 of this module. If you are attending instructor-led training, use this as an opportunity to answer any questions around troubleshooting and debugging SmartForms.

After completing Part 6 you should know:

- The basic architecture of SmartForms
- How communication flows between users, SmartForms, K2 and underlying systems
- What logging options and outputs to configure and review when troubleshooting SmartForms
- What tools are available to test SmartForms and SmartObjects
- How to use the Debug logging output for Forms and Views

Knowledge-check questions

Q: Suppose a List View that displayed data from SQL is reporting an error. How would you go about troubleshooting the error?
Reveal answer**A:** It depends on the error, but the basic process is to verify that the SmartObject actually retrieves data with something like the SmartObject Service Tester utility, then go from there. If data is returned then the problem must be on the View. If data is not returned the problem must be somewhere in the SmartObject side of things.

Q: Which of the following debug levels is most verbose: Level 2 (Message) or Level 4 (Warning)?
Reveal answer**A:** Level 2 is more verbose, lower number are more verbose.

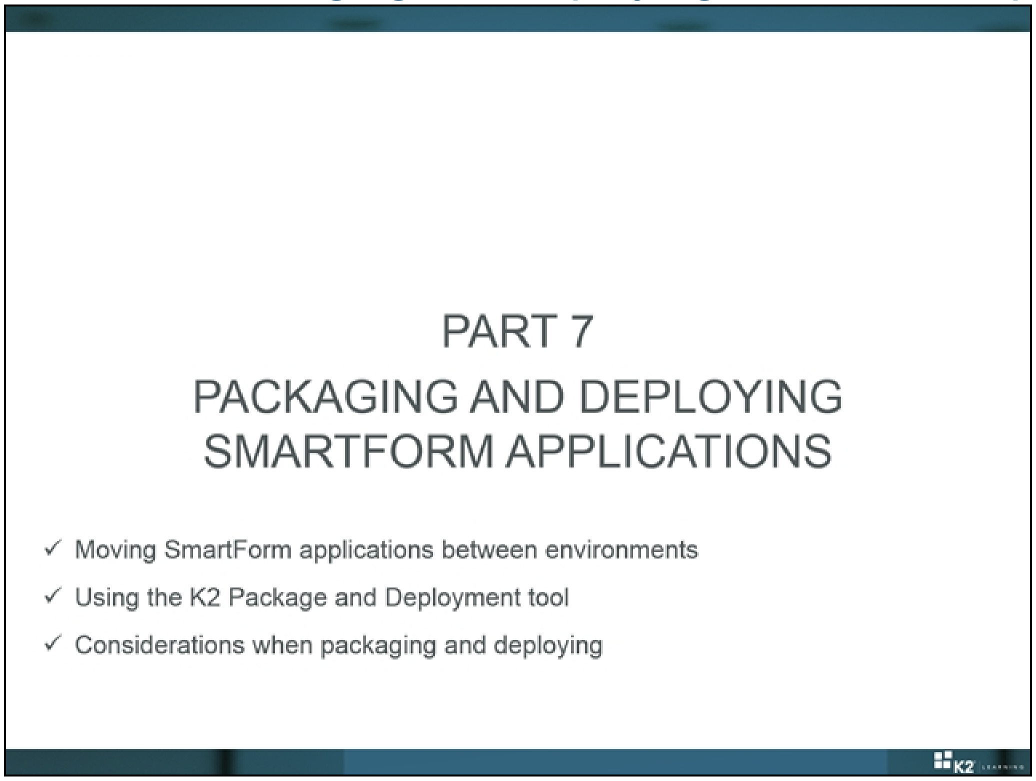
Q: Can you use tools like Firebug or F12 IE developer tools to debug Forms and Views?
Reveal answer**A:** Yes

Q: In SmartForms, what is the difference between the K2 Designer website and the runtime Website?
Reveal answer**A:** K2 Designer is the site that hosts the K2 Designer and is used to build SmartForms, SmartObjects and Workflows. Runtime is the website that end users use to interact with Forms.

Q: What communication protocol is used between client machines and the SmartForms Runtime site?
Reveal answer**A:** HTTP or HTTPS

Q:What communication protocol is used between the K2 Server and providers?
Reveal answer**A:** It depends on the provider’s API.

PART 7: Packaging and Deploying SmartForm Applications



PART 7

PACKAGING AND DEPLOYING SMARTFORM APPLICATIONS

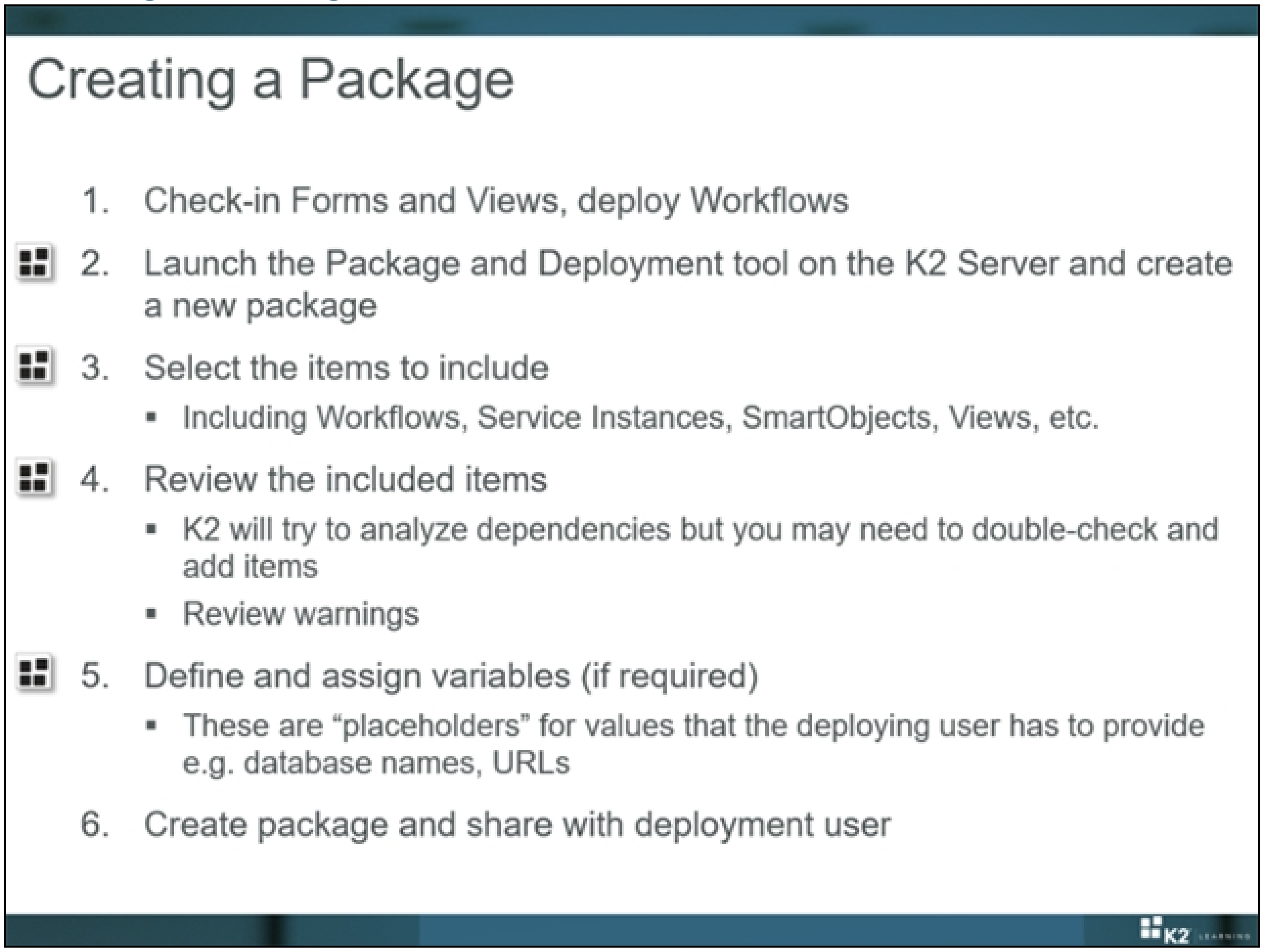
- ✓ Moving SmartForm applications between environments
- ✓ Using the K2 Package and Deployment tool
- ✓ Considerations when packaging and deploying

Video

In Part 7 we will look at how to Package and Deploy SmartForm applications between environments. We will learn how to use the K2 Package and Deployment tool to create deployment packages and then deploy those packages to a different environment. We will also cover some considerations when packaging and deploying SmartForms applications.

At the end of Part 7, we will do an exercise to create a package of the artifacts in our Sales Orders application. Because we only have a single environment to work with, we will just deploy that same package back to the current environment to simulate deploying the application.

Creating a Package



Creating a Package

1. Check-in Forms and Views, deploy Workflows
2. Launch the Package and Deployment tool on the K2 Server and create a new package
3. Select the items to include
   - Including Workflows, Service Instances, SmartObjects, Views, etc.
4. Review the included items
   - K2 will try to analyze dependencies but you may need to double-check and add items
   - Review warnings
5. Define and assign variables (if required)
   - These are “placeholders” for values that the deploying user has to provide e.g. database names, URLs
6. Create package and share with deployment user

Video

Your organization may have more than one K2 environment (for example, separated Development, Testing and Production environments). In these situations, the usual approach is to build a solution in the Development environment and then deploy the solution components to the Test environment, then onward to the Production environment.

To deploy SmartForm-based solutions to different environments, you will need to use the K2 Package and Deployment tool. This tool allows you to select a range of artifacts (and optionally the dependencies for those artifacts), and create a single deployment package (.kspx) file. This file can then be deployed to a different K2 environment using the Package and Deployment tool or a PowerShell-based script.

Naturally, the first part of packaging and deploying solutions is to create a deployment package, You will do this in the environment were the application was created (for example, your Development environment) and we will refer to this as the Source environment. When using the K2 Package and Deployment tool, you need to run this tool on one of the physical K2 Application servers in that environment.

Steps to create a deployment package:

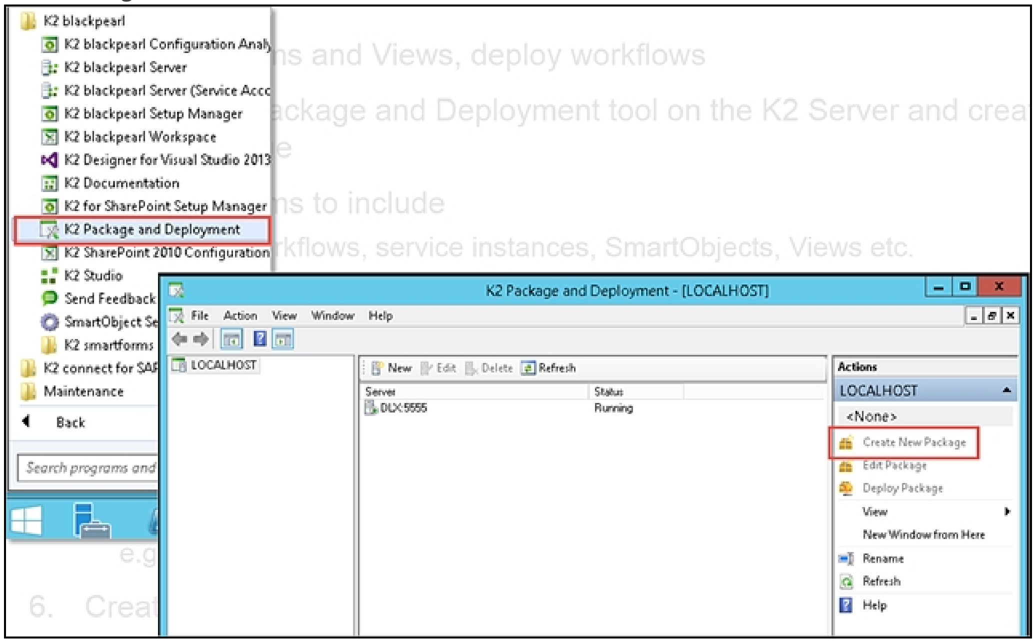
1. Check-in Forms and Views, deploy Workflows.

   The Package and Deployment (P&D) tool only grabs artifacts from the K2 server. This means you should check in any Forms and Views to be included in the application, and deploy any Workflows or K2 Studio projects to the source environment before creating a new package. (Generally speaking, P&D uses the last checked-in version of a Form/View so it is always safer to check in a Form/View before creating a package for that item.)

2. Launch the Package and Deployment tool on the K2 Server and create a new package.

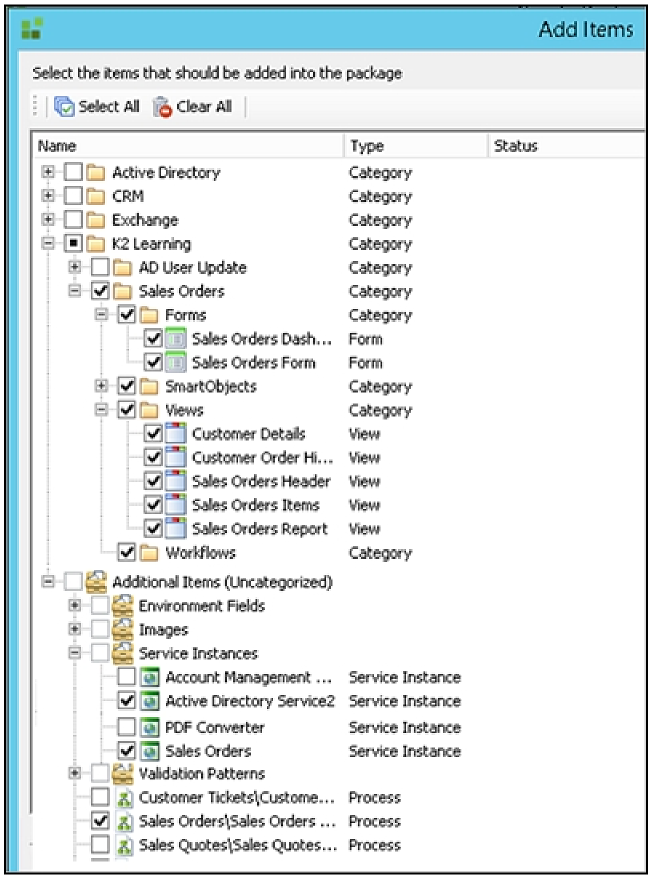
Note
Note that you have to run the MMC-based P&D tool on a physical K2 Application server in the Source environment.

Launching the P&D tool



3. Select the items to include.

Packages normally include Workflows, Service Instances, SmartObjects, Views, Forms, etc. Select all of the items that need to be included in your package. (This is another case where a well-structured Category Browser folder system will be useful!) The P&D tool will include all of the items in the selected folders for you in the deployment package.



Add Items

Select the items that should be added into the package

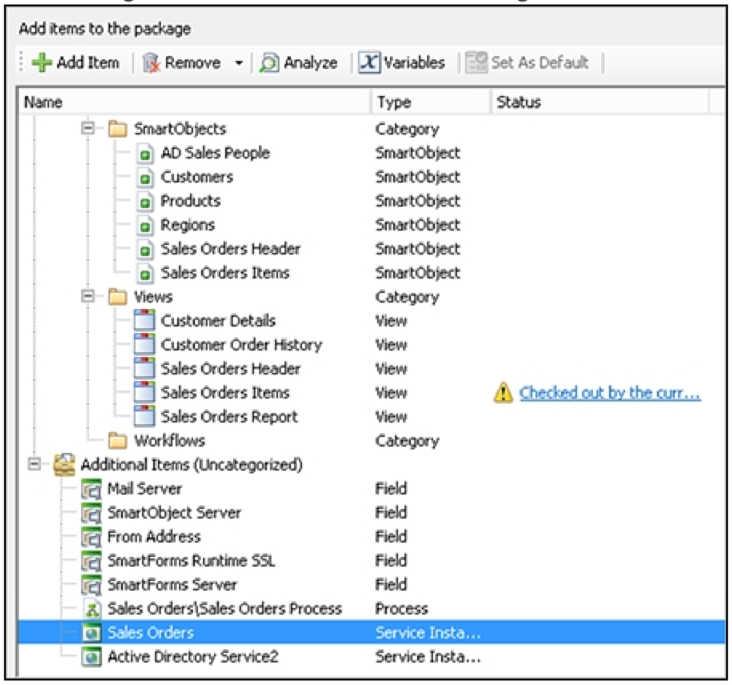
Select All | Clear All

Name	Type	Status
☐ Active Directory	Category	
☐ CRM	Category	
☐ Exchange	Category	
■ K2 Learning	Category	
☐ AD User Update	Category	
✓ Sales Orders	Category	
✓ Forms	Category	
✓ Sales Orders Dash...	Form	
✓ Sales Orders Form	Form	
✓ SmartObjects	Category	
✓ Views	Category	
✓ Customer Details	View	
✓ Customer Order Hi...	View	
✓ Sales Orders Header	View	
✓ Sales Orders Items	View	
✓ Sales Orders Report	View	
✓ Workflows	Category	
☐ Additional Items (Uncategorized)		
☐ Environment Fields		
☐ Images		
☐ Service Instances		
☐ Account Management ...	Service Instance	
✓ Active Directory Service2	Service Instance	
☐ PDF Converter	Service Instance	
✓ Sales Orders	Service Instance	
☐ Validation Patterns		
☐ Customer Tickets\Custome...	Process	
✓ Sales Orders\Sales Orders ...	Process	
☐ Sales Quotes\Sales Quotes...	Process	

4. Review the included items and review any warnings or errors.

   If you selected the **Automatically include dependent items** check box when creating the package, K2 will do a best-guess to resolve dependencies and include the artifacts that are associated with the items you selected (for example, any Views used by Forms you select, or SmartObjects used by the Views you select). You may need to double-check and verify that all the necessary items are selected.

   Reviewing the included items and warnings

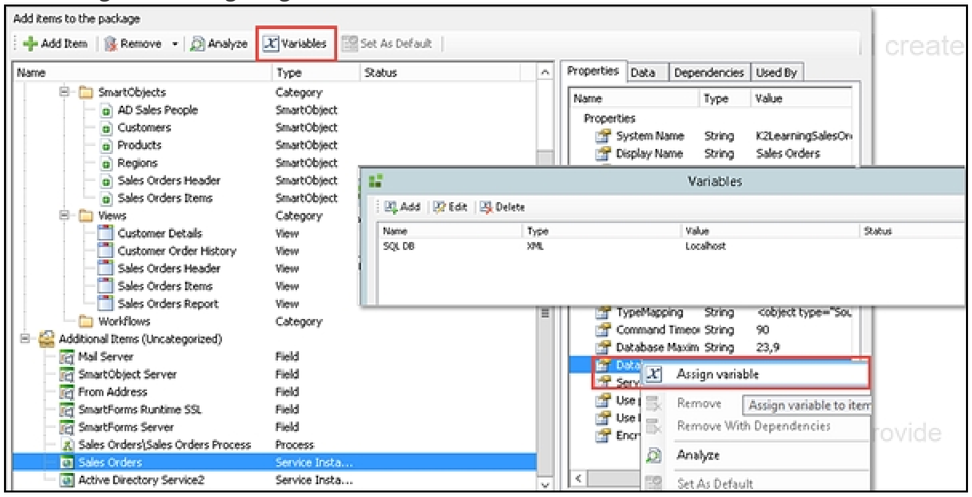


Name	Type	Status
SmartObjects	Category	
AD Sales People	SmartObject	
Customers	SmartObject	
Products	SmartObject	
Regions	SmartObject	
Sales Orders Header	SmartObject	
Sales Orders Items	SmartObject	
Views	Category	
Customer Details	View	
Customer Order History	View	
Sales Orders Header	View	
Sales Orders Items	View	Checked out by the curr...
Sales Orders Report	View	
Workflows	Category	
Additional Items (Uncategorized)		
Mail Server	Field	
SmartObject Server	Field	
From Address	Field	
SmartForms Runtime SSL	Field	
SmartForms Server	Field	
Sales Orders\Sales Orders Process	Process	
Sales Orders	Service Insta...	
Active Directory Service2	Service Insta...	

5. Define and assign variables (if required).

   If necessary, you can create variables and assign properties to those variables in the package. Variables are “placeholders” for values that the deploying user has to provide and are usually used for things like database names and URLs required by Service Instances; email addresses to include in the Environment library and so on.

   Creating and assigning a Variable



Name	Type	Status
SmartObjects	Category	
AD Sales People	SmartObject	
Customers	SmartObject	
Products	SmartObject	
Regions	SmartObject	
Sales Orders Header	SmartObject	
Sales Orders Items	SmartObject	
Views	Category	
Customer Details	View	
Customer Order History	View	
Sales Orders Header	View	
Sales Orders Items	View	
Sales Orders Report	View	
Workflows	Category	
Additional Items (Uncategorized)		
Mail Server	Field	
SmartObject Server	Field	
From Address	Field	
SmartForms Runtime SSL	Field	
SmartForms Server	Field	
Sales Orders\Sales Orders Process	Process	
Sales Orders	Service Insta...	
Active Directory Service2	Service Insta...	

Name	Type	Value
Properties		
System Name	String	K2LearningSalesOr
Display Name	String	Sales Orders
TypeMapping	String	<object type="Sou
Command Timeo	String	90
Database Maxim	String	23,9
Data		
Serv		
Use		
Use		
Encr		

Name	Type	Value	Status
SQL DB	XML	Localhost	

6. Create the package and share the resulting .kspx file with deployment user.

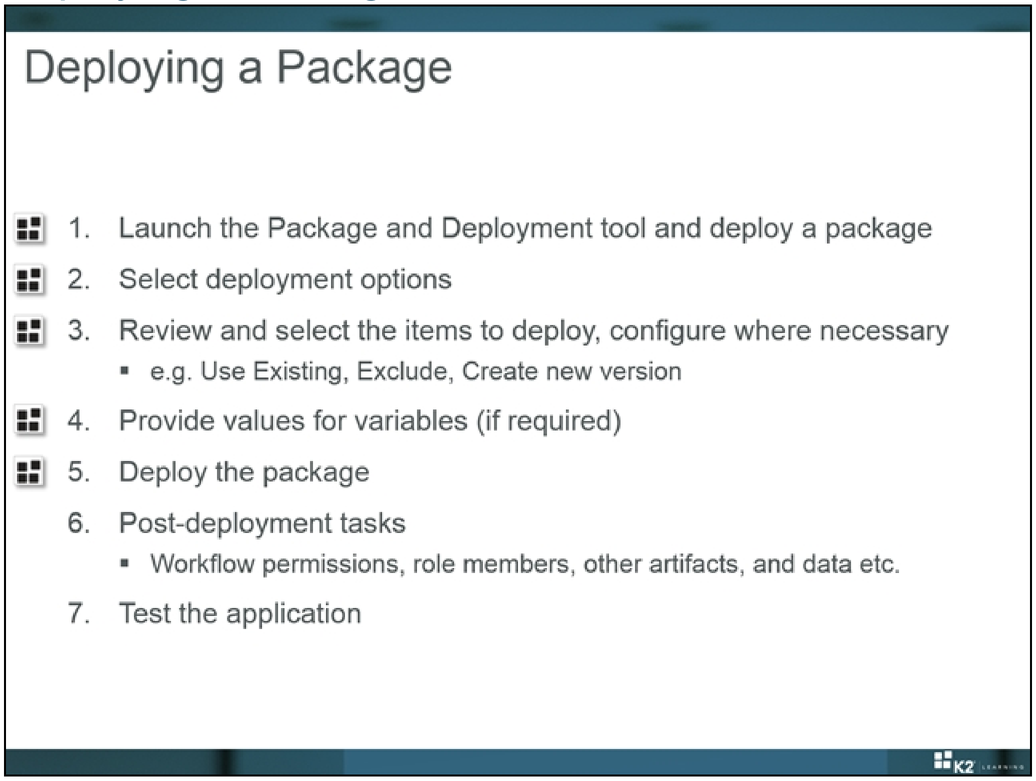
Summary

Steps to create a deployment package:

1. Check-in Forms and Views, deploy Workflows
2. Launch the Package and Deployment tool on the K2 Application Server and create a new package

3. Select the items to include
4. Review the included items
5. Define and assign variables (if required)
6. Create the package and share the resulting .kspx file with deployment user

Deploying a Package



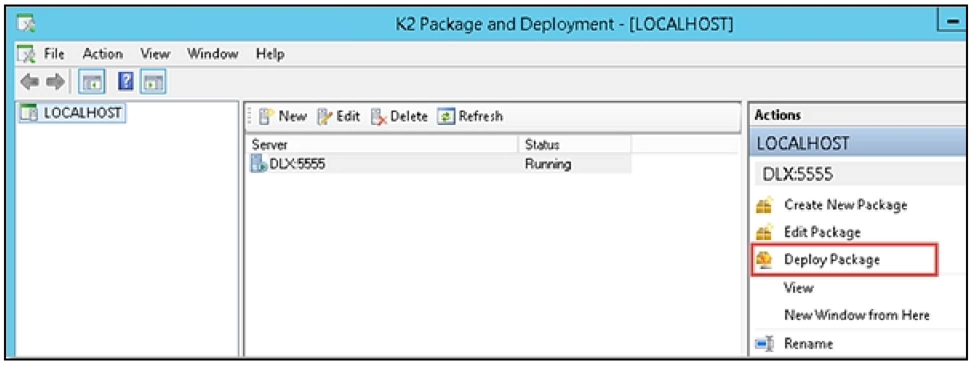
Deploying a Package

1. Launch the Package and Deployment tool and deploy a package
2. Select deployment options
3. Review and select the items to deploy, configure where necessary
   - e.g. Use Existing, Exclude, Create new version
4. Provide values for variables (if required)
5. Deploy the package
6. Post-deployment tasks
   - Workflow permissions, role members, other artifacts, and data etc.
7. Test the application

To deploy (.kspx) packages in an environment, you will again need to use the K2 Package and Deployment tool on a K2 Application Server in the target environment.

Steps to create a deployment package:

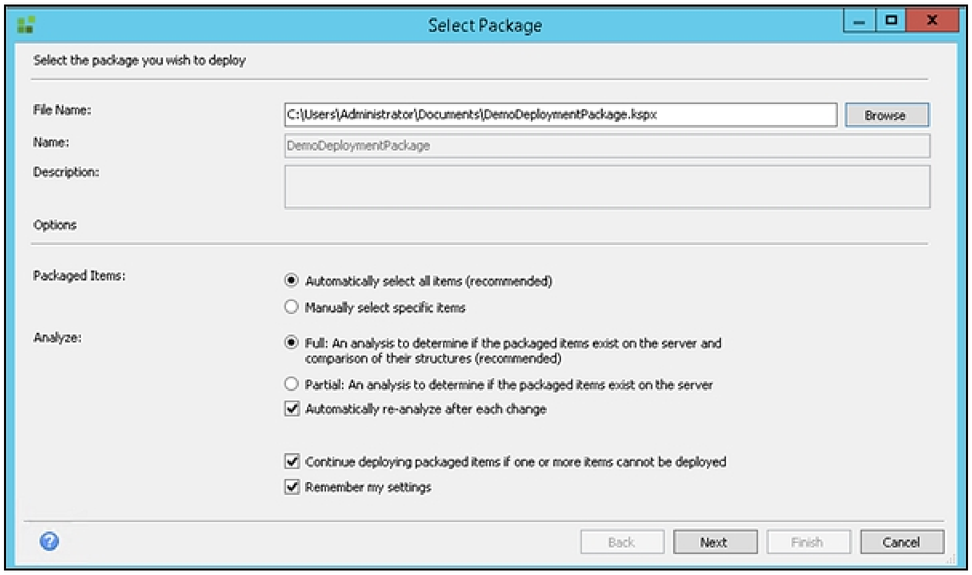
1. Launch the Package & Deployment tool and select “Deploy Package”.



2. Browse for the (.kspx) package you want to deploy, and set the deployment options:
   - Whether all items should be deployed or you will select which items to deploy. If you are deploying a brand-new application to an environment or if you are updating all the components for an application and these components are not used by any other application, you can select the **Automatically select all items** option. If you are a more advanced user or if your application contains Items that are used by other application (such as shared SmartObjects, Service Instances or Views), you can use the **Manually select specific items** option to individually select the items to deploy.
   - Whether to analyze all the deployment artifacts fully (**Full**) or only by name (**Partial**), and whether to re-analyze dependencies after anything is changed on the deployment screen. The recommended approach is to use the **Full** option so

that K2 can check, based on artifact names and internal structures, whether you are deploying "new" or "updated" items. If you are a more advanced user and understand the structure of your artifacts well, you can select the **Partial** option since it tends to run faster. The **Automatically re-analyze after each** change option will re-check dependencies if you manually select or deselect items for deployment. Selecting this option can add extra time to the deployment procedure, but might be necessary if you have many shared artifacts or if you are manually selecting items and are not absolutely certain of the dependencies between items.

- Whether to continue deploying even if a preceding deployment item failed.


Select Package

Select the package you wish to deploy

File Name:	C:\Users\Administrator\Documents\DemoDeploymentPackage.kspx
Name:	DemoDeploymentPackage
Description:	

Options

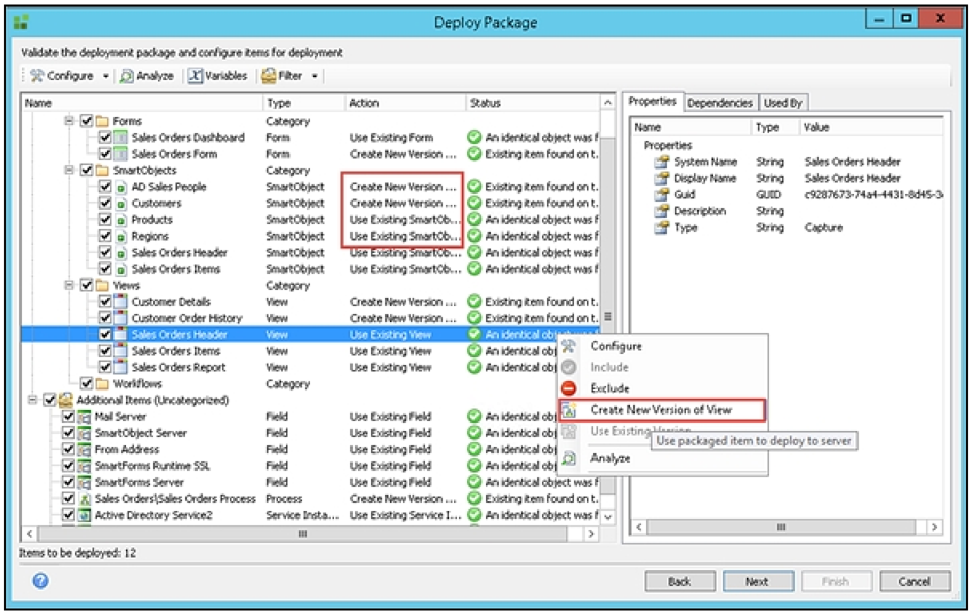
Packaged Items:
- (●) Automatically select all items (recommended)
- ( ) Manually select specific items

Analyze:
- (●) Full: An analysis to determine if the packaged items exist on the server and comparison of their structures (recommended)
- ( ) Partial: An analysis to determine if the packaged items exist on the server
- [✓] Automatically re-analyze after each change

- [✓] Continue deploying packaged items if one or more items cannot be deployed
- [✓] Remember my settings

3. Review and select the items to deploy, and configure where necessary.

You may need to manually select whether to **Use Existing**, **Exclude** or **Create New Version** of items that already exist in the target environment. K2 will attempt to identify existing items. If existing items are found, you will need to tell K2 how to handle the existing items. Remember that overwriting existing items could affect other applications.


Deploy Package

Validate the deployment package and configure items for deployment

Name	Type	Action	Status
Forms	Category		
Sales Orders Dashboard	Form	Use Existing Form	An identical object was f
Sales Orders Form	Form	Create New Version ...	Existing item found on t.
SmartObjects	Category		
AD Sales People	SmartObject	Create New Version ...	Existing item found on t.
Customers	SmartObject	Create New Version ...	Existing item found on t.
Products	SmartObject	Use Existing SmartOb...	An identical object was f
Regions	SmartObject	Use Existing SmartOb...	An identical object was f
Sales Orders Header	SmartObject	Use Existing SmartOb...	An identical object was f
Sales Orders Items	SmartObject	Use Existing SmartOb...	An identical object was f
Views	Category		
Customer Details	View	Create New Version ...	Existing item found on t.
Customer Order History	View	Create New Version ...	Existing item found on t.
Sales Orders Header	View	Use Existing View	An identical obj
Sales Orders Items	View	Use Existing View	An identical obj
Sales Orders Report	View	Use Existing View	An identical obj
Workflows	Category		
Additional Items (Uncategorized)			
Mail Server	Field	Use Existing Field	An identical obj
SmartObject Server	Field	Use Existing Field	An identical obj
From Address	Field	Use Existing Field	An identical obj
SmartForms Runtime SSL	Field	Use Existing Field	An identical obj
SmartForms Server	Field	Use Existing Field	An identical object was f
Sales Orders\Sales Orders Process	Process	Create New Version ...	Existing item found on t.
Active Directory Service2	Service Insta...	Use Existing Service I...	An identical object was f

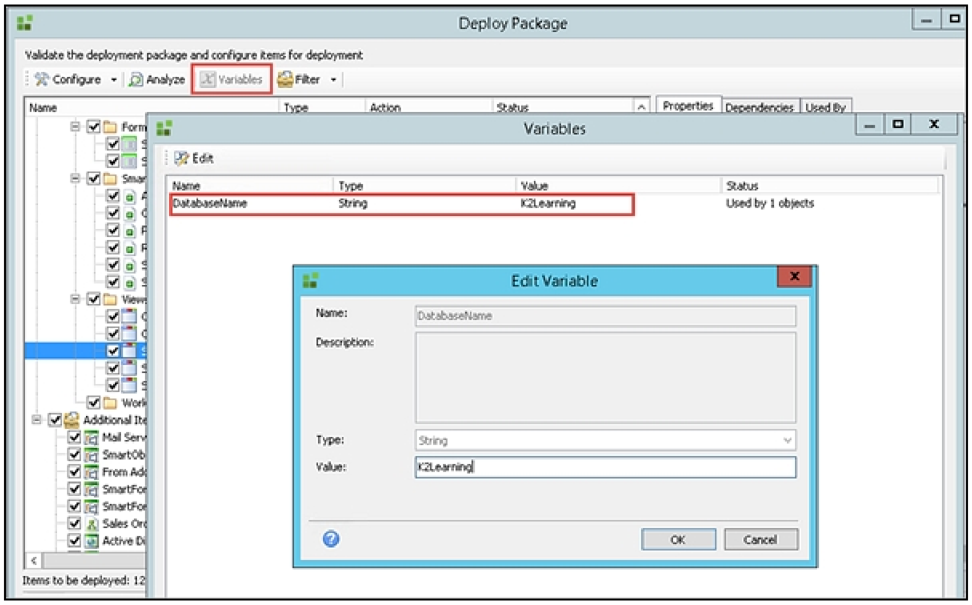
Items to be deployed: 12

Configure
Include
Exclude
Create New Version of View
Use Existing Version
Use packaged item to deploy to server
Analyze

Name	Type	Value
Properties		
System Name	String	Sales Orders Header
Display Name	String	Sales Orders Header
Guid	GUID	c9287673-74a4-4431-8d45-3
Description	String	
Type	String	Capture

4. Provide values for variables (if required).

If the package creator defined variables you need to provide values for those variables.



Validate the deployment package and configure items for deployment

Variables

Name	Type	Value	Status
DatabaseName	String	K2Learning	Used by 1 objects

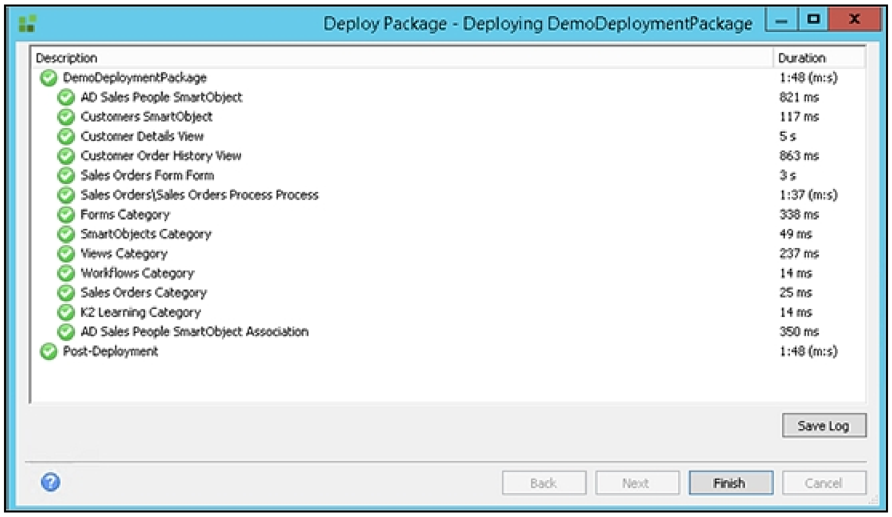
Edit Variable

Name: DatabaseName
Description:
Type: String
Value: K2Learning

Items to be deployed: 12

5. Deploy the package

   Review the status of each item and address errors. If there are errors, sometimes you can just re-deploy the items and the errors will go away. It's worth trying this if something failed during deployment and you are not sure why the deployment failed.



Description	Duration
DemoDeploymentPackage	1:48 (m:s)
AD Sales People SmartObject	821 ms
Customers SmartObject	117 ms
Customer Details View	5 s
Customer Order History View	863 ms
Sales Orders Form Form	3 s
Sales Orders\Sales Orders Process Process	1:37 (m:s)
Forms Category	338 ms
SmartObjects Category	49 ms
Views Category	237 ms
Workflows Category	14 ms
Sales Orders Category	25 ms
K2 Learning Category	14 ms
AD Sales People SmartObject Association	350 ms
Post-Deployment	1:48 (m:s)

6. Complete post-deployment tasks

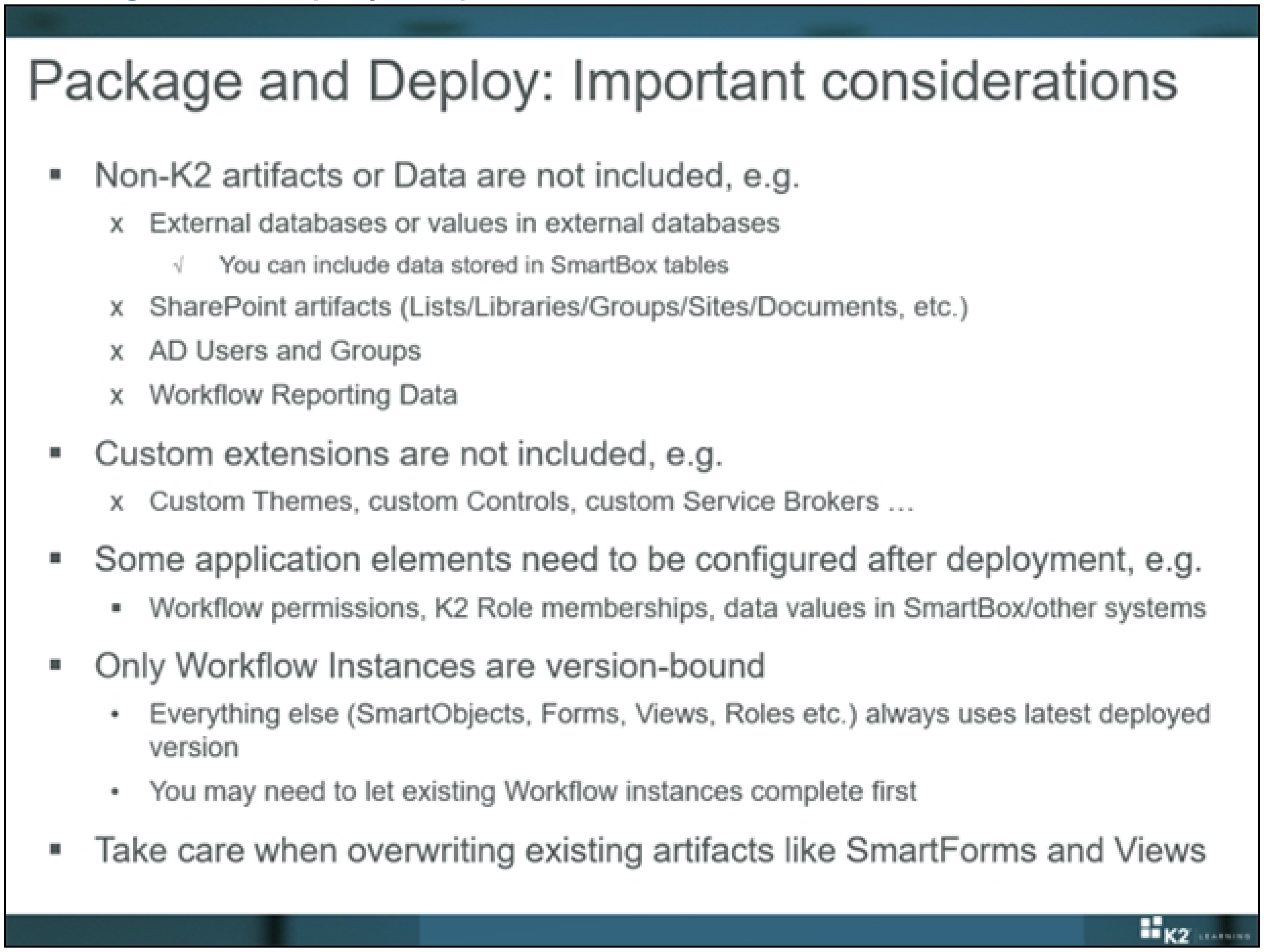
   You may need to set up additional artifacts or items that are not included with the package such as Workflow permissions, Role members, third-party system artifacts, external services, group memberships, permissions, data and so on.

7. Remember to test the application once the deployment has completed.

Summary

- Steps to deploy a deployment package
  1. Launch the Package & Deployment tool and select "Deploy Package"
  2. Browse to the (.kspx) package and set the deployment options
  3. Review and select the items to deploy, configure where necessary
  4. Provide values for variables (if required)
  5. Deploy the package
  6. Perform additional post-deployment tasks and deploy any non-included items
  7. Test the application

Package and Deploy: Important considerations



Package and Deploy: Important considerations

- Non-K2 artifacts or Data are not included, e.g.
  - x External databases or values in external databases
    - √ You can include data stored in SmartBox tables
  - x SharePoint artifacts (Lists/Libraries/Groups/Sites/Documents, etc.)
  - x AD Users and Groups
  - x Workflow Reporting Data
- Custom extensions are not included, e.g.
  - x Custom Themes, custom Controls, custom Service Brokers …
- Some application elements need to be configured after deployment, e.g.
  - Workflow permissions, K2 Role memberships, data values in SmartBox/other systems
- Only Workflow Instances are version-bound
  - Everything else (SmartObjects, Forms, Views, Roles etc.) always uses latest deployed version
  - You may need to let existing Workflow instances complete first
- Take care when overwriting existing artifacts like SmartForms and Views

Video

There are some important considerations regarding Package and Deployment of SmartForms applications.

Included and excluded artifacts

Deployment packages only include certain K2 components such as:

- Forms
- Views
- SmartObjects
- Deployed K2 Workflows
- If selected, Data that is stored in K2 SmartBox SmartObjects
- Supporting artifacts such as
  - Categories
  - Roles (but not Role memberships)
  - Service Instances
  - Notification events
  - Environment Library fields

If an item is not specified in the list above, assume that it will not be included in the deployment package. Common items that are not included in deployment packages (which you will need to deploy or configure manually) include:

- External databases or data from external databases (the exception is that you can select to include data from your SmartBox SmartObjects as part of the deployment package creation).
- SharePoint artifacts like Lists, Libraries, List Items, Sites
- Active Directory artifacts like users and groups
- Workflow reporting data
- Customizations are not included in packages, such as
  - Custom Themes
  - Custom Controls

    - Custom Service Brokers
    - Custom notification providers
    - Custom user managers

Post-installation configuration of K2 items

Some K2 elements must be configured after deployment, including:

- Workflow permissions
- SmartBox SmartObject permissions
- K2 Role memberships
- Data values in SmartBox tables or other third-party systems
- Permissions for access to other providers

Versioning

Remember that Workflow Instances are version-bound and will not be upgraded to the newer version of a deployed Workflow, but that Forms, Views and SmartObjects always use the latest version. Therefore, you may need to allow existing Workflow Instances complete before you deploy new items, depending on what changes were made.

Similarly, remember that other applications may re-use SmartObjects, Views or even Forms that you are deploying, so deploying new versions could affect those applications. As a general rule, it is safe to add things to artifacts, but not to remove things. For example, adding a SmartObject property should generally be safe, but removing an existing property might not be safe and could break other applications.

MASTERY CHECKPOINT: Packaging and Deploying SmartForm applications

Packaging and Deploying SmartForm applications

- How to create and deploy SmartForm applications
- Considerations when creating and deploying packages
  - Overwriting existing artifacts upon deployment
- Versioning implications and considerations

MASTERY CHECKPOINT

Video

This is a checkpoint for the information covered in Part 7 of this module. If you are attending instructor-led training, use this as an opportunity to answer any questions around packaging and deploying SmartForm applications.

After completing Part 7, you should know:

- How to create and deploy packages using the K2 Package and Deployment tool
- Considerations when creating packages
  - Including items, manually selecting additional items, using variables
- Considerations when deploying packages
  - Manually selecting items, configuring whether to overwrite, re-use or exclude items
  - Setting variables
- Implications of how versioning in K2 works and how it affects existing Workflow Instances

Knowledge-check questions

Q: True or false: when you deploy a package that contains a Workflow definition, all existing Workflow Instances are upgraded to the new definition.
Reveal answer**A:** False, existing Workflow Instances continue to run on their definition of the Workflow.

Q: True or false: Workflows will use the versions of Forms and SmartObjects that existed when that Workflow was deployed, and are bound to those versions.
Reveal answer**A:** False, Workflows will always use the latest version of Forms, Views or SmartObjects.

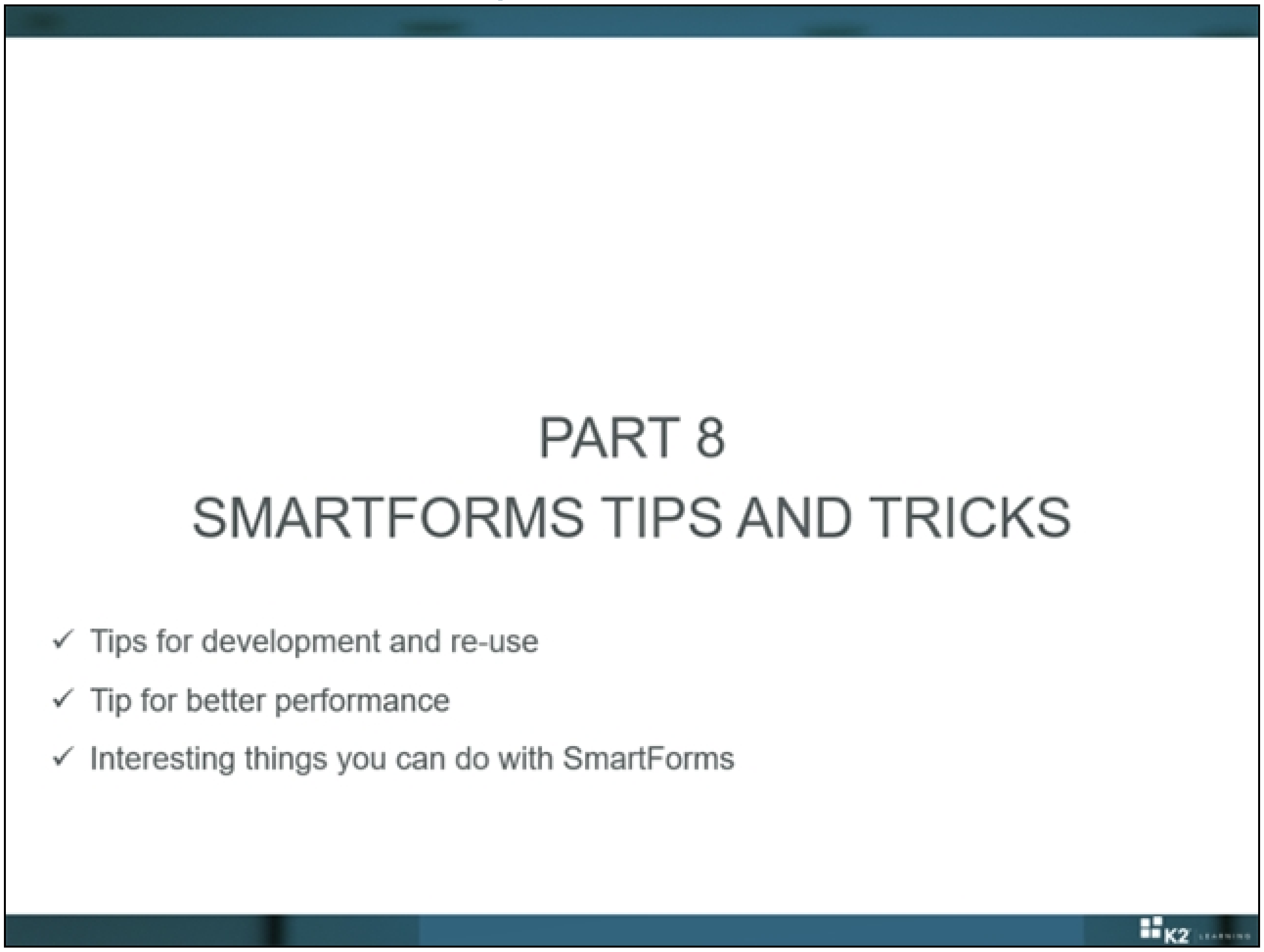
Q: True or False: you can manually revert to a previous version of a Form or View using K2 Designer.
Reveal answer**A:** False, there is no UI to do so, although you can run DB scripts (contact K2 support) to revert to a previous version of a Workflow.

Q: What happens when you deploy a package that contains items that already exist on the target environment?
Reveal answer**A:** You need to select whether to use the existing versions or overwrite the existing versions. K2 will try

to do a best-guess to determine what should happen, but you should always double-check what will happen to the existing versions.

Q: What happens when you deploy a package that contains a SmartObject, View or Form that is used by another applications?
Reveal answer**A:** A The other application will use the deployed version of the SmartObject, View or Form, so you need to be careful that the new version does not break that other application.

PART 8: SmartForms Tips and Tricks



PART 8

SMARTFORMS TIPS AND TRICKS

- ✓ Tips for development and re-use
- ✓ Tip for better performance
- ✓ Interesting things you can do with SmartForms

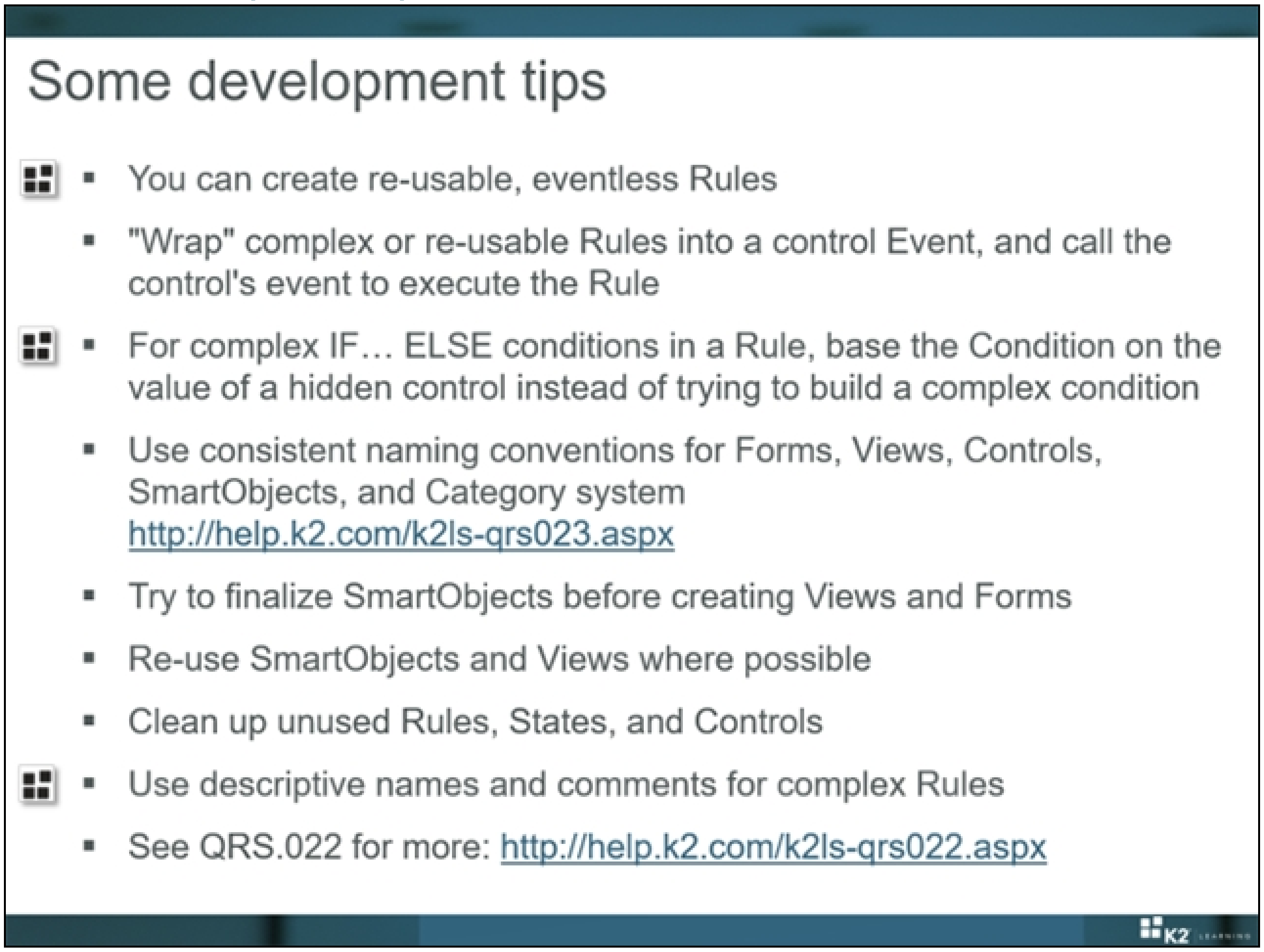
Video

This is the last part of this learning module where we cover some tips and tricks for building SmartForms, including:

- Best practices for ensuring well-performing SmartForms
- How to re-use Rules
- Some interesting things you can do with SmartForms

There is no lab exercise at the end of this module. Since there is only limited time for training, we cannot dive into each tip and trick in detail with hands-on exercises. You can refer to the SmartForms product documentation to learn how to work with specific Controls, as well as some of the how-to scenarios.

Some development tips



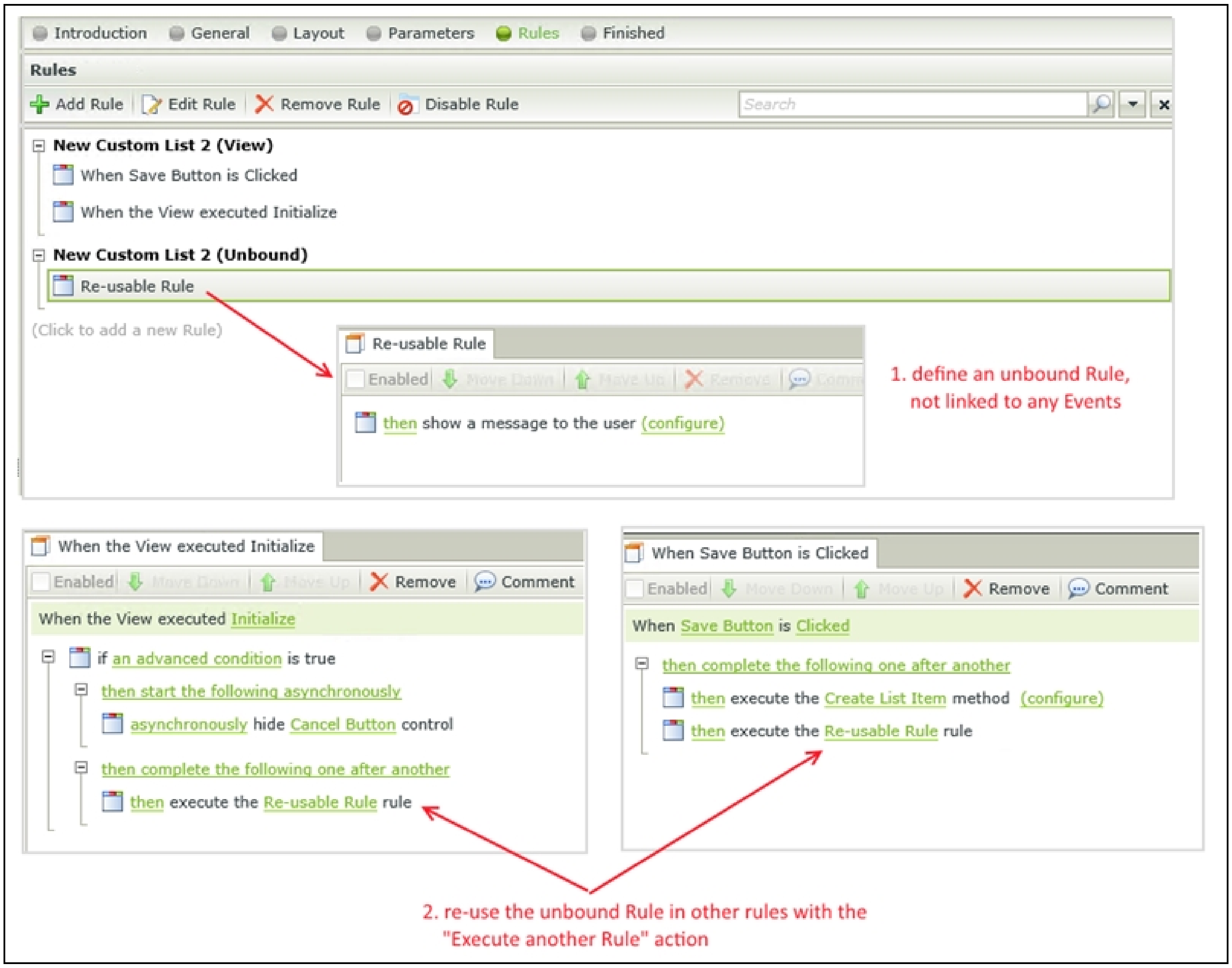
Some development tips

- You can create re-usable, eventless Rules
- "Wrap" complex or re-usable Rules into a control Event, and call the control's event to execute the Rule
- For complex IF… ELSE conditions in a Rule, base the Condition on the value of a hidden control instead of trying to build a complex condition
- Use consistent naming conventions for Forms, Views, Controls, SmartObjects, and Category system http://help.k2.com/k2ls-qrs023.aspx
- Try to finalize SmartObjects before creating Views and Forms
- Re-use SmartObjects and Views where possible
- Clean up unused Rules, States, and Controls
- Use descriptive names and comments for complex Rules
- See QRS.022 for more: http://help.k2.com/k2ls-qrs022.aspx

Video

This topic explains some development tips and tricks that will help you when building SmartForms.

- Use eventless Rules to implement re-usable Rules. That way, you only need to maintain the Rule setup in one place and then use the Execute Another Rule Action to call the eventless Rule from other Rules.



New Custom List 2 (View)

When Save Button is Clicked

When the View executed Initialize

New Custom List 2 (Unbound)

Re-usable Rule

(Click to add a new Rule)

Re-usable Rule

then show a message to the user (configure)

1. define an unbound Rule, not linked to any Events

When the View executed Initialize

When the View executed Initialize

if an advanced condition is true

then start the following asynchronously

asynchronously hide Cancel Button control

then complete the following one after another

then execute the Re-usable Rule rule

When Save Button is Clicked

When Save Button is Clicked

then complete the following one after another

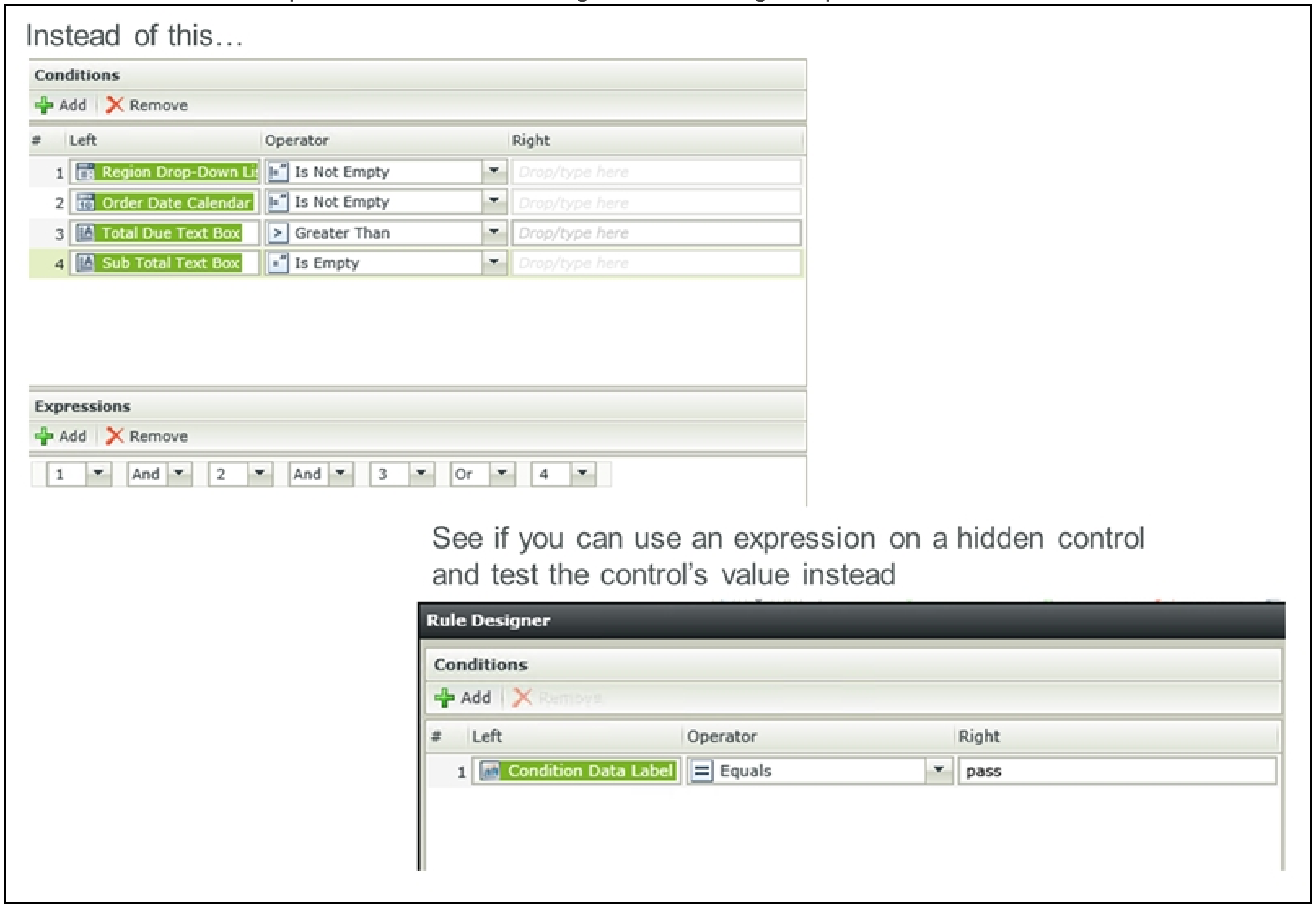
then execute the Create List Item method (configure)

then execute the Re-usable Rule rule

2. re-use the unbound Rule in other rules with the "Execute another Rule" action

- "Wrap" complex or re-usable Rules into a Control Event, and call the Control's Event to execute the Rule (i.e. use the “call a control's method” Action which fires the Rule, instead of re-implementing the same Rule for another Control).
- For complex IF…ELSE conditions in a Rule, base the condition on the value of a hidden Control populated by an expression, instead of trying to build a complex condition. This is not always possible but if you can do this it is

easier to maintain the expression than maintaining and evaluating complex if-then-and conditions.



Instead of this…

Conditions

#	Left	Operator	Right
1	Region Drop-Down Li	Is Not Empty	Drop/type here
2	Order Date Calendar	Is Not Empty	Drop/type here
3	Total Due Text Box	Greater Than	Drop/type here
4	Sub Total Text Box	Is Empty	Drop/type here

Expressions

1 And 2 And 3 Or 4

See if you can use an expression on a hidden control and test the control's value instead

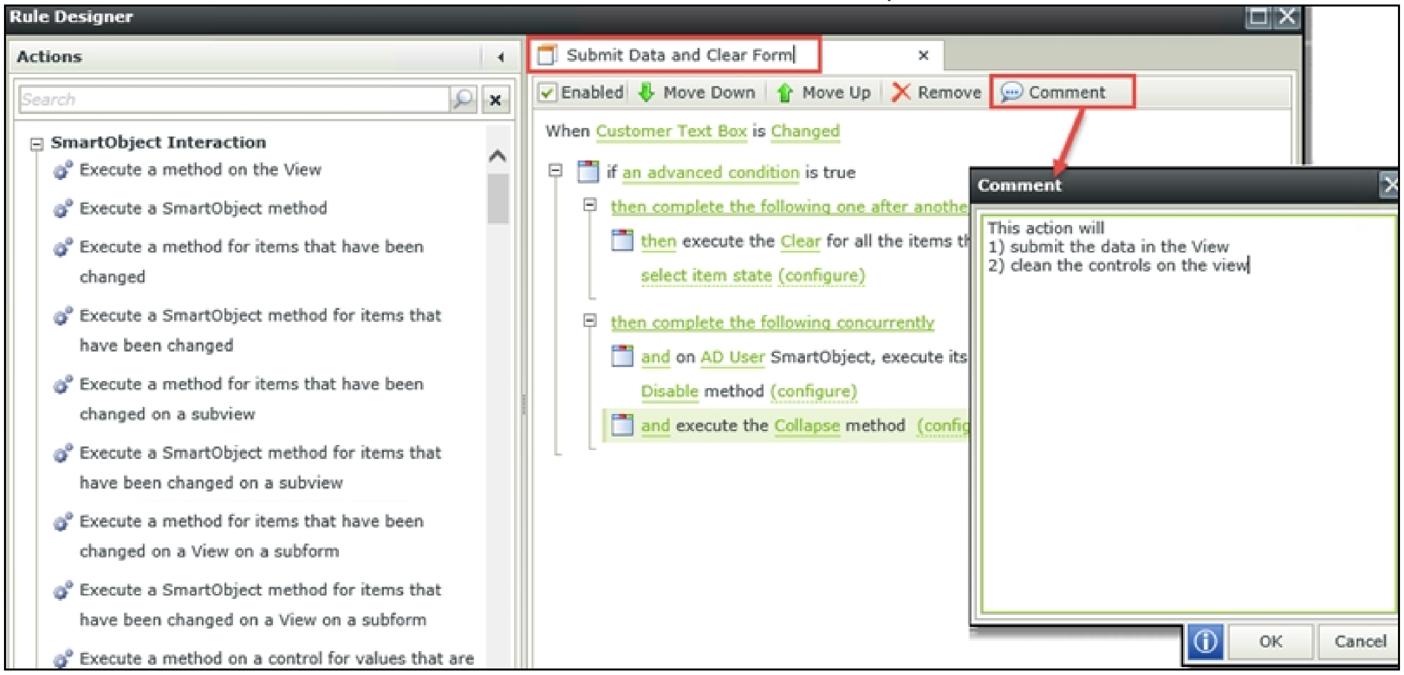
Rule Designer

Conditions

#	Left	Operator	Right
1	Condition Data Label	Equals	pass

- Use consistent naming conventions for Forms, Views, Controls and Data and use a logical and consistent convention for the Category Browser (System). See the Quick-Reference Guide [Naming Conventions for K2 Artifacts (http://help.k2.com/k2ls-qrs023.aspx)](http://help.k2.com/k2ls-qrs023.aspx) for recommendations and ideas on implementing a naming convention in your organization.
- Try to finalize the design and properties of SmartObjects before creating Views, Forms and Workflows. If you add or remove Properties/Methods from a SmartObject, you may have to do a lot of manual re-binding.
- Re-use SmartObjects and Views where possible. It is easier to maintain fewer items, but remember the impact of deploying new versions of SmartObjects/Views/Forms that are used by other applications.
- Clean-up unused Rules, States and Controls. Unused items in Views and Forms can an effect on the performance of the Views and Forms, and make maintenance unnecessarily complex.

- Use **Rule names** and **Rule Comments** to describe and document complex Rules.


Rule Designer
Actions
Search
SmartObject Interaction
Execute a method on the View
Execute a SmartObject method
Execute a method for items that have been changed
Execute a SmartObject method for items that have been changed
Execute a method for items that have been changed on a subview
Execute a SmartObject method for items that have been changed on a subview
Execute a method for items that have been changed on a View on a subform
Execute a SmartObject method for items that have been changed on a View on a subform
Execute a method on a control for values that are
Submit Data and Clear Form
Enabled
Move Down
Move Up
Remove
Comment
OK
Cancel

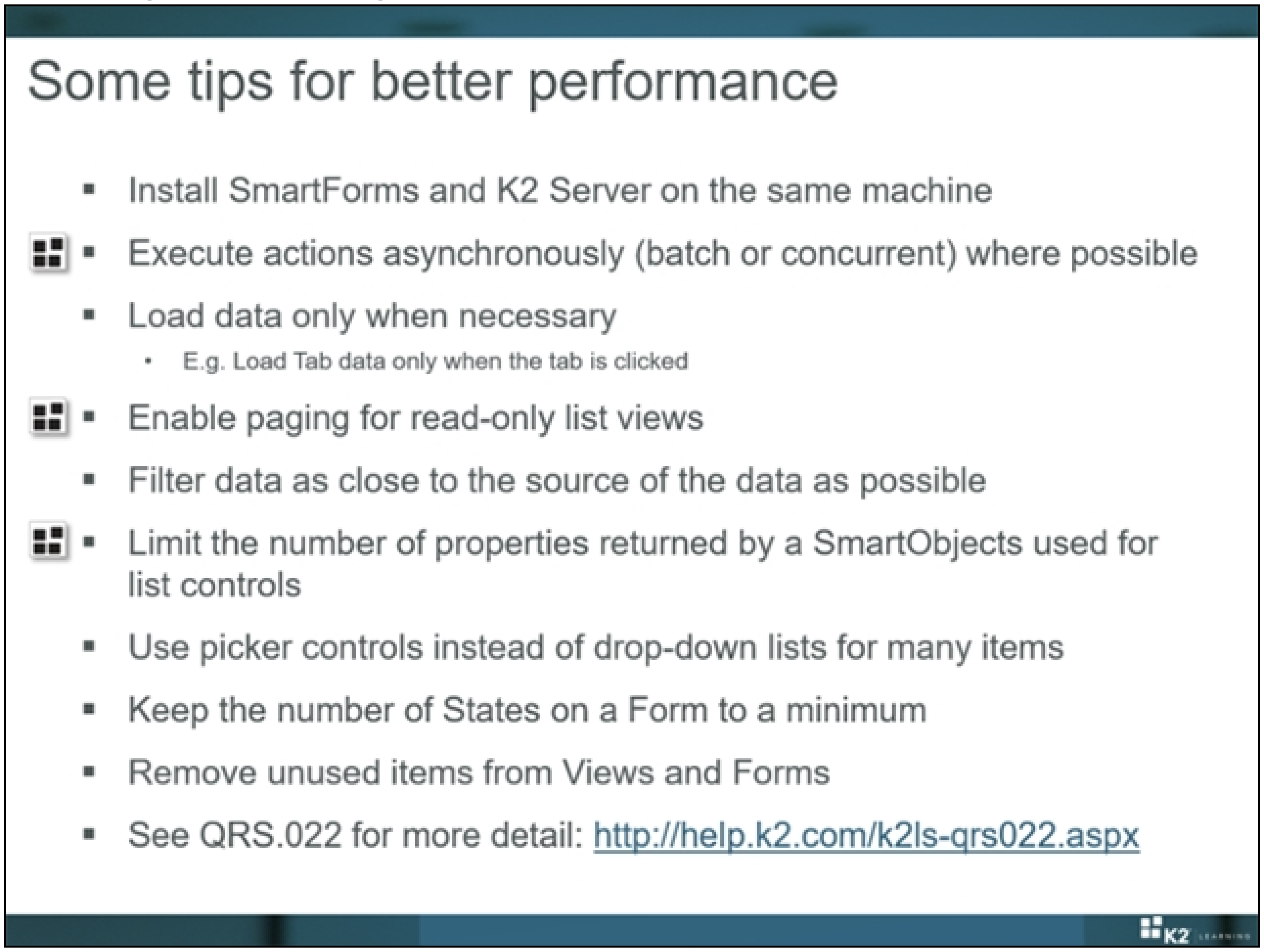
When Customer Text Box is Changed
if an advanced condition is true
then complete the following one after anothe
then execute the Clear for all the items th
select item state (configure)
then complete the following concurrently
and on AD User SmartObject, execute its
Disable method (configure)
and execute the Collapse method (config

Comment

This action will
1) submit the data in the View
2) clean the controls on the view

- See the Quick-Reference Guide K2 smartforms implementation tips (http://help.k2.com/k2ls-qrs022.aspx) for more suggestions.

Some tips for better performance



Some tips for better performance

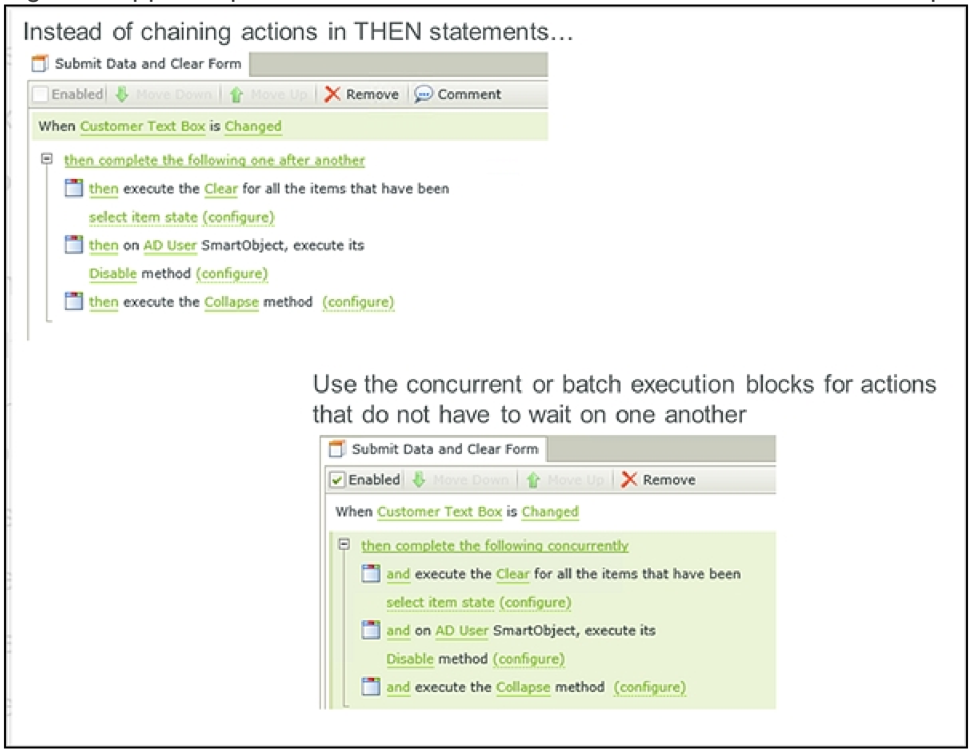
- Install SmartForms and K2 Server on the same machine
- Execute actions asynchronously (batch or concurrent) where possible
- Load data only when necessary
  - E.g. Load Tab data only when the tab is clicked
- Enable paging for read-only list views
- Filter data as close to the source of the data as possible
- Limit the number of properties returned by a SmartObjects used for list controls
- Use picker controls instead of drop-down lists for many items
- Keep the number of States on a Form to a minimum
- Remove unused items from Views and Forms
- See QRS.022 for more detail: http://help.k2.com/k2ls-qrs022.aspx

Video

This topic explains some performance tweaking tips and tricks that will help you when building SmartForms.

- Install the SmartForms and K2 Servers on the same machine. That way, all communication stays local and no additional network overhead is added. In addition, you can define a Hosts file entry to force all communication to remain local (see http://help.k2.com/kb001678# for more information).

- Execute Actions asynchronously (batch or concurrent) where possible so that things like independent data loading can happen in parallel and the Actions do not wait for each other to complete.



Instead of chaining actions in THEN statements...

Submit Data and Clear Form

Enabled | Move Down | Move Up | Remove | Comment

When Customer Text Box is Changed

then complete the following one after another
- then execute the Clear for all the items that have been select item state (configure)
- then on AD User SmartObject, execute its Disable method (configure)
- then execute the Collapse method (configure)

Use the concurrent or batch execution blocks for actions that do not have to wait on one another

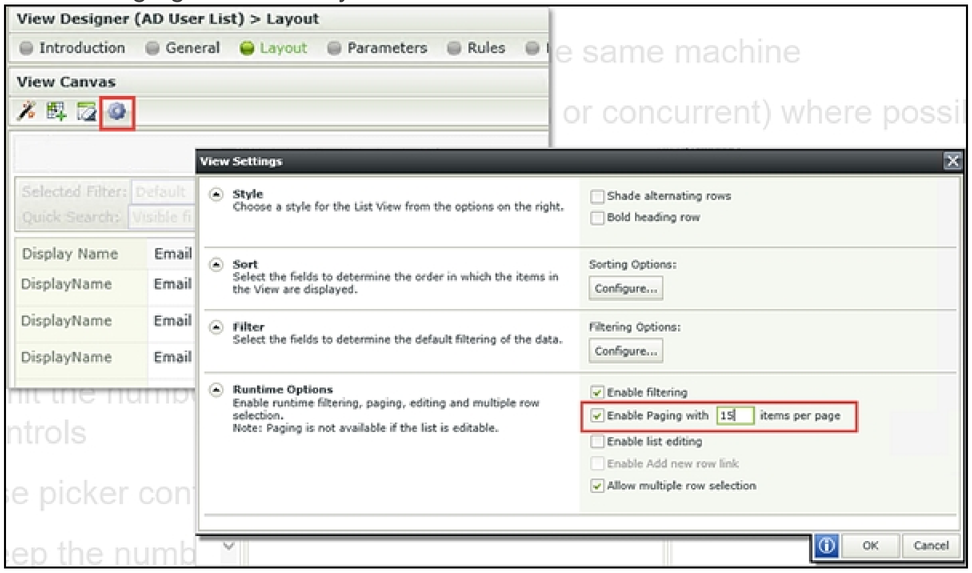
Submit Data and Clear Form

☑ Enabled | Move Down | Move Up | Remove

When Customer Text Box is Changed

then complete the following concurrently
- and execute the Clear for all the items that have been select item state (configure)
- and on AD User SmartObject, execute its Disable method (configure)
- and execute the Collapse method (configure)

- Load data only when necessary. For example, load List Views on a Tab only when the Tab is clicked, or delay loading of List Views until the data is actually needed.
- Enable Paging for read-only List Views.



View Designer (AD User List) > Layout

Introduction | General | Layout | Parameters | Rules

View Canvas

Selected Filter: Default
Quick Search: Visible fi

Display Name	Email
DisplayName	Email
DisplayName	Email
DisplayName	Email

View Settings

Style
Choose a style for the List View from the options on the right.
- ☐ Shade alternating rows
- ☐ Bold heading row

Sort
Select the fields to determine the order in which the items in the View are displayed.
Sorting Options: Configure...

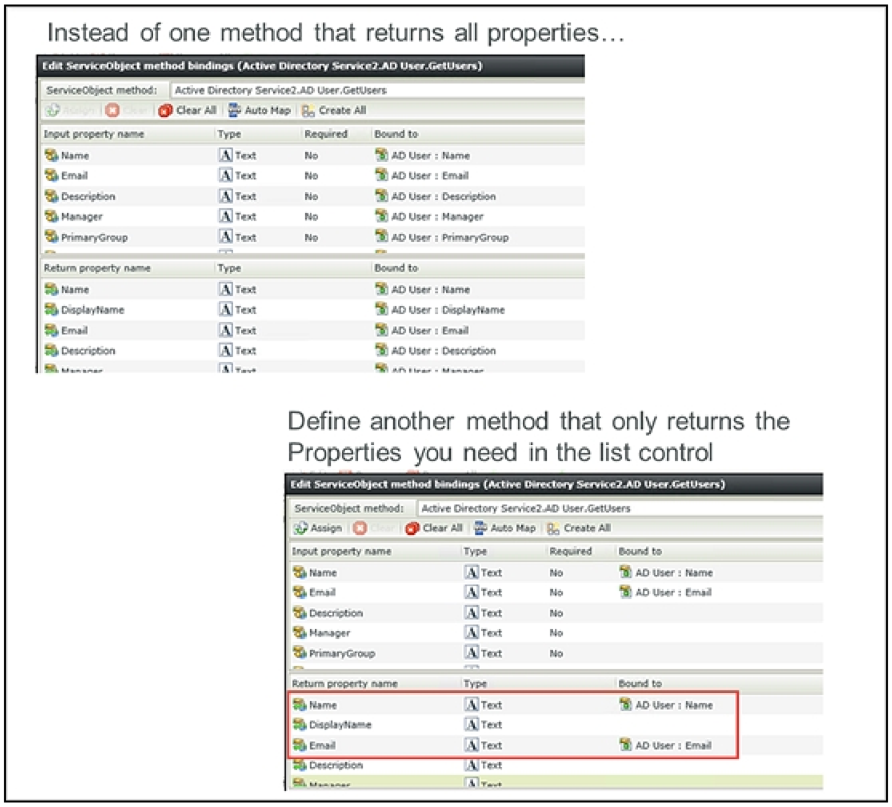
Filter
Select the fields to determine the default filtering of the data.
Filtering Options: Configure...

Runtime Options
Enable runtime filtering, paging, editing and multiple row selection.
Note: Paging is not available if the list is editable.
- ☑ Enable filtering
- ☑ Enable Paging with 15 items per page
- ☐ Enable list editing
- ☐ Enable Add new row link
- ☑ Allow multiple row selection

OK | Cancel

- Filter data as close to the source of the data as possible. The closer to the source of the data that you filter the data, the better performance will be. You can use things like stored procedures, views or SmartObject methods with hard-coded parameters to filter data.
- Limit the number of properties returned by SmartObjects used for list Controls. For example, if a SmartObject has ten properties but you only need two of those in a drop-down list, create a separate method for that SmartObject that only retrieves those two properties you need. That way you are not needlessly returning unused data to

the Form.


Instead of one method that returns all properties…

Edit ServiceObject method bindings (Active Directory Service2.AD User.GetUsers)

ServiceObject method: Active Directory Service2.AD User.GetUsers

Assign | Clear | Clear All | Auto Map | Create All

Input property name	Type	Required	Bound to
Name	Text	No	AD User : Name
Email	Text	No	AD User : Email
Description	Text	No	AD User : Description
Manager	Text	No	AD User : Manager
PrimaryGroup	Text	No	AD User : PrimaryGroup

Return property name	Type	Bound to
Name	Text	AD User : Name
DisplayName	Text	AD User : DisplayName
Email	Text	AD User : Email
Description	Text	AD User : Description
Manager	Text	AD User : Manager

Define another method that only returns the Properties you need in the list control

Edit ServiceObject method bindings (Active Directory Service2.AD User.GetUsers)

ServiceObject method: Active Directory Service2.AD User.GetUsers

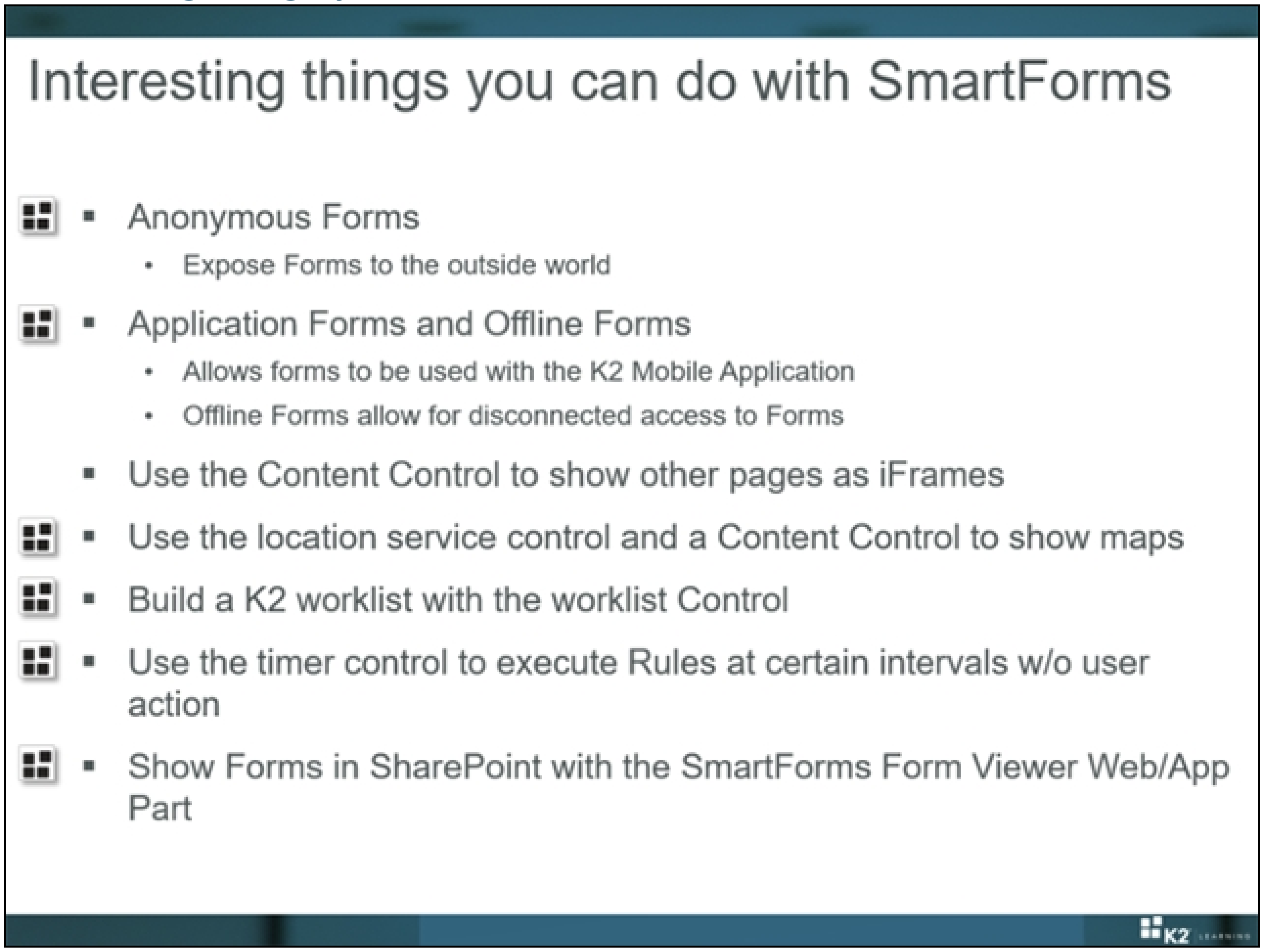
Assign | Clear | Clear All | Auto Map | Create All

Input property name	Type	Required	Bound to
Name	Text	No	AD User : Name
Email	Text	No	AD User : Email
Description	Text	No	
Manager	Text	No	
PrimaryGroup	Text	No	

Return property name	Type	Bound to
Name	Text	AD User : Name
DisplayName	Text	
Email	Text	AD User : Email
Description	Text	
Manager	Text	

- Use Picker or AutoComplete Controls instead of drop-down lists for data sources returning many items. Drop-down lists get very cumbersome for users when there are more than 20 items.
- Keep the number of States on a Form to a minimum. Every State duplicates all the Rules on a Form, so the fewer States there are, the fewer duplication of Rules there are.
- Remove things like unused Controls, unused States and unused Views.
- See the Quick-Reference Guide [K2 smartforms implementation tips (http://help.k2.com/k2ls-qrs022.aspx)](http://help.k2.com/k2ls-qrs022.aspx) for more suggestions and details on how to implement some of these performance tweaks.

Interesting things you can do with SmartForms



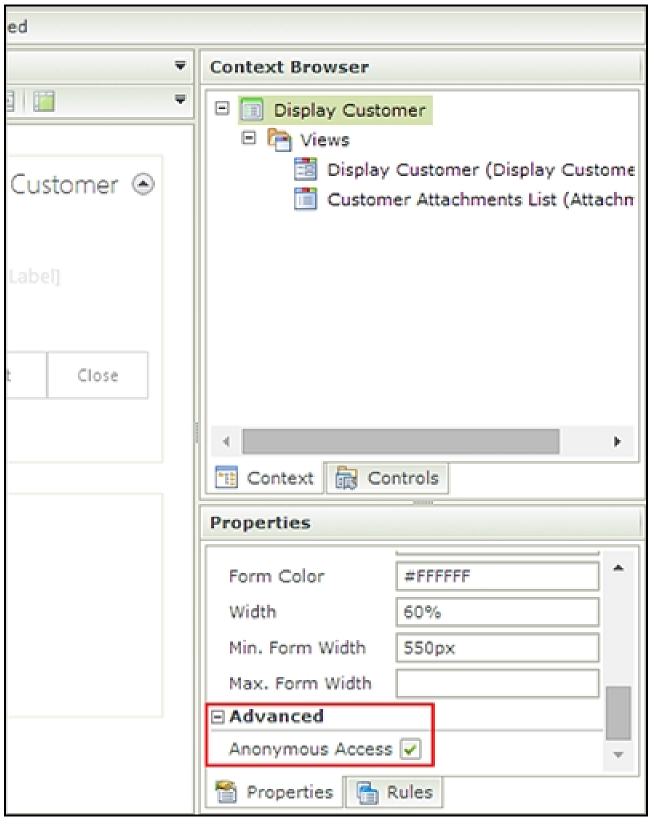
Interesting things you can do with SmartForms

- Anonymous Forms
  - Expose Forms to the outside world
- Application Forms and Offline Forms
  - Allows forms to be used with the K2 Mobile Application
  - Offline Forms allow for disconnected access to Forms
- Use the Content Control to show other pages as iFrames
- Use the location service control and a Content Control to show maps
- Build a K2 worklist with the worklist Control
- Use the timer control to execute Rules at certain intervals w/o user action
- Show Forms in SharePoint with the SmartForms Form Viewer Web/App Part

Video

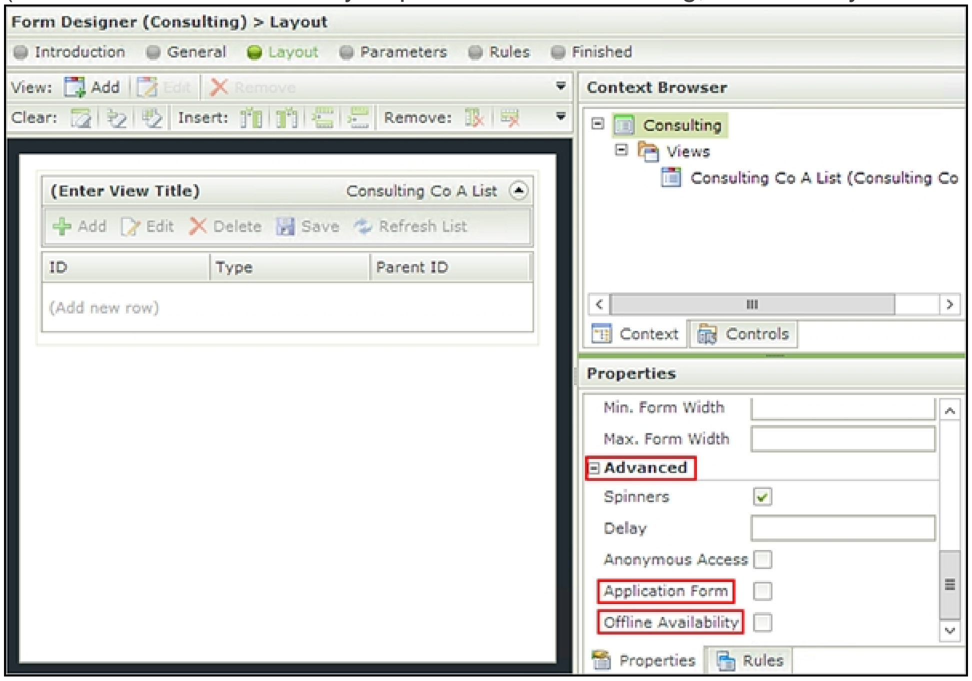
This topic lists some (but by no means all) of the interesting things you can do with SmartForms. If you would like to know more, see the K2 smartforms User Guide at http://help.k2.-com/onlinehelp/K2smartforms/UserGuide/current/default.htm

- Anonymous Forms: expose specific Forms to the outside world without requiring the user to log in. In these cases, K2 usually uses the application pool identify to authenticate with the K2 Server, but of course those anonymous users need to be able to access the Form URL. An alternative option is to create an anonymous access Runtime Website and expose Forms to users that way. See the Anonymous Views/Forms topic in SmartForms Help for more information.



Form Color	#FFFFFF
Width	60%
Min. Form Width	550px
Max. Form Width	
Advanced	
Anonymous Access	✓

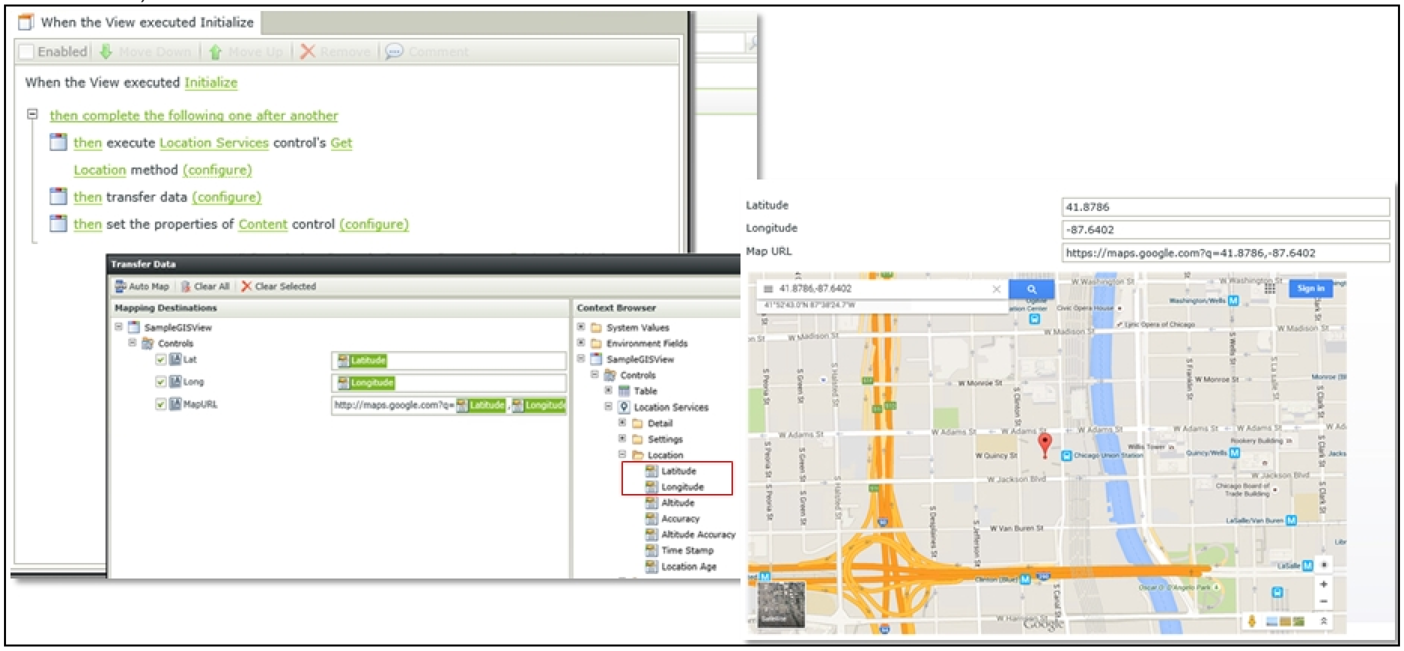
- Application Forms and Offline Forms. The Application Form option allows Forms to render natively in the K2 mobile App. When this option is selected, you can also enable the Offline Availability switch, which will allow users to complete and submit the Form while they are disconnected (with some limitations). Behind the scenes, the K2 mobile App caches the Form and submitted data, and submits any completed Forms as a batch when the user is back online. See Offline and Application Forms topic in the SmartForms User Guide for more information. (Note that offline Forms may require additional licensing, check with your local K2 office for details.)



Min. Form Width	
Max. Form Width	
Advanced	
Spinners	✓
Delay	
Anonymous Access	
Application Form	
Offline Availability	

- Use the Content Control to show other webpages inside a Form as iFrames. (Note: not all web pages support loading in iFrames.)
- Use the Location Services Control and a Content Control to show maps. You can use the Location Services Control's latitude and longitude properties to determine the user's location (or load Lat/Long data from a SmartObject), then construct an iFrame URL or a query to a mapping provider to show a map of the user's location. (Note that you require additional developer licensing from the mapping provider to construct queries and show data in

an iFrame.)



When the View executed Initialize

Enabled | Move Down | Move Up | Remove | Comment

When the View executed Initialize

then complete the following one after another

then execute Location Services control's Get Location method (configure)

then transfer data (configure)

then set the properties of Content control (configure)

Transfer Data

Auto Map | Clear All | Clear Selected

Mapping Destinations

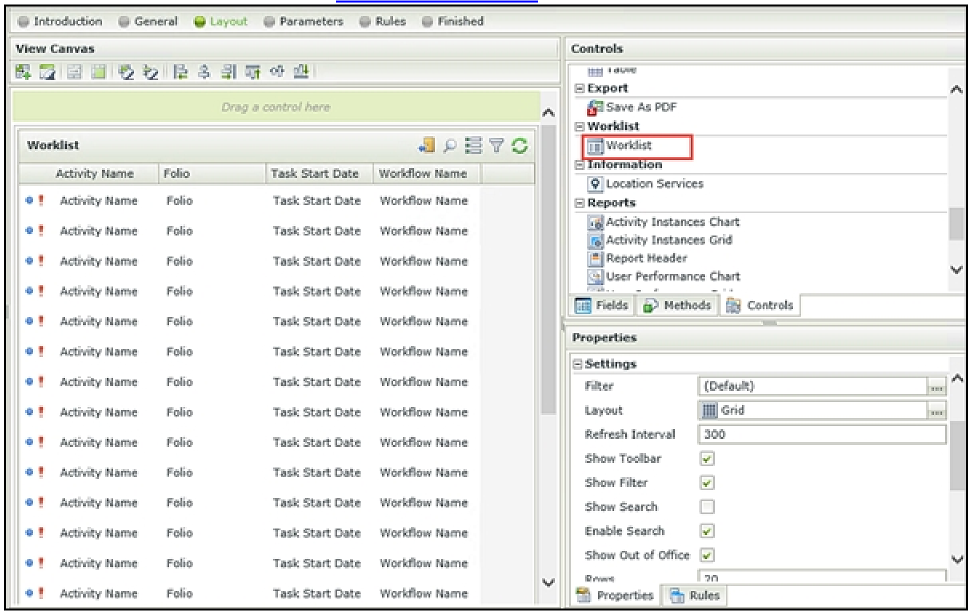
SampleGISView	
Controls	
Lat	Latitude
Long	Longitude
MapURL	http://maps.google.com?q= Latitude , Longitude

Context Browser

System Values
Environment Fields
SampleGISView
Controls
Table
Location Services
Detail
Settings
Location
Latitude
Longitude
Altitude
Accuracy
Altitude Accuracy
Time Stamp
Location Age

Latitude	41.8786
Longitude	-87.6402
Map URL	https://maps.google.com?q=41.8786,-87.6402

- Build a K2 worklist with the Worklist Control to show K2 tasks



Introduction | General | Layout | Parameters | Rules | Finished

View Canvas

Drag a control here

Worklist

Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name
Activity Name	Folio	Task Start Date	Workflow Name

Controls

Export
Save As PDF
Worklist
Worklist
Information
Location Services
Reports
Activity Instances Chart
Activity Instances Grid
Report Header
User Performance Chart

Fields | Methods | Controls

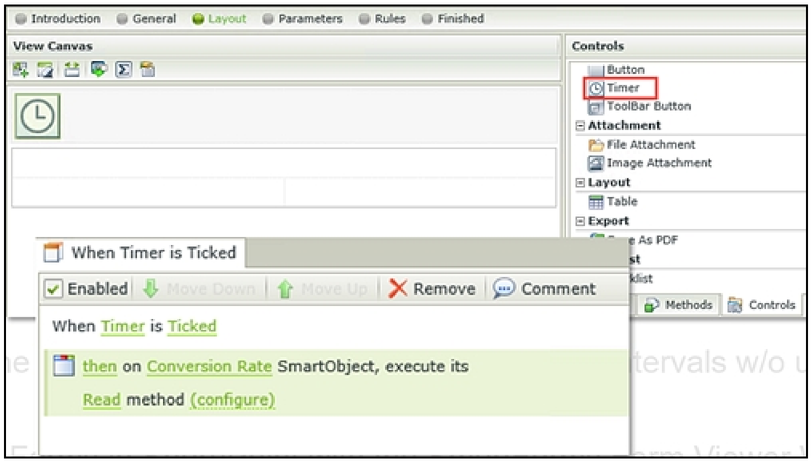
Properties

Settings

Filter	(Default)
Layout	Grid
Refresh Interval	300
Show Toolbar	✓
Show Filter	✓
Show Search	
Enable Search	✓
Show Out of Office	✓
Rows	20

Properties | Rules

- Use the Timer Control to execute Rules at certain intervals without user action. For example, automatically refresh a Control's value at certain intervals.



Introduction | General | Layout | Parameters | Rules | Finished

View Canvas

Controls

Button
Timer
ToolBar Button
Attachment
File Attachment
Image Attachment
Layout
Table
Export
Save As PDF

Methods | Controls

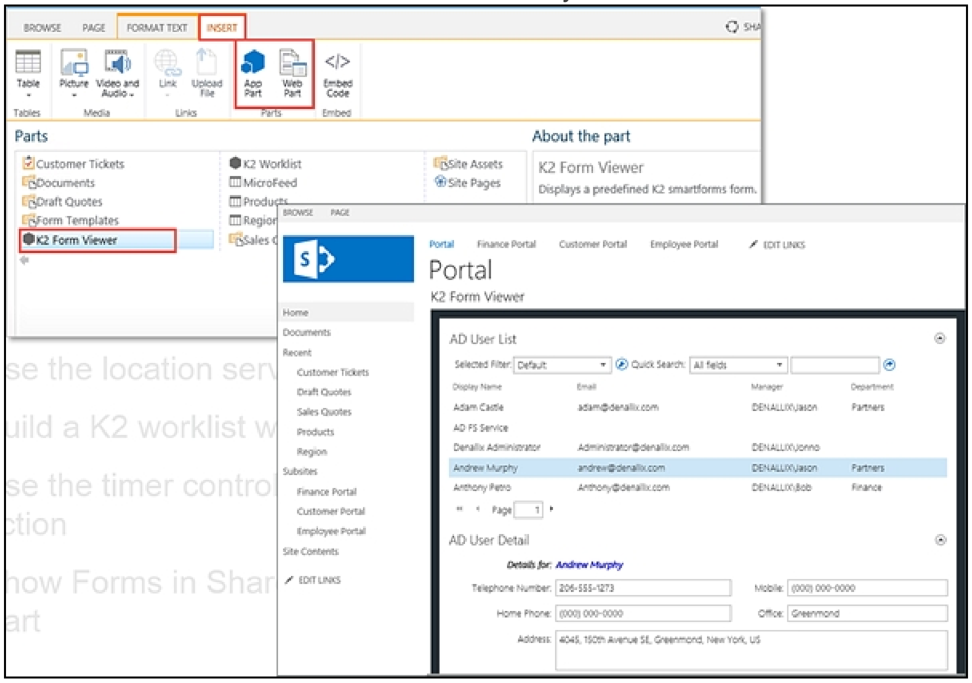
When Timer is Ticked

Enabled | Move Down | Move Up | Remove | Comment

When Timer is Ticked

then on Conversion Rate SmartObject, execute its Read method (configure)

- Show Forms in SharePoint with the SmartForms Form Viewer Web/App Part. You can even use the SharePoint Themes to make the Forms look SharePoint-y.



K2 Form Viewer

Displays a predefined K2 smartforms form.

Portal

K2 Form Viewer

AD User List

Selected Filter: Default Quick Search: All fields

Display Name	Email	Manager	Department
Adam Castle	adam@denallix.com	DENALLIX\Jason	Partners
AD FS Service			
Denallix Administrator	Administrator@denallix.com	DENALLIX\Jonno	
Andrew Murphy	andrew@denallix.com	DENALLIX\Jason	Partners
Anthony Petro	Anthony@denallix.com	DENALLIX\Bob	Finance

Page 1

AD User Detail

Details for: *Andrew Murphy*

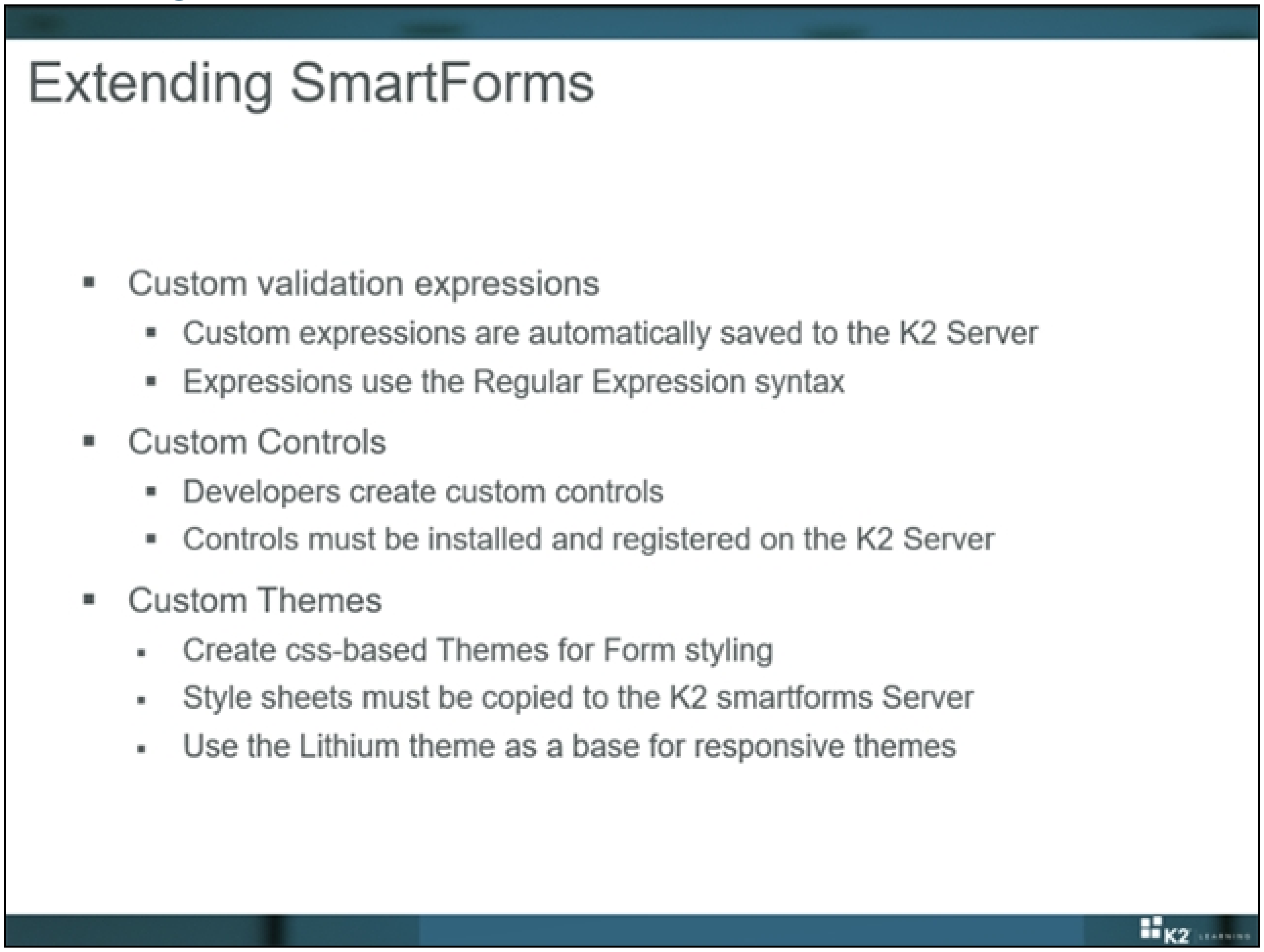
Telephone Number: 206-555-1273 Mobile: (000) 000-0000

Home Phone: (000) 000-0000 Office: Greenmond

Address: 4045, 150th Avenue SE, Greenmond, New York, US

These are just some of the things that are possible with SmartForms, but there is much more of course. See the SmartForms User Guide and articles on http://help.k2.com and the K2 Community site at http://community.k2.com for more ideas, how-to's and inspiration.

Extending SmartForms



Extending SmartForms

- Custom validation expressions
  - Custom expressions are automatically saved to the K2 Server
  - Expressions use the Regular Expression syntax
- Custom Controls
  - Developers create custom controls
  - Controls must be installed and registered on the K2 Server
- Custom Themes
  - Create css-based Themes for Form styling
  - Style sheets must be copied to the K2 smartforms Server
  - Use the Lithium theme as a base for responsive themes

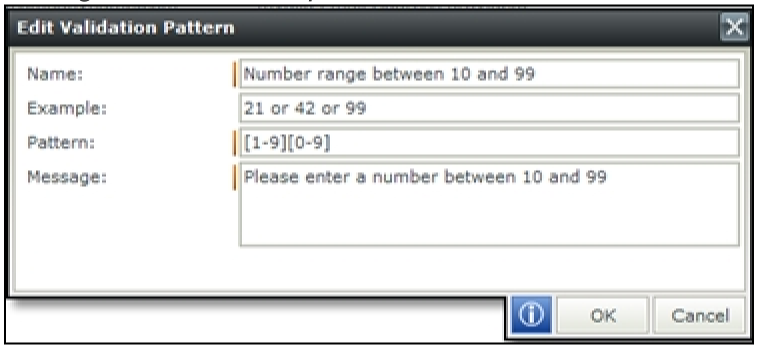
Video

You can extend SmartForms with custom validation expressions, custom Controls and custom themes. To learn more about any of these extension points (or others that may have been subsequently added), see the K2 smartforms Developer Reference on help.k2.com.

Custom validation expressions

The first (and simplest) extension point is to define custom expressions for use in validation rules. You probably recall that SmartForm validation is based on Regular Expressions. When you access the validation pattern for a Control on a View, you can use the Add Pattern button to define a new expression, which will then be available to other users.

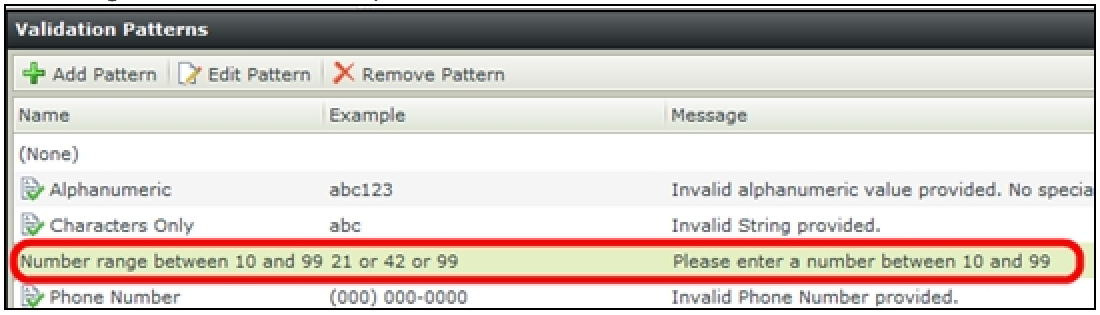
Adding a new validation pattern



Edit Validation Pattern

Name:	Number range between 10 and 99
Example:	21 or 42 or 99
Pattern:	[1-9][0-9]
Message:	Please enter a number between 10 and 99

Selecting a custom validation pattern



Name	Example	Message
(None)		
Alphanumeric	abc123	Invalid alphanumeric value provided. No specia
Characters Only	abc	Invalid String provided.
Number range between 10 and 99	21 or 42 or 99	Please enter a number between 10 and 99
Phone Number	(000) 000-0000	Invalid Phone Number provided.

Tip
There are many useful tools and websites that will help you create more advanced Regular Expressions. One of the best tools we've found to build and test Regular Expressions is RegexBuilder: http://renschler-.net/RegexBuilder/
Many others are available, such as http://regexlib.com/CheatSheet.aspx and http://regexlib.com.

Custom controls

The standard selection of Controls in K2 smartforms can also be extended by creating and registering Custom Controls. Note that this is a developer task and requires knowledge of C#.NET code, JavaScript, jQuery and an intermediate level of coding expertise. These custom Controls are usually created in a development tool like Visual Studio, and are then installed and registered in the K2 environment with a custom installer program.

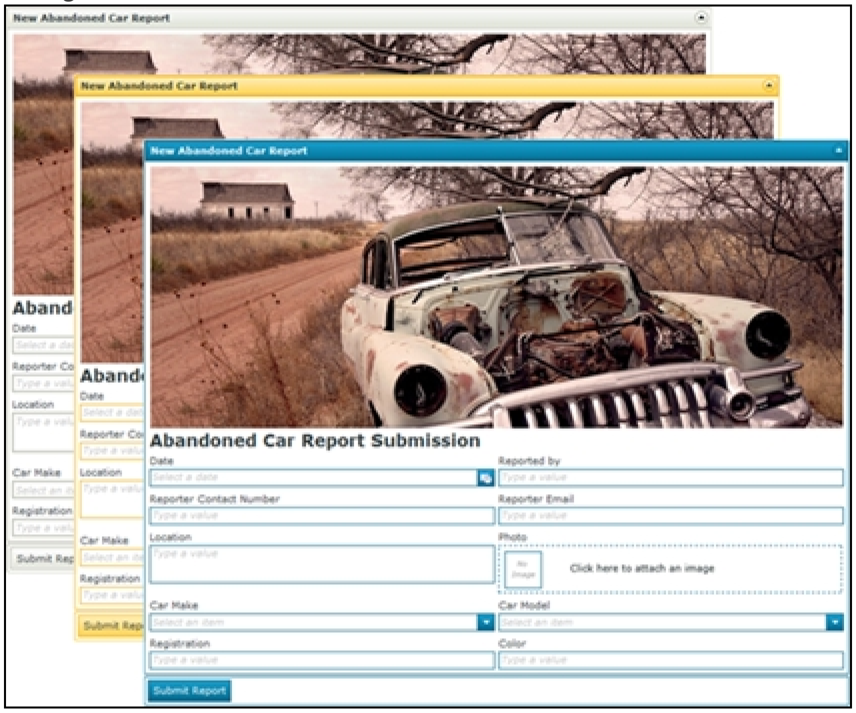
See the Custom Controls section of the SmartForms Developer Reference for guidance on creating custom Controls. You can also refer to the K2 community for samples of custom Controls.

Custom Themes

You can extend SmartForms by defining custom themes. This is normally a web developer task, or a task for a user familiar with Style Sheets. (If you are familiar with web development, themes are just based on Style Sheets). Custom themes are most often created by creating a copy of an existing Theme, editing it and then publishing it to the K2 SmartForms server.

Tip
Use the Lithium theme as a base for building responsive themes.

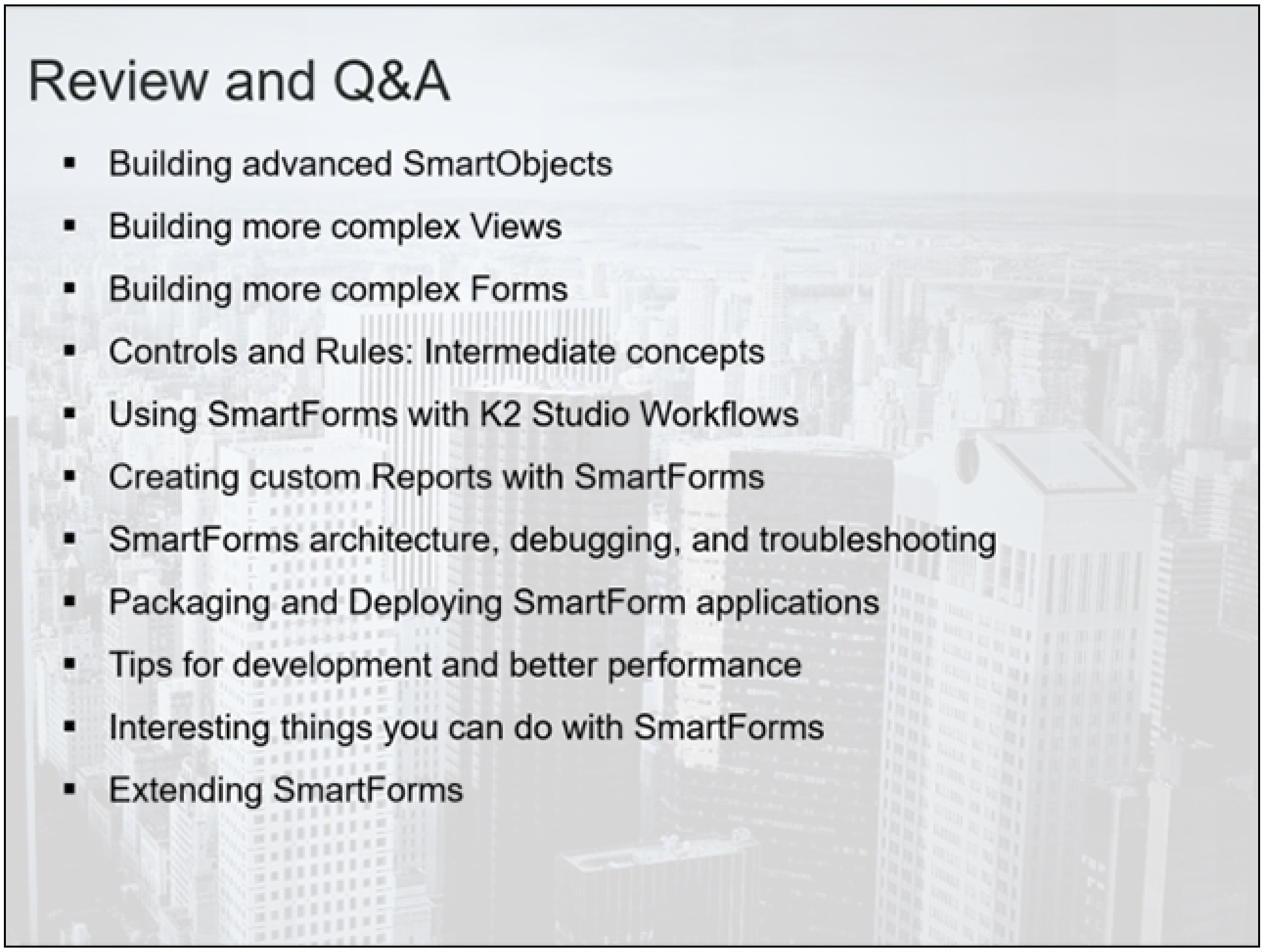
Using different themes on the same Form



Summary

- SmartForms can be extended with
  - Custom validation patterns
  - Custom themes
  - Custom Controls

Review and Q&A



Review and Q&A

- Building advanced SmartObjects
- Building more complex Views
- Building more complex Forms
- Controls and Rules: Intermediate concepts
- Using SmartForms with K2 Studio Workflows
- Creating custom Reports with SmartForms
- SmartForms architecture, debugging, and troubleshooting
- Packaging and Deploying SmartForm applications
- Tips for development and better performance
- Interesting things you can do with SmartForms
- Extending SmartForms

Video

This brings us to the end of the SmartForms Intermediate training module. You have learned quite a lot about building more complex Views and Forms and how to integrate them with SmartObjects and Workflows.

After completing this module and the exercises in this module you should know how to:

- Build advanced-mode SmartObjects in K2 Designer to integrate with external systems
- Build more complex Views including editable List Views
- Build more complex Forms including master-detail and Tabbed Forms
- Use Controls like the drop-down list Control (including cascading drop-downs), Picker Control, AutoComplete Control
- Define and use aggregations and expressions
- Use Execution Blocks to control how Rule Actions execute
- Build more advanced Rules with conditions and multiple Actions
- Use SmartForms in Workflows created in K2 Studio, and the options for integrating with those Workflows
- Use the reporting Controls to create custom reports with SmartForms
- The high-level architecture of SmartForms
- How and where to debug, review logging output and troubleshoot SmartForms
- How to use the K2 Package and Deployment tool to create and deploy packages
- Develop SmartForms easily
- Tune SmartForms for performance
- Some interesting things that are possible with SmartForms
- Some extension points that are available for SmartForms

Knowledge-check and discussion questions

Q: What kind of data sources would you integrate within your organization?
A: (discussion question)

Q: Can you identify a scenario in your organization where the master/detail Forms approach could be used?
A: (discussion question)

Q: In your role or in your organization, would you be building Workflows with K2 Designer or in K2 Studio?
A: (discussion question)

Q: What was the most interesting thing about the sample Sales Orders application that we built in this module?
A: (discussion question)

Q: What type of interesting things do you intend to do with SmartForms? Have you learned anything in this course that made you think about how else you might use SmartForms in your organization?
A: (discussion question)

Terms Of Use