

**Entity Modeling for OpenText
Process Platform 16.3**

Open Text Internal Use Only
Do Not Distribute

OpenText Process Suite

The software described in this Workbook is furnished under a license agreement or non-disclosure agreement (the "Agreement") with Open Text Corporation or one of its affiliates ("Open Text"). The software may be used or copied only in accordance with the terms of the Agreement. The information contained in this Workbook is the valuable property of Open Text and represents trade secrets of Open Text. No part of this Workbook may be copied, reproduced, translated or transmitted in any form or by any means without the prior written consent of Open Text.

The information contained in this Workbook is subject to change without notice. If this Workbook is provided in both printed and electronic form, the electronic form will govern in the event of any inconsistency. Open Text makes no representations or warranties concerning such information or the software described in this Workbook except as expressly set forth in the Agreement.

Copyright © 2018 Open Text. All rights reserved. Trademarks owned by Open Text.

Entity Modeling for OpenText Process Platform v16.0 - First Edition: April 2016
Entity Modeling for OpenText Process Platform v16.3 - Second Edition: March 2018

Comments or suggestions about this manual can be sent to LearningContentDev@opentext.com.

Based on OpenText Process Suite 16.3

Developed by OpenText Learning Content Development and Learning Services

Course Name: Entity Modeling for OpenText Process Platform

Course Number: 2-4912

Part Number: 2-4912-163-00

Welcome

Welcome to the Entity Modeling course for OpenText Process Suite Platform.

Entity modeling is an exciting feature available with Process Platform for OpenText Process Suite which allows more teams in an organization to model and describe common elements of an industry's data without performing low-level code development. By opening up development cycles to more members of an organization, this development process allows work to be distributed evenly across a business, and includes more teams in the contribution of the overall solution.

In this course, you will learn how to create and build entities in OpenText Process Suite Platform and populate those entities with rich content using the entity modeling tools and building blocks. You will then use your entities to create full user interfaces, including forms and layouts, which can be used in the end user environment: Process Experience.

Thank you for participating in this course. Should you require any further information, please contact us at OpenText Learning Services.

Good luck, and enjoy your learning experience.

OpenText Learning Services

Open Text Internal Use Only
Do Not Distribute

Table of Contents

1. Introduction

What you will build in this chapter	1-2
Timing	1-2
References.....	1-3
Using Process Suite to build more efficient solutions	1-3
Course agenda	1-4
OpenText Process Suite 16.3 components	1-6
Process Suite base	1-6
Applications	1-8
Integration products	1-9
Add-ons	1-9
Enterprise information management	1-11
Enterprise information management suites	1-12
Content Suite	1-12
Experience Suite	1-12
Information exchange products	1-13
Discovery Suite products	1-14
Business drivers for OpenText Process Suite	1-15
Straight-through processing	1-15
Key features of Process Platform	1-18
OpenText Process Suite community.....	1-19
Process Platform user start page.....	1-21
Views and sorting	1-23
Searching	1-24
Tag cloud	1-25
Shortcut bar	1-29
User start page preferences	1-30
Task bar	1-32
Summary.....	1-33
Exercises	1-37

2. Entity modeling

What you will build in this chapter	2-1
Timing	2-1
References.....	2-1
Building solutions in Process Platform.....	2-2
Multi-tenancy through organization management.....	2-4
Organizations	2-4
Collaborative workspace	2-7
Solution overview	2-9
Solution workspace and project	2-10
Building entities	2-13
Building an entity in Process Platform	2-15
Validating a solution.....	2-23
Entity types and subtypes	2-26

Summary.....	2-29
Exercises	2-33
Tips, tricks, and traps	2-34
3. Entity properties and relationships	
What you will build in this chapter.....	3-1
Timing	3-1
References.....	3-1
Building blocks	3-2
Property building block	3-3
Changing property building blocks	3-11
Relationship building blocks.....	3-14
Relationship types and directions	3-14
Multiplicity	3-15
Adding relationships in Process Platform	3-16
Title building block	3-19
Summary.....	3-20
Exercises	3-21
4. Lists in Process Experience	
What you will build in this chapter.....	4-1
Timing	4-2
References.....	4-2
About Process Experience.....	4-2
Navigate the Process Experience desktop	4-4
Desktop composition	4-4
Working with panels	4-9
Types of panels	4-11
Process Experience themes	4-12
List building block.....	4-13
Properties and filters on lists	4-15
Action bar building block	4-22
Publishing a solution to Process Experience	4-25
Process Experience Administration Console	4-27
Default entity roles	4-31
Summary.....	4-33
Exercises	4-34
Tips, tricks, and traps	4-37
5. Process Experience user interfaces: forms and layouts	
What you will build in this chapter.....	5-2
Timing	5-2
References.....	5-2
Forms.....	5-2
Create a form	5-3
Form configuration	5-4
Forms and relationships	5-11
Sub-forms	5-17
Layouts	5-22

Building the presentation of a layout	5-25
Layouts with flow objects	5-36
Home page layout	5-37
Discussion building block.....	5-39
Summary.....	5-41
Exercises	5-42
Tips, tricks, and traps	5-45
6. Rules	
What you will build in this chapter	6-1
Timing	6-1
References.....	6-2
Rule building block.....	6-2
Editing the logic of a rule	6-3
Basic mode	6-4
Advanced mode	6-12
Expression language for rules	6-12
Using a rule as a dynamic action on an action bar	6-17
History building block	6-22
Adding history	6-23
Summary.....	6-25
Exercises	6-26
7. Identity package and security	
What you will build in this chapter	7-1
Timing	7-2
References.....	7-2
Identity package.....	7-2
Entities in the identity package	7-3
Working with organizational models	7-12
Worklists	7-16
Identity package administration	7-17
Tracking building block	7-18
Tracking on child entities	7-19
Security building block	7-21
Summary.....	7-31
Exercises	7-32
8. Entity lifecycle	
What you will build in this chapter	8-1
Timing	8-1
References.....	8-2
About lifecycle	8-2
Elements of a lifecycle	8-2
Lifecycle building block	8-5
Configuring the lifecycle tasks	8-6
Designing forms with lifecycle	8-6
The lifecycle model designer	8-13
Follow a lifecycle in the entity	8-24

Activity flow building block	8-25
Activity flow editor	8-27
Summary.....	8-32
Exercises	8-33
9. Flow support	
What you will build in this chapter.....	9-1
Timing	9-1
References.....	9-2
The Email building blocks	9-2
Email building block	9-4
Email template building block	9-6
Deadline building block	9-11
Deadline policies	9-12
Alerts and escalations	9-14
Assignee building block	9-20
Assignee configuration	9-21
Summary.....	9-26
Exercises	9-27
10. Integration and advanced features	
What you will build in this chapter.....	10-1
Timing	10-2
References.....	10-2
Attach files and content to items	10-2
File building block	10-2
File building block at run-time	10-4
Content building block	10-6
Entities and web services	10-9
Web service building block	10-10
Find service operation	10-12
External entities	10-15
Establishing a connection to an external entity	10-15
Importing entities from a database	10-19
Summary.....	10-29
Exercises	10-30
Tips, tricks, and traps	10-33

Text Conventions

This workbook uses the following conventions:

Convention	What it is Used For
<i>Italic</i>	Italics are used for Workshops and Exercises.
Monospace	Monospaced text is used to represent sample code.
Bold	In instruction steps, indicates the action to be taken. In text it indicates emphasis
<>	Angle brackets (<>) represent an element of syntax you must substitute with a specific value.
	This icon represents a lesson symbol where the student watches the instructor.
	This icon represents a lesson symbol where the student follows along with the instructor.
	This icon represents a lesson symbol where the students perform the exercise on their own.
	This icon represents an optional or advanced lesson symbol where the students perform the exercise on their own.
	This icon represents a note that supplies additional information.
	This icon represents a tip that supplies additional shortcut information.
	This icon represents a collection of Tricks, Tips, and Traps that is used the end of a chapter.
	This icon represents a caution that supplies warning information.

Open Text Internal Use Only
Do Not Distribute

Student Attendance Form

Training Date: _____

Instructor: _____

Location: _____

Student Name: _____

Position: _____

Management Technical Other
Implementation End User Administrator

Industry: _____

Federal Government Legal
Other Government Manufacturing
Education Financial/Insurance
Integrator Other

Company: _____

Street Address: _____

E-mail: _____

Phone Number: _____

1. Introduction

Objectives

On completion of this module, participants should be able to:

- Explain Process Suite and the elements of Process Suite
- Explain the role of Process Platform in Process Suite
- Explain the business drivers to start implementing Process Suite
- Describe the role of Entity Modeling in development with Process Suite Platform.
- Launch the Process Platform Explorer
- Navigate through the Process Platform Explorer
- Customize the look and feel
- Launch applications in the Process Platform Explorer

Overview

This chapter introduces participants to Process Suite and Process Platform. It gives an overview of the various elements that together build OpenText Process Suite. In this course, we introduce entity modeling as a component of Process Platform and its essential position in solution development.

Process Suite is the key offering in the business process management pillar of the OpenText core capabilities. The core capabilities enable customers to manage content, continually improve their business processes, discover and exchange information, and create rich, engaging experiences for employees, partners, and markets. Process Suite contains a number of products, including Process Platform, to assist businesses. Process Platform includes the core development and integration features available in Process Suite. Entity modeling is one component of Process Platform available on the user start page.

The user start page is the single user interface for all types of users in Process Platform, whether they are administrators, developers, or end users. Working your way through this section, you will familiarize yourself with this user interface. It is the first step in getting introduced to Process Platform.

When you launch Process Platform in your web browser, a dialog box will prompt you to enter your user name and password. Upon successful authentication, the Process Platform user start page opens. Your user name and your current (default) organization are displayed at the top of the page.



In previous versions of this product, the Process Platform User Start Page was called the *Cordys User Start Page*, or *CUSP*. You may find references to CUSP in the product still. Be mindful that CUSP refers to the user start page.

By default, the user start page displays a shortcut bar, task bar, and one or more *application palettes* that contain the associated applications and documents. Entity modeling is an application in the palette. Process Platform uses the concept of *roles* to determine the level of access to Process Platform and its applications. While the shortcut bar and task bar are common across all roles, the content of the application palettes is role-specific and depends on the permissions granted to a role. The applications and documents available in the user start page and the application palettes are referred to as *artifacts*.



The content of the application palette depends on the roles that are assigned to you.

What you will build in this chapter

The exercises in this chapter are intended to familiarize you with Process Platform and applications in the palette. You will launch, filter, sort, and bookmark applications, as well as create custom application views and shortcuts to work with applications. These are fundamental skills for completing the remaining chapters.

Timing

Lecture: 60-75 minutes

Exercises: 30-45 minutes

References

More information about this subject is available:

- Documentation > Process Platform - An Overview > Getting Started with Explorer
- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)

Using Process Suite to build more efficient solutions

Process Suite presents a philosophical change to the way in which solutions are developed. Historically, business teams draft business requirements for an organization's solution, and depend upon development teams to implement the business vision. Often, the solution that is delivered does not necessarily match the design that the business teams drafted. The solution requires some incremental iterations before arriving at the intention. The difficulty with this development process is that the implementation aspects of the business vision are out of the hands of the most practical team - the business team - and left to development teams who specialize in development, but may not necessarily have intimate knowledge of the business processes.

This historical method of solution implementation does not maximize an organization's skills. In order to produce a solution efficiently, business teams should contribute business aspects to the solution and development teams should focus on contributing lower-level, technical software components. However, since business teams typically hand off use cases, usually in a printable format, they rely upon the business skills of technical teams to design their vision. This frequently results in miscommunication between the teams, poorly stated goals, and designs which are incorrectly implemented. The solution implementation loses time and progress as it is forced to incrementally repeat itself and fine tune goals and use cases. In some unfortunate cases, development teams invest a great deal of effort into an inefficient direction and, instead of aborting their work, choose to force the refined use cases into an inappropriate framework.

OpenText Process Suite offers organizations the opportunity to maximize the skills of their teams in the areas that they are most competent. It allows business teams to make a greater contribution to the overall solution while at the same time allowing development teams to focus their energy on building more low-level, technical functionality. Business teams do not need technical skills to implement their business vision in OpenText Process Suite: they simply need to focus on the aspects of the solution which suit their skills and collaborate with their peers by developing solutions in shared workspaces. The result is a more efficient progression from design to development and deployment, with less iterations and less miscommunication.

Entity modeling is a stage in solution development in which members of the business team describe and structure key elements of organizational data which will contribute to the overall solution without performing any actual development at the code level. By laying out the structure, business teams can become heavily involved in the representation of the data models in the end user environment: Process Experience. Business teams and development teams will work together to create the solution in a common development environment: Process Platform. Business teams will use entity modeling in Process Platform, and development teams will use other tools, such as process management, case management, and web service development.



This course will focus entirely upon entity modeling. Process modeling and development are available in course 4-4913: Process Modeling for Process Platform.

Course agenda

Introduction This chapter introduces you to OpenText as a company and Process Suite as a solution. It gives an overview of the various elements that together build Process Suite, including Process Experience and Process Platform. Entity modeling is a component of Process Platform. You will also learn about the Process Platform user start page, how to navigate it, and how to create a workspace for building your entities.

Entity modeling This chapter covers the basics of entity modeling such as the definition of an entity model and how to create basic models which you will then validate. In this course, you will build basic models then add features to them.

Entity properties and relationships In this chapter, you will learn about entity properties and learn how to distinguish between entities and their properties. You will then apply this knowledge by beginning to broaden the entities you built by adding properties.

Some entities may also possess intrinsic relationships, such as the relationship between an employee and a job position, or a product and an order. You will also learn how to model such relationships between entities.

Entities and lists in Process Experience Process Experience is the interface which end users will use to work with your overall business solution. Using entities, you will create a simple list in Process Experience for end users to work with your entities.

User interfaces: forms and layouts In this chapter, you will build user interfaces for Process Experience - forms and layouts - which you will compose for users to work in-depth with your entities. The instances which users create will be displayed in lists.

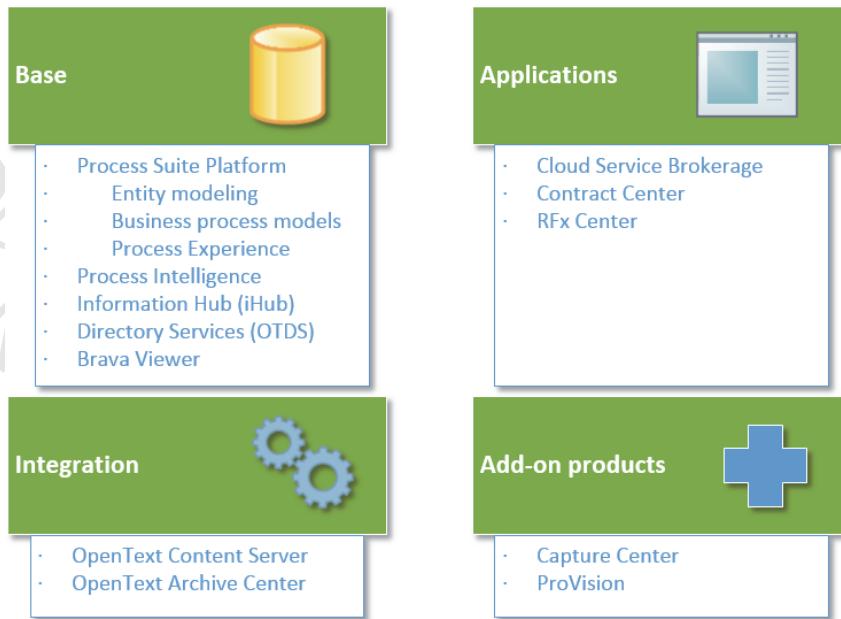
Entity rules In this chapter, you will learn about rule actions as they apply to entities and you will learn about the expression language used to build rules. You will then create your own rules to apply to entities.

- Identity package and security** The *identity package* is a construct in Process Suite which represents an organizational model across multiple platforms, including Process Experience and OpenText Directory Services (OTDS). In this chapter, you will learn how to work with and manipulate organizational structures in an identity package which includes lists and more. You will also learn how to apply the security building block to govern access to your entities.
- Entity lifecycle** The instance of an entity may go through several phases. For example, a request for customer support starts at the information gathering phase before it eventually arrives at a solution phase. The different phases of an entity's instance, from start to finish, express the item's *lifecycle*. In this chapter, you will build the item's complete lifecycle and map different layouts to each phase.
- Some phases of an entity's instance may be a more simple, straightforward progression. In these cases, you may choose to use an *activity flow* instead of a lifecycle.
- Entity flow support** The second chapter dedicated to the lifecycle and activity flow will further explore how to support flow-type constructs with additional building blocks, such as deadlines, tracking, and assignees.
- Integration** After having built and expanded upon your entities, the next step will be to integrate the entities within the overall solution. This is the stage in the process in which hand-off will begin between business and development teams. In this chapter, you will learn about adding external files to entities, importing entities from external sources, mapping the properties and data types of imported entities, wrapping entities as web services, and so on.

OpenText Process Suite 16.3 components

OpenText Process Suite relates to other OpenText products: the Process Suite base platform components relate to applications that run on the platform components, integration components, and add-on products.

Figure 1-1:
Relationship of OpenText products to Process Suite



Process Suite base

The principal base components of Process Suite are Process Platform, Process Experience, OpenText Directory Services, Process Intelligence and the Information Hub (iHub).

Process Suite Platform

Process Platform is based on heritage technology from Cordys. Process Platform provides a *business process management* (BPM) solution on a single platform. It is designed to bring the business and IT worlds together by enabling organizations to design, execute, monitor, change, and continuously optimize their critical business processes and operations. A set of integrated building blocks deliver a seamless, reliable, and high-performance design and run time environment to conduct enterprise business.



The product was renamed from Cordys Platform to Process Platform. It is likely that you will still discover some of the Cordys heritage names in Process Platform.

Process Experience Using Process Experience, which is included with Process Platform, organizations can work with ongoing cases, accounts, checklists, and tasks. They can also create customized user interfaces to access data on external systems. Process Experience brings together processes, application instances, and content from Process Intelligence, Process Platform, and third-party sources with ease.

Process Intelligence and Information Hub (iHub) OpenText Process Intelligence is a product from OpenText that provides intelligent analytics for BPM and case management systems. It works with OpenText BPM systems to deliver insight and operational status information, making process applications more powerful and useful. It can also integrate with other third-party BPM systems by means of the Process Intelligence adapter feature.

Process Intelligence receives information from a business process, transforms it into useful business intelligence, and reflects it in analytics, such as business intelligence reports. You can use OpenText Information Hub (iHub) or third party products such as Microsoft Excel, SQL Server Reporting Services, SharePoint, analysis tools, or other reporting software to display information in the Process Intelligence database. You can perform queries, create business intelligence reports, and create dashboards.

OpenText Directory Services OpenText Directory Services (OTDS) contain the services for identity synchronization between Process Suite components and other OpenText products. This synchronizes the users and groups with a central directory service to provide network users with single login access.



The virtual machine with which you will be working in this course has OpenText Directory Services already installed with default users, each with their own roles. These users will demonstrate and simulate a functional environment which uses Process Suite. You will be using OTDS in only the most basic way in which it applies to Process Suite.

If you would like more experience using OTDS, please consider course 3-0300: OpenText Directory Services Installation and Configuration.

The OTDS software provides seamless connectivity between Process Suite components and other integrated OpenText products, for example, Content Server, and a central directory service supporting the Lightweight Directory Access Protocol (LDAP) or the NT LAN Manager (NTLM) protocol used by Microsoft Windows Active Directory. In this example, organizations can administer users and groups in OTDS and push the information out to Content Server.

Once again, using Content Server as an example, once users log into Process Platform, they are transparently logged on to Content Server with OTDS, and are not required to enter their user names and passwords again. This authentication facility is supported for web browser access (under the control of the HTTP or web application server), and for applications that access Content Server through the OpenText Application Programming Interface (OTAPI).

Brava! Viewer Brava! provides access to content in virtually any format—including PDFs, Microsoft Office documents, image files, CAD drawings, 3D models and even video clips—without allowing the source file to be edited or downloaded. It enables workers to view the content they need and collaborate with coworkers and external stakeholders, all within the business rules so organizations can operate efficiently while meeting compliance and security objectives.

Users can add markups and collaborate using Brava!'s robust markup tools, create new file renditions as PDF or TIFF—even redact sensitive information in the new file—for easy sharing with external recipients, and add electronic signatures.

Applications

Process Suite applications include OpenText Cloud Service Brokerage and Contract Center.

Cloud Service Brokerage OpenText Cloud Service Brokerage is an add-on product for OpenText Process Platform. It extends the platform with capabilities for provisioning of the organizations, users, and applications across multiple clusters.

Contract Center Legally binding contracts and agreements touch every area of the enterprise, defining relationships with partners, suppliers, customers, and employees. The problem is many organizations find contracts are spread throughout various departments, where different types are not managed correctly, there are manual steps in processing and contracts end up in different formats and stored in different systems. The result is long cycle times and high costs for contract management, and the risk of not complying with internal information governance mandates or regulatory requirements.

OpenText Contract Center provides a complete integrated solution for all types of contracts, including buy-side, sell-side, and other legal agreements. It is an out-of-the-box solution for all aspects of contract processing, from initiation and request, to authoring, negotiation, approval, execution, management, and renewal.

OpenText RFx Center OpenText RFx Center is a point solution targeted at automating and improving the efficiency of bid management - creating, advertising, distributing, evaluating, and awarding opportunities. Customers using OpenText Content Server or OpenText Process Suite can easily deploy RFx Center on their existing infrastructure.

OpenText RFx Center automates procurement lifecycles to eliminate manual steps, reduce errors and inefficiency, and cut costs.

Integration products	The following OpenText products can be integrated with Process Suite:
OpenText Content Server	<p>OpenText Content Server provides a single, comprehensive solution for managing information and makes collaboration a part of every business process, even across organizational and geographic boundaries. It is the core content repository for Process Suite, as well as for a broad range of OpenText products such as the Content Suite Platform (formerly CLM), OpenText Content Suite, Application Governance and Archiving for SharePoint, Extended ECM for SAP Solutions, Extended ECM for Oracle® E-Business Suite, Email Management, and other OpenText Enterprise Information Management (EIM) offerings.</p> <p>Content Server provides richly featured enterprise services based upon a unique combination of integrated tools that include document management, workflow, and information retrieval services, all tightly integrated into a solution that is easily customized and extended.</p> <p>Content Server is used as a potential document management system in Process Suite and the connectivity is facilitated through the Document Store connector in OpenText Process Platform. The Brava Viewer, which is included in Content Server, provides integration for advanced viewing and editing of documents maintained in storage.</p>
OpenText Archive Center	OpenText Archive Center is a highly scalable, multi-tenant archive helping organizations manage the explosion of unstructured data (e.g., email and files) while ensuring it is cost effectively and securely stored in the cloud, on premise or as a hybrid. End users access data from anywhere—online, offline or mobile—and it's completely stub-free and transparent. Business administration features provides a broad set of archive and retention management capabilities along with system reporting, legal hold and export.
Add-ons	The following OpenText add-on products are available for use with Process Suite:
OpenText Capture Center	OpenText Capture Center enables an organization to more quickly and efficiently capture, digitize, and fully index paper documents, forms, and faxes, and then move them into virtually any back-end system. It performs recognition tasks such as scans from mobile devices, helping users to reduce their dependence on paper based processes and deliver business critical electronic content to corporate workflows and business processes.
ProVision	OpenText ProVision is an end-to-end solution for enterprise and business architecture and business process analysis – allowing customers to translate business strategy and operational objectives into successful enterprise change through views that describe enterprise assets and their relationships.
AppWorks Gateway	AppWorks is the OpenText developer platform that is designed to speed the path from requirements to solutions for IT organizations. AppWorks allows organizations to quickly and easily build well-designed, purpose-specific applications for your enterprise, with two very special capabilities.

First, AppWorks applications can tap into the wealth of information capabilities of your Enterprise Information Management solution. This means that your rapidly assembled applications can have media, processes, discovery, and content management capabilities with minimal effort.

The second capability of AppWorks is that the applications will immediately work on multiple devices and platforms.

AppWorks Developer is the OpenText Developer Network: the primary resource for all developers wishing to create cross-platform Enterprise Applications with AppWorks. It should be your first stop if you are looking to create AppWorks applications. AppWorks Developer helps you to learn about creating, deploying, and managing your own Enterprise Applications that connect to OpenText EIM services through AppWorks, and make them available from all platforms.



AppWorks is not installed in the course image, nor will it be used in this course. Please consult the AppWorks Developer Gateway for more information about AppWorks:

<http://developer.opentext.com/>

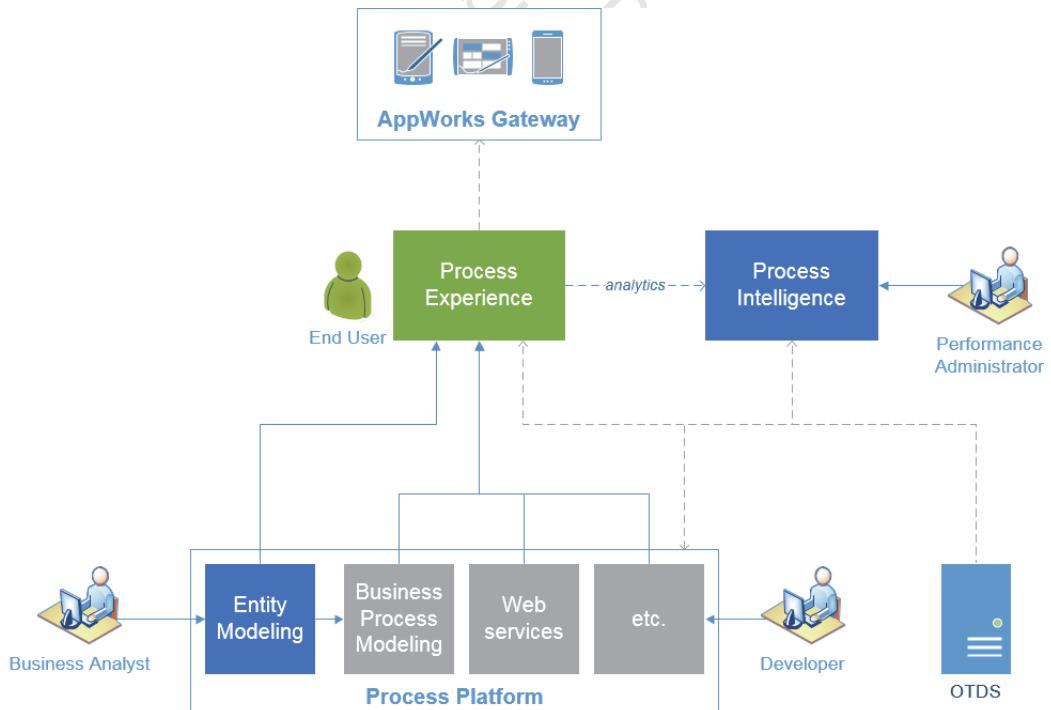


Figure 1-2: Relationship of platform components in Process Suite

Enterprise information management The core capabilities of enterprise information management (EIM) include:

- Manage content, and make it available without compromising security and governance.
- Create and continually improve business processes.
- Create rich, engaging experiences for customers, markets, employees and partners.
- Exchange information.
- Discover information.
- Provide platforms and enable application development.
- Help customers extract more value out of their existing investments, particularly investments into the largest enterprise SW companies.
- Finally, provide solutions in the cloud and on premises.

Each of the EIM pillars is available as a standalone suite of tools:

- Discovery Suite: organize and visualize all relevant enterprise information.
- Process Suite: work more efficiently.
- Content Suite: build information confidence - manage all of your content throughout its lifecycle. Reduce governance and security risks and costs.
- Experience Suite: build exceptional information experiences across channels and devices.
- Information Exchange Suite: secure, compliant exchange of information inside and outside of the enterprise.

OpenText Cloud OpenText Cloud offers managed hosting services. By including OpenText Cloud in your Process Suite solution, your OpenText EIM applications are hosted in our Cloud Infrastructure so that you can focus on what truly matters: your business.

- Built for the enterprise
- Secure, elastic and cost-effective
- High availability and redundancy at your fingertips

Information Exchange cloud services Our Information Exchange cloud services are designed to facilitate efficient, secure, and compliant exchange of information inside and outside of the enterprise.

- Safely exchange any information with anyone
- Integrate with your business processes
- Fast on-boarding and rapid ROI

Enterprise information management suites

Content Suite

Content Suite is used to capture technology in order to scan, index and classify content thereby transforming physical records into valuable digital assets. Content Suite provides:

- A fully featured, highly scalable, Web-based document management system providing a secure, single repository for organizing and sharing enterprise content.
- Workflow to automate processes, for accuracy and consistency. Processes can be designed graphically, according to corporate or regulatory standards.
- A flexible user interface designed for easy access and interaction with content directly from office productivity applications, and processes.
- Records management to manage the full lifecycle of all enterprise content, electronic or physical, enabling you to control retention and ensure destruction at the right time.
- Intelligent archiving that optimizes storage according to business context and metadata, leveraging less expensive media and providing high-end storage reduction services.
- A common, documented, standard layer of development tools to rapidly create, deploy and manage enterprise applications.

Experience Suite

Web content management Create and deliver dynamic, targeted interactions across multiple geographic touch points with this comprehensive solution built for high-performance, enterprise-scalable, and transaction-oriented web applications.

- Leverage responsive design to display content consistently across mobile devices.
- Increase revenue by satisfying customers with compelling experiences and meaningful conversations.
- Take advantage of every customer touch point with device neutral, content-centric experiences.

Customer communications management Enhance customer relationships and value by enabling automated and interactive communications of business-to-business (B2B) and business-to-consumer (B2C) organizations.

- Personalize touchpoints for 1-on-1 contact in mass produced documents.
- Simplify and automate document handling.
- Improve communication adoption rates.

Social communities	Socially-change your current business processes in a safe, compliant and easy to use fashion; creating an immersive user experience both for social business and the social marketplace.
	<ul style="list-style-type: none"> ● Create a social workplace. ● Augment social marketing. ● Build a more socialized intranet.
Portal	Improve engagement with your employees, partners, and customers through highly interactive and contextual online experiences.
	<ul style="list-style-type: none"> ● Create and deploy mashups and composite applications. ● Deploy micro sites and global initiatives with ease. ● Improve customer service with an online community to express and share opinions.
Digital asset management	The proven, tested enterprise digital asset management (e-DAM) solution that accelerates the workflow-driven creation, collaboration, production and distribution of digital media.
	<ul style="list-style-type: none"> ● Customizable, browser-based portal. ● Any number of “consumer users” to read, search, view, browse and download. ● Embedded file acceleration to transfer large files faster.
Information exchange products	
Connectivity	Provide high-performance access to critical business applications regardless of geographical boundaries with virtual desktop solutions and terminal emulation for desktops and mobile devices.
	<ul style="list-style-type: none"> ● Accelerate business processes and improve enterprise agility. ● Increase user productivity and application experience. ● Improve security while reducing IT costs.
Secure messaging	Secure messaging and file transfer solutions to exchange sensitive business information inside and outside the firewall with air-tight security and complete audit tracking.
	<ul style="list-style-type: none"> ● Protect business information and control access. ● Accelerate information exchange of any size. ● Ensure regulatory compliance in the cloud or on premise.
B2B integration	Optimize the reliability, reach, and cost efficiency of your electronic B2B supply chain with EDI and Integration services that go beyond traditional VANs.
	<ul style="list-style-type: none"> ● World-class EDI transaction network. ● Managed EDI services for maximum cost efficiency. ● Bridge the gap with non-EDI partners.

Capture and recognition Capture paper and electronic documents and transform them into digital content to reduce cost, speed up processes, and ensure legal compliance.

- Turn paper documents into machine readable information.
- Integrate non-digital information with business processes.
- Control the flow of documents across your organization.

Discovery Suite products

Content analytics Extract meaning, nuance, and context from vast amounts of unstructured content to analyze it and harness its true business potential.

- Create machine-readable content from unstructured data.
- Connect people with relevant content.
- Discover valuable factual information to support decisions.
- Boost productivity and reduce business risk.

OpenText semantic navigation Deliver highly relevant information to website visitors and drive engagement by letting them find what they are actually looking for rather than what they search for.

- Automatically analyzes and tags content to produce insightful facets.
- Improve website conversion rates through relevant result sets.
- Aggregate information from all systems and web properties for unified access.

Auto-classification Establish a highly defensible, completely transparent records management program as part of a broader information governance strategy while automating policy application tasks for business users.

- Reduce litigation risk, eDiscovery and storage costs.
- Improve compliance, security, and user productivity.
- Save time in addressing the need to classify huge volumes of legacy content, email, and social media.

eDiscovery Reduce risk and cost by being “Litigation Ready” and empowering your eDiscovery team to perform expensive, typically outsourced activities.

- Significantly reduce the cost and risk of legal discovery.
- Centrally search and preserve information from disparate sources.
- Dramatically reduce the amount of content to be reviewed.

Business drivers for OpenText Process Suite

There are several business drivers for implementing an efficient BPM solution with OpenText Process Suite. These drivers are frequent and fast regulatory changes, combined product offerings, the necessity for an integrated client view, and the need to establish effective 'straight-through' processing. By implementing lean process principles, businesses eliminate waste.

Regulatory changes	A company's policies and regulations often change, sometimes monthly or even more frequently. Regulations can also change when departments or companies merge. Governments change their regulations annually, if not more often. A business must be prepared to change processes from inside as well as outside the organization.
Combined product offerings	When companies merge they often offer new or adjusted products. With the agility Process Platform platform provides, you can combine the processes and operations of your company to adjust to the changes and provide seamless transformation to the new products.
Integrated client view	Most companies use many different systems to provide services and software. A CRM, PLM, ERP, SCM, etc. Users will often need to switch between programs or sometimes systems and hardware to access the various systems. This is time-consuming and non-effective. What you need is an integrated view; a single piece of software that allows you to access all the necessary systems and services without the need to change between systems and software, regardless of the hardware or software involved.
Straight-through processing	This is the process of enabling total automation of corporate processes. Once the process has been started it completes without further need of the user to perform actions. As a business you want to deliver your products to the customer as fast as possible. Implementing straight-through processing will significantly increase efficiency and lower overall costs, resulting in more profit. Example: A car manufacturer produces 5000 cars per day. They have a lot of storage for the components needed. They produce a set amount of cars and store the overhead. Lean processing removes all the "waste" in processes. Toyota, for example, exploits this way of working. They have no storage facilities for car parts. The parts are delivered exactly on time when they're needed. Preferably the same day or even the same hour before production. They only produce cars that have been ordered. They've effectively cut their waste processes, resulting in high efficiency and low costs. The risk of lean processes is that it's volatile; if one action in the chain goes wrong the whole chain is affected and the overall process delayed.

The 7 wastes of a lean process are:

- Transportation,
- Inventory,
- Motion,
- Wait or Work in Progress,
- Over-processing,
- Over-production, and
- Defect.

You can remember these principles of lean processes with the acronym:
TIMWOOD.

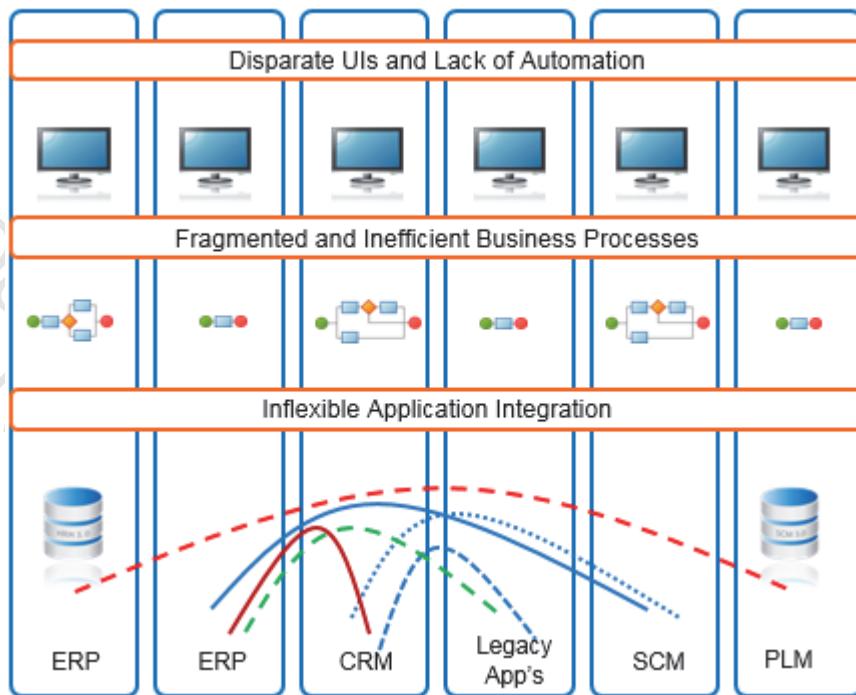
The period of time in which business is likely to change is much shorter than the period of time that IT is able to change. Business strategy changes over the years, while business execution and organization changes take place every 3 to 6 months.

IT infrastructure and software ask a lot of investment both in development, implementation and embedding in the company. A system lifecycle of 6-10 years is quite common. Though perfectly understandable, this difference in pace is a common source of frustration. The business needs agile IT to support their activities.

The “old way” The infrastructure of a company used to be a maze of systems, software, services, connections and processes. Every system would require their own connection and software and would run separately from other systems. If one system needed data from another you’d have to go through several hoops to manage it, sometimes resulting to manually putting in the data.

The business layer with its different UIs, the business process layer and the back-end applications were completely separate and independent of each other. Information was scattered across databases and systems.

Figure 1-3:
The “old way” of
processing

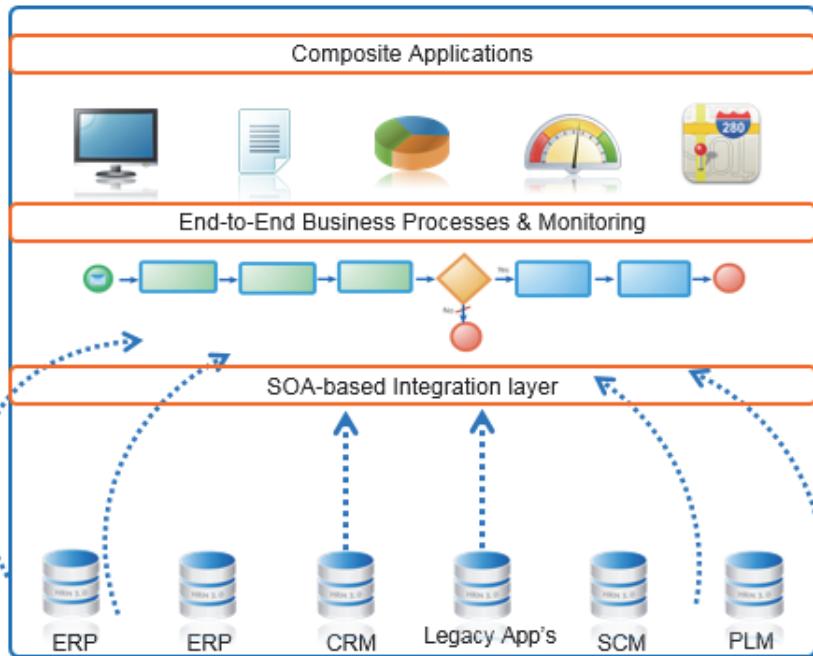


The “new way” The new way of working solves a lot of issues. The process layer is now between the front-end and back-end. The front-end UIs and applications connect to the back-end through the process layer. XML is a common language for building and translating homogeneous messages. This ensures that all the information is available in one place and removes the need for separate systems or software to administer data. Within one process you can connect to multiple back-end systems and use the information in a combined UI, displaying data from more than one source.

Not only can you input and update data, you can read back-end data and use it for KPIs and dashboards to quickly get an overview of your business’s performance.

The technology has changed from point-to-point connections to an enterprise bus system, where the information is available to all the connected systems.

Figure 1-4:
The “new way” of
processing



Key features of Process Platform

Entity modeling is only one aspect of Process Suite Platform (or “Process Platform”). Process Platform offers many other important aspects in designing a solution.

- One platform to perform integration, business process management, and composite application development: no need for separate applications or hardware.
- It brings the business and IT closer together. The gap between the two is largely bridged using a collaborative workspace. While the business users design entities, developers can add functionality.
- Many users work(ed) with either human-to-human or system-to-system workflows. Process Platform allows the user to make any combination.
- Process Platform is designed for multi-tenancy (a large group of users) as well as cloud deployment. This means Process Platform can be deployed in the cloud as *software as a service* (SaaS), where everyone can make use of the software. It is up to the company to decide the best way to distribute the software.

Process Platform is made to cater to the modern demands of businesses and technology. We continually review and improve our product to keep up with the latest innovations. It offers an open platform which can be used in a versatile way.



In this course, you will use the entity modeling aspect of Process Platform. In the process modeling course, 4-4913, you will learn to build more solution components with Process Platform, such as *business process diagrams* (BPDs).

OpenText Process Suite community

OpenText uses its online community to share and exchange knowledge. OpenText shares all its knowledge real-time. Knowledge is shared in a wiki, forums, presentations, and white paper documents. Process Platform is inside this Process Suite Community. Whatever OpenText produces is made available for you in your role as a Process Suite professional. Users find great value in being able to be updated immediately on any event they're subscribed to inside the Process Suite community to be able to connect with people that have knowledge from the source and are willing to share it.

Inside the Process Suite community, you can find detailed explanations on entity modeling and how to apply business process management (BPM), specifically with OpenText Process Platform products. The Process Suite community also contains information about composite applications and *service-oriented architecture* (SOA), two other vital ingredients for successful BPM initiatives.

You can get up to speed with the latest updates and best practices. The community is available to OpenText employees, partners and customers, and contains the following sources:

- Content
- Product downloads
- Product knowledge, blogs, and forums
- Product documentation
- Customer support
- How-to's
- Demos
- Implementation methodology

The screenshot shows the OpenText Process Suite Developer Community home page. At the top, there's a navigation bar with links for 'WHAT WE DO', 'WHO WE ARE', 'VIDEOS', 'CUSTOMER STORIES', 'COMMUNITY', 'SUPPORT', and 'EVENTS AND WE'. Below the navigation is a banner with a blue gradient background. On the left, there's a sidebar with 'Sign Out' at the top, followed by 'Community > Process Suite Developer Community'. The main content area has a title 'Process Suite Developer Community'. On the left side of the main content, there are two sections: 'Tasks' and 'Forums'. The 'Tasks' section includes links for 'Documentation', 'Training & Certification', 'Get Support', 'Download', 'Expert Spaces', and 'Information'. The 'Forums' section includes links for 'Training & Certification', 'Expert', and 'Information'. To the right of these sections is a 'Recently Updated' feed. It shows four recent posts from users Ethan Beisher, van Vliet Bas, David Redman, and gupta sourabh. Each post includes a small profile picture, the user's name, the topic they commented on, and a timestamp indicating when they commented. A dropdown menu labeled 'All Spaces' is visible above the first post.

Figure 1-5:

Process Suite developer community home

The Process Suite community uses spaces and pages. Spaces contain multiple pages with information related to the space. Once you are at a page you have several options. The menu on the top right lets you perform actions based on your access rights.



You are encouraged to register with the OpenText Process Suite community in order to take advantage of the forums, documents, and information shared there. Visit:
<http://www.opentext.com/community>

Process Platform user start page

When a user connects to Process Platform using a browser (i.e., Google Chrome, Microsoft Internet Explorer, or Firefox) the user start page is opened by default.

At the top, the My Application palette is displayed, which includes the user name, a sign-out button (if applicable), and the current organization.

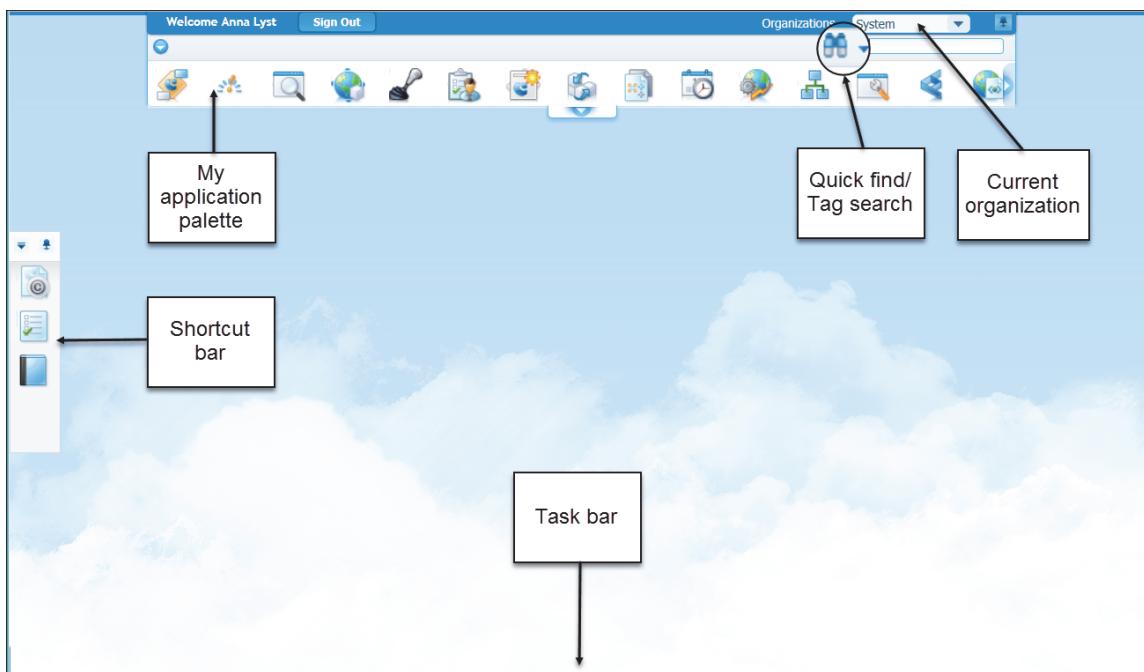


Figure 1-6: User start page

The My Application palette works like a roll down curtain: you can click the blue handle at the center to open it and show all the artifacts that are assigned to you based on the roles you have in Process Platform. An *artifact* is a common name for something a user can start, like My Inbox, a web page, a document, etc. It also contains a search option to search for artifacts based on their name or the tags assigned to them.

On the left, a shortcut bar is displayed. Next to the default options, you can also add your own favorites.

On the bottom you have the task bar. The user start page works similar to most client desktops. You can open multiple applications at the same time and the application is displayed on the task bar so you can switch between applications without the need to reopen them.

When you have opened the My Application palette you see all the artifacts available to you. When you click on an artifact, the artifact is opened and added to the recently used list at the bottom of the My Application palette.

This way you do not need to open the My Application palette to start this artifact the next time. Also, the open application is displayed on the task bar.

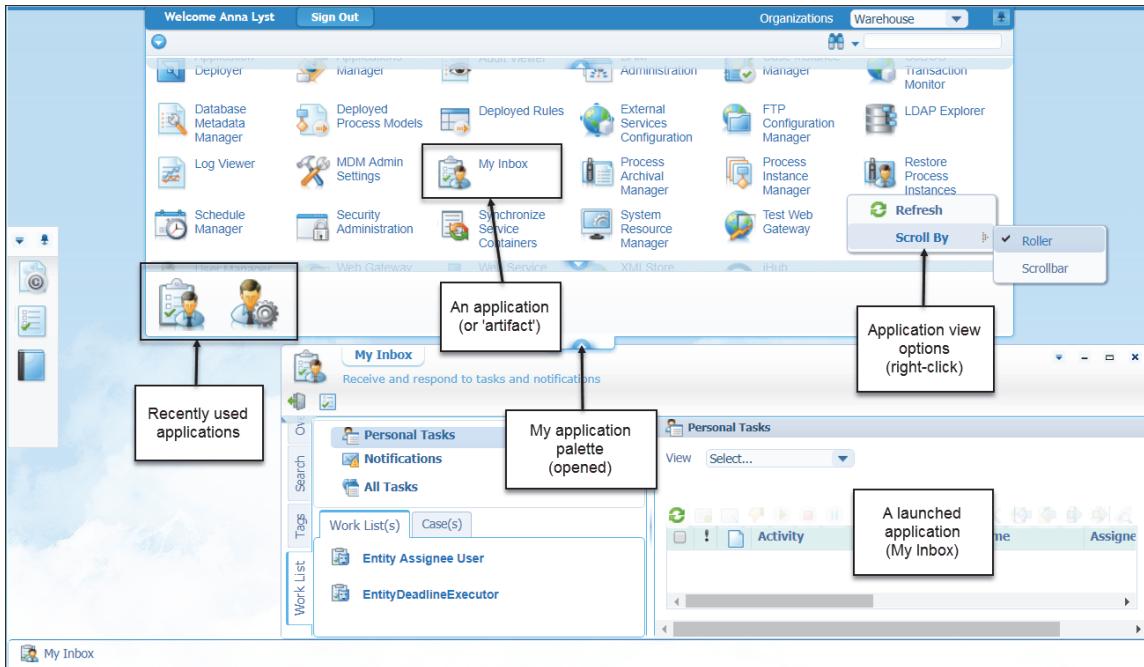


Figure 1-7: User start page with opened palette

You can right-click in any application palette, which shows a context menu with applicable options, like refresh, sort, etc.



Log in to Process Platform and acquaint yourself with the user start page

1. Open Google Chrome or Microsoft Internet Explorer.



You may use any web browser, but Internet Explorer is the preferred browser. Both Internet Explorer and Google Chrome have been installed and pre-configured with shortcuts for this course.

2. In the shortcut bar of the web browser, click Process Suite 16.3 > Warehouse Org > Process Platform.



You may use the shortcut which has been prepared in Internet Explorer, or use the following URL:

`http://localhost:81/home/warehouse`

This URL is specific to this training environment. In general, the URL for accessing Process Platform is:

`http://<server_name>:<port>/home/<organization>`

3. Sign in to Process Platform using the user name **analyst@training.local** and the password **opentext**.
4. Examine the shortcut bar on the left and the My Application palette at the top.
5. Open the My Application palette with the blue handle in the center of the palette.
6. Examine the artifacts in the My Application palette.
7. Use the handle to roll down and see the other artifacts in the My Application palette.
8. Right-click on any blank space in the My Application palette and select Scroll By > Scrollbar to change to a traditional scroll bar.

Views and sorting

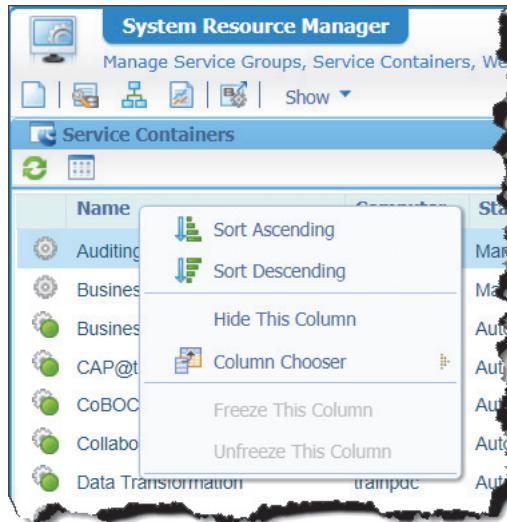
An application palette can be used for many things. For example, the My Application palette shows the artifacts you can start based on your role; the System Resource Manager shows the back-end connections configured to maintain them. In your own application, you can use them to show employees, customers, products, pictures, etc.

A context menu is available to change and control the behavior and display of the application palette.



Options differ per application palette because each palette has its own purpose. Compare the context menu of the System Resource Manager with the My Application palette, even the display chosen has an effect on the options.

Figure 1-8:
Context menu for System Resource Manager



Searching

Most applications you use will come with a search option. For example, the My Application palette has a search box on top to search for artifacts based on their name or a tag name.

Figure 1-9:
Search field in My Application palette (palette collapsed)

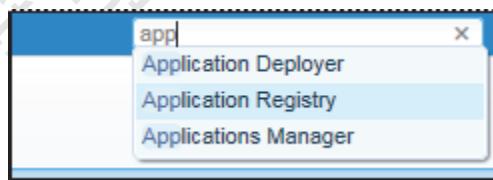
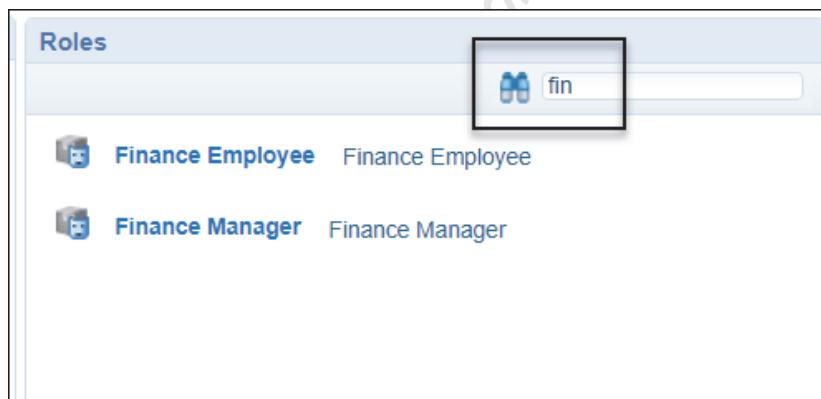


Figure 1-10:
Search field in Roles group box of User Manager



When you look at the Process Platform User Manager it has multiple search boxes: a general one for finding users and roles and then in specific parts of the User Manager to search for an entry in a displayed group. In this example, the search field is in the Roles group box for a role containing the text "Adm".

Tag cloud

An artifact can be tagged with a name which offers you the option to filter the artifacts based on the tag name.

In Process Platform, the artifacts in the My Application palette are already tagged with one or more default tags like Administration, Processes, etc. You can also add your own personal tags to filter based on your criteria.

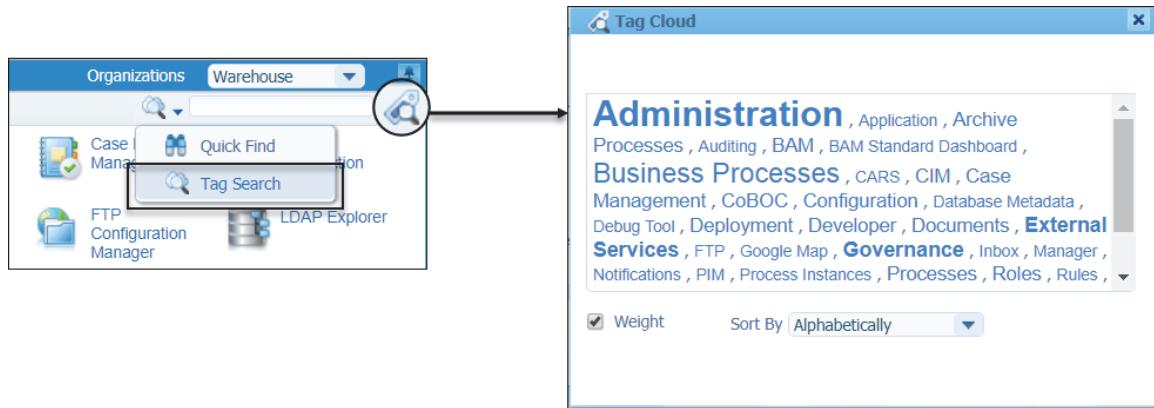
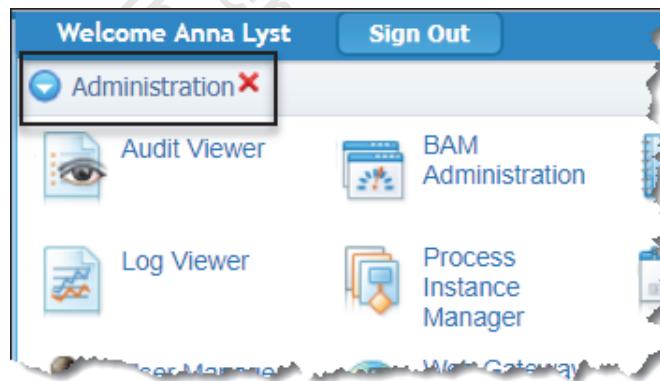


Figure 1-11: Switching to tag search and showing the tag cloud

Figure 1-12:
My Application palette
with Administration tag
crumb



To remove a tag crumb you can click the red cross next to a tag name.



Configure features of the *My Application* palette

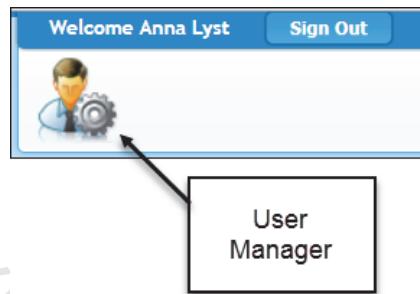
1. In the Process Platform Explorer, open the My Application palette.
2. Use the search bar in the upper right of the palette to search for any artifact that contains the letters "USER". How many artifacts are there?
3. Click the User Manager application.

The User Manager application is started and the My Applications palette, if open, is now closed.

4. Maximize the User Manager application.

5. Click the hide my applications button () in the top-right of the My Application palette to hide the palette.
6. Restore the User Manager application to its original size.
7. Hover over the My Application palette handle to display the collapsed palette.
8. Click the show my applications button to show the palette again. You will notice that the User Manager application has been added as a recently used application.

Figure 1-13:
User Manager as a
recently used application



View and sort applications in the application palette

1. Use the search bar in the upper right corner of the application palette filter to search for artifacts that contain the word “MANAGER”. How many artifacts are there?
2. Switch the search bar to Tag Search.
3. Click the show tags button.
4. Select “Processes” from the tag cloud.
5. Close the tag cloud. How many artifacts have been tagged with “Processes”?
6. Click the red X next to the “Processes” tag crumb to remove it.
7. Change the search bar back to Quick Find mode.

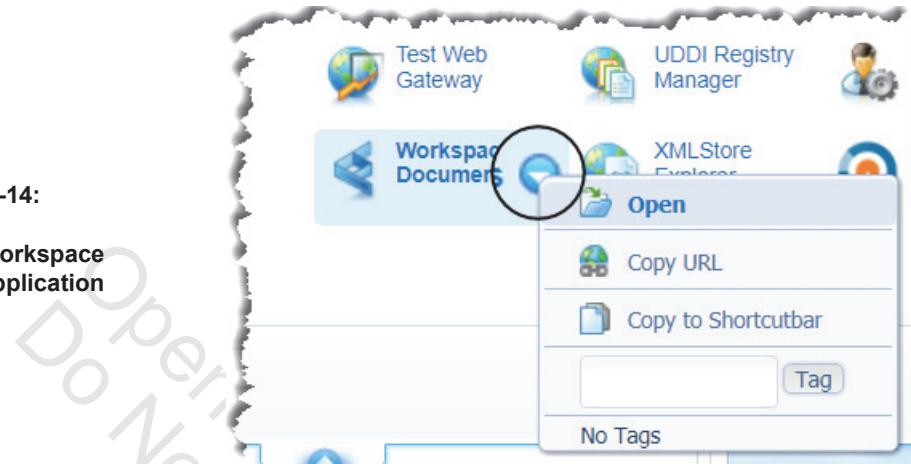


Add your own tags to the application tag cloud

1. Open the applications palette.

2. Hover your mouse over the *Workspace Documents* artifact and click the blue options button.

Figure 1-14:
Options for Workspace Documents application

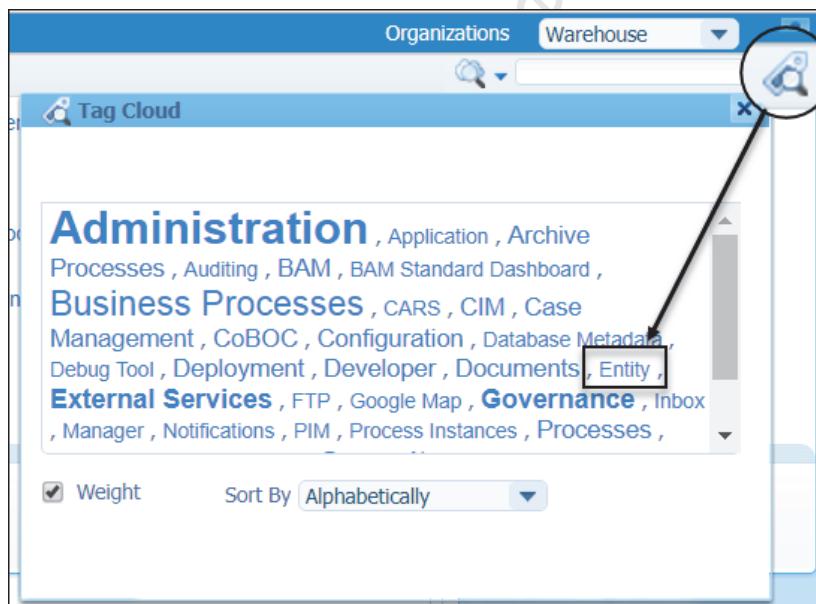


3. Enter the value "Entity" and click the Tag button.
4. Apply the "Entity" tag to the New Document artifact.

When you start typing the tag, a selection list appears from which you can select existing tags.

5. Click the Tag Search icon in the My Applications palette.
6. Click Show Tags on the right.
7. Click "Entity" from the tag cloud and close the Tag Cloud.

Figure 1-15:
Entity tag in tag cloud





Create a personalized view from a tag

1. Click the blue Options button in front of the Entity tag crumb.



Figure 1-16:

Options on tag crumb

2. Select **Make current selection as personalized view.**
3. Enter the name “Entity Modeling View” and click OK.
4. Dismiss the Entity tag crumb of the application palette (the lower part).
You will now see both the applications palette and your personalized view.
5. Refresh your browser.
6. Open the applications palette.

The palette will now contain both the personalized view (on top) and the default applications (at the bottom).

7. Open the Preferences application via the shortcut bar.
8. Select the Views tab.
9. Select Entity Modeling View and click Disable in the toolbar.

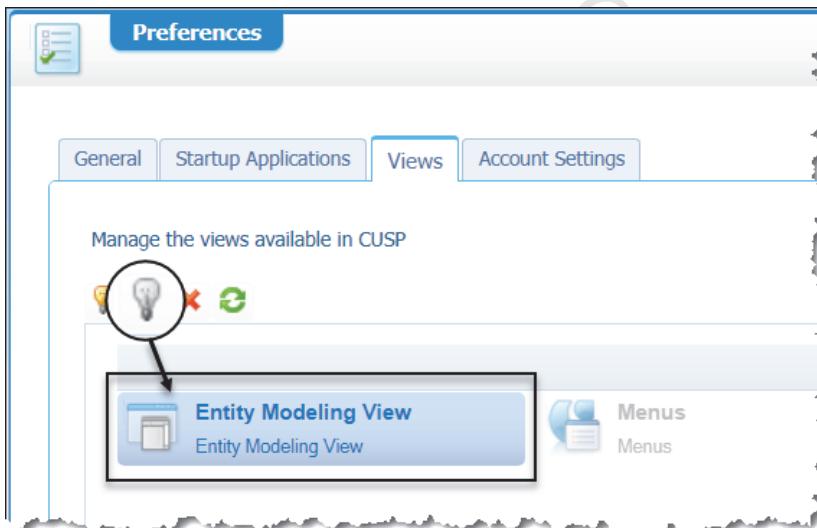


Figure 1-17:

Disable Entity Modeling View in preferences

10. Click OK.

11. Refresh your browser (tab) to make sure the view has been disabled.

You will notice that the Entity Modeling view is not displayed. The view is still listed in the Views tab of the user start page Preferences, but has been grayed out to indicate it is disabled.

Shortcut bar

On the left in the user start page you have a shortcut bar which comes with some default options, next to that you can add your own options. There are two ways to do this:

1. Drag-and-drop an artifact from the application palette, or,
2. Manually add an entry by providing the details, like the URL address.

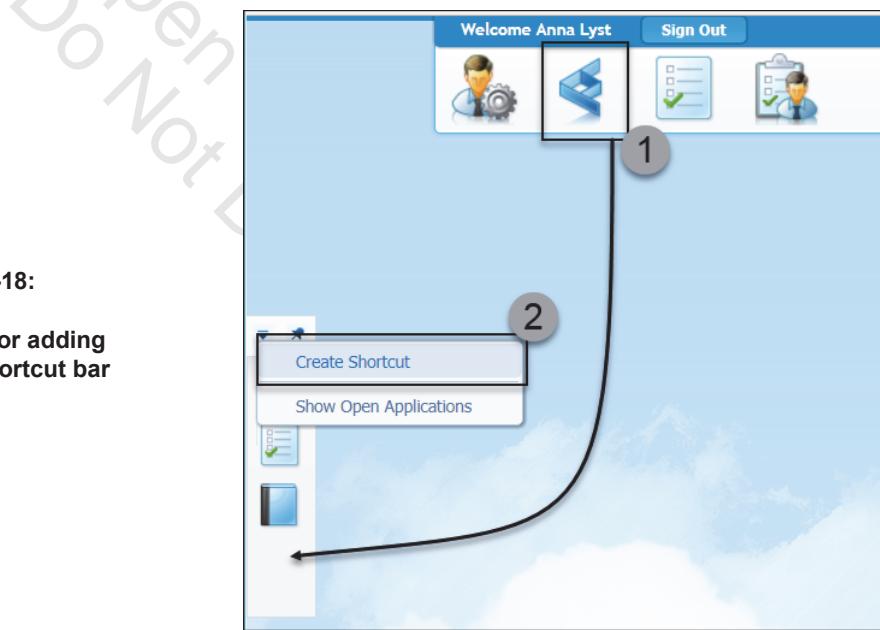


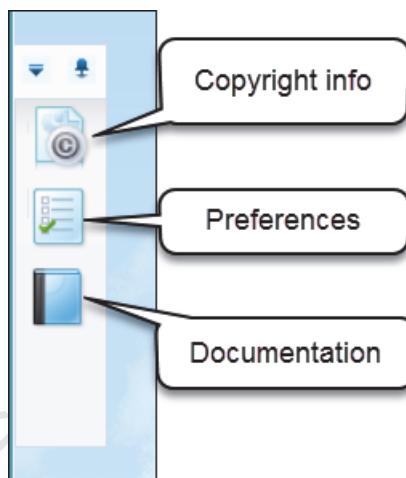
Figure 1-18:

Two methods for adding
shortcuts to shortcut bar

There are three default shortcuts provided on the shortcut bar:

- Copyright information and installation details
- Preferences
- Documentation

Figure 1-19:
Default shortcuts on
shortcut bar



User start page preferences

Users can specify their own preferences for the user start page:

- The **General** tab contains options such as specifying the language in which you want forms to be displayed. If a form is available in the target language, all labels will be displayed in that language.
- The **Startup Applications** tab offers the possibility to automatically start certain applications when you connect to Process Platform like, for example, the Workspace Documents application.
- The **Views** tab gives you the option to open a view which uses a tag filter to display artifacts based on the selected tags.
- The **Account Settings** tab provides information regarding the current logged in user. The user can change this information and update it.

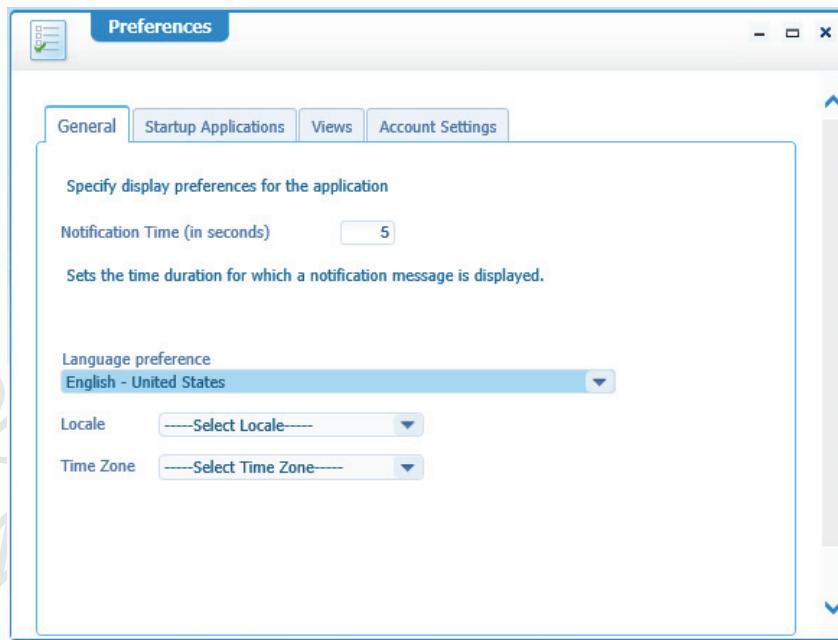


Figure 1-20:
User start page preferences (General tab)

The Process Platform user start page can be accessed by any user and based on the roles assigned to the user the allowed artifacts (applications) are displayed.



For your application, you can decide to integrate or offer Process Platform applications via your own customizations. In course 4-4913, Process Modeling for Process Platform, student developers build an application and add it to the palette.



Working with the shortcut bar

1. Use the search bar in the My Application palette to filter all artifacts which contain the word "Document".
2. Drag the Workspace Documents artifact from the palette to the shortcut bar.
3. Now add the New Document artifact to the shortcut bar.
4. Click the Preferences shortcut.
5. Under the General tab, select English - United States as the language preference, if it is not already selected.
6. Under the Account Settings tab, set the Full Name field to your full name.
7. Click Apply.
8. Click Close to dismiss the dialog box.
9. Click OK.
10. Refresh the browser to load the changes.

11. Use the pin in the shortcut bar to float the bar.



Figure 1-21:

**Anchor or float the
shortcut bar**

12. Test the floating option by moving your mouse in and out the shortcut bar area.
13. Set the shortcut bar to either fix or float to suit your favorite setting.

Task bar

When an application is opened from the palette, it is added to the list of open applications along the bottom of the Process Platform User Start Page in the Task bar. Open applications in the Task bar are displayed as individual tabs. You can switch between open applications by clicking on the appropriate tab in the Task bar.

By right-clicking on an individual application in the Task bar, you can close or hide the application, or show hidden/minimized applications. By right-clicking in the Task bar itself, you can either minimize all open applications or restore (i.e., "show") all open applications.



Figure 1-22:

**Applications minimized
in the task bar**



Working with the task bar

1. Press F11 (browser full screen mode) to allow more screen space for the user start page.
2. Start the Workspace Documents artifact from the shortcut bar.

The task bar along the bottom of the user start page shows all open applications.

3. Right-click in the task bar at the bottom, not on, but adjacent to the open applications tabs.
4. Select Minimize All Applications.
5. Right-click in the task bar again and select Show Open Applications.
6. Close Workspace Documents.

Summary

Having completed this chapter, you should be able to:

- Explain Process Suite and the elements of Process Suite
- Explain the role of Process Platform in Process Suite
- Explain the business drivers to start implementing Process Suite
- Describe the role of entity modeling in development with Process Suite Platform.
- Launch the Process Platform Explorer
- Navigate through the Process Platform Explorer
- Customize the look and feel
- Launch applications in the Process Platform Explorer

Open Text Internal Use Only
Do Not Distribute

Test your knowledge

1. What is an application palette?

2. What is an artifact?

3. When you open the User Manager application, which users do you see by default?

4. Which version of OpenText Process Platform is currently installed?

Answers to test your knowledge

1. What is an application palette?

An Application palette is a window from which you can open or select applications, documents etc. Examples are: My Applications, the Shortcut bar and your personalized view.

2. What is an artifact?

An application palette contains options that you can start, open, select, etc., like the User Manager and My Inbox in the My Applications App Palette. The generic name for each is an Artifact. In your own application you can also use this technique to select options instead of, for example, a drop-down box.

3. When you open the User Manager application, which users do you see by default?

By default, only your own user name is displayed when opening the User Manager. When you click the Show All button, you will see all the users in your organization.

4. Which version of OpenText Process Platform is currently installed?

The build information comes in three parts:

- *Version: 16.3*
- *Build: 2039*
- *Revision: 329539*

Exercises

1. Use the copyright information and installation details application to record the version and build of Process Platform which is currently installed.
-

This screen is also known as “About Process Platform” and OpenText Support may ask you to send the XML of this page, to collect all relevant details of your Process Platform installation.

2. Create your own shortcut for the shortcut bar.

In the shortcut bar, click the options icon and select Create Shortcut. Provide the following details:

- URL of the shortcut: <https://core.opentext.com/>
- Tooltip for the shortcut: **OpenText Core**
- ID: **OpenTextCoreShortcut**
- Caption: **OpenText Core**
- Icon URL: **wcp/theme/default/icon/Academy/core.png**

Make sure you test the shortcut by selecting it from the shortcut bar.

3. Add the Workspace Documents artifact to your Startup Applications preferences.

Drag the Workspace Documents artifact from My Applications palette and drop it into the Startup Applications area of the Preferences application. Refresh your browser afterwards.

To disable the application, click the disable icon in the Startup Applications tab of the preferences application.

Sign-out from Process Platform when you've completed these exercises.

Open Text Internal Use Only
Do Not Distribute

2. Entity modeling

Objectives

On completion of this chapter, participants should be able to:

- Create a workspace and project
- Define entities in an enterprise-level solution
- Describe entities and their role in a solution
- Create simple entities
- Run validation on a solution with entities

Overview

In the previous chapter, you learned that entity modeling is performed in Process Suite Platform (or “Process Platform”). There are many applications which are available in the Process Platform application palette, but you will not use them all in this course: in this course, you will concentrate on building entities as part of the overall enterprise solution.

Entities are created in a workspace. This workspace can be shared with other members of your team, or other teams, to build the complete solution. In this chapter, you will create a new workspace and project for your entities, you will learn about entities, and you will build some simple entities to support an enterprise solution.

What you will build in this chapter

- Entities: Product and its subtype PerishableProduct, Order, Customer and its subtypes Business and Person.

Timing

Lecture: 60-75 minutes

Exercises: 45-60 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform Entity Modeling Guide

Building solutions in Process Platform

Building a solution in Process Platform is a complete team effort, from business analysts to developers. The applications which are produced are then deployed and made available to end users in Process Experience. Application development is managed at three different levels:

- Software Configuration Management (SCM)
- Team collaboration
- Release management

Software Configuration Management (SCM) is meant to manage and secure the source code of an application. During development, software code exists in different versions at different systems. For this reason, it is important to know who is working on which part of the code. SCM systems usually provide version control and branching (i.e., each version can have its own branch).

Solution designers are not uniquely developers. Process Suite allows different teams with specific talents to contribute to the overall solution; this provides a true and complete end-to-end enterprise-level solution and reduces the overall amount of effort and time to market of solutions which are customarily attributed to only development teams.

Since multiple solution designers are working on a project concurrently, documents need to be synchronized. Once the application is released it is important to know which document belongs to which release, so releases must be labeled. Especially in a (loosely coupled) web service environment, (automated) testing is of great value, so broken links are found before they move into run time.

Once content is developed and ready for release, the content is bundled into one or more application packages (CAPs). The CAPs are used to transfer the developed solution from one system or environment to another.

Usually, the development team has access to a development environment where development is done, and a test environment, where the deployment of the release can be tested. Another advantage is that while release tests are performed developers can proceed with development on the next iteration or version.

When test results are positive, the package, including installation manual, is handed over to system administration. System administrators deploy the application package and perform an acceptance test of the release. Finally the package is deployed on the production server. Usually these releases are versioned on a versioning system.

Entity modeling as an ecosystem for solution development When you create an application with entities, there are some basic steps to follow:

1. Create a workspace and project in Process Platform.
2. Create an application with entities and entity building blocks.
3. Publish the application.
4. Test the application.
5. Package and deploy the application to a production environment.

Application development should be an iterative process. This means that there are several repeated cycles of design, development, and deployment. As the requirements of your application adapt and change in the design phases, the functions you require will subsequently change. It is important to note that not every entity will require every type of building block, but the entity modeling environment provides enough flexibility to allow you to create applications iteratively over several phases.

Consider the following diagram to develop your entity models:

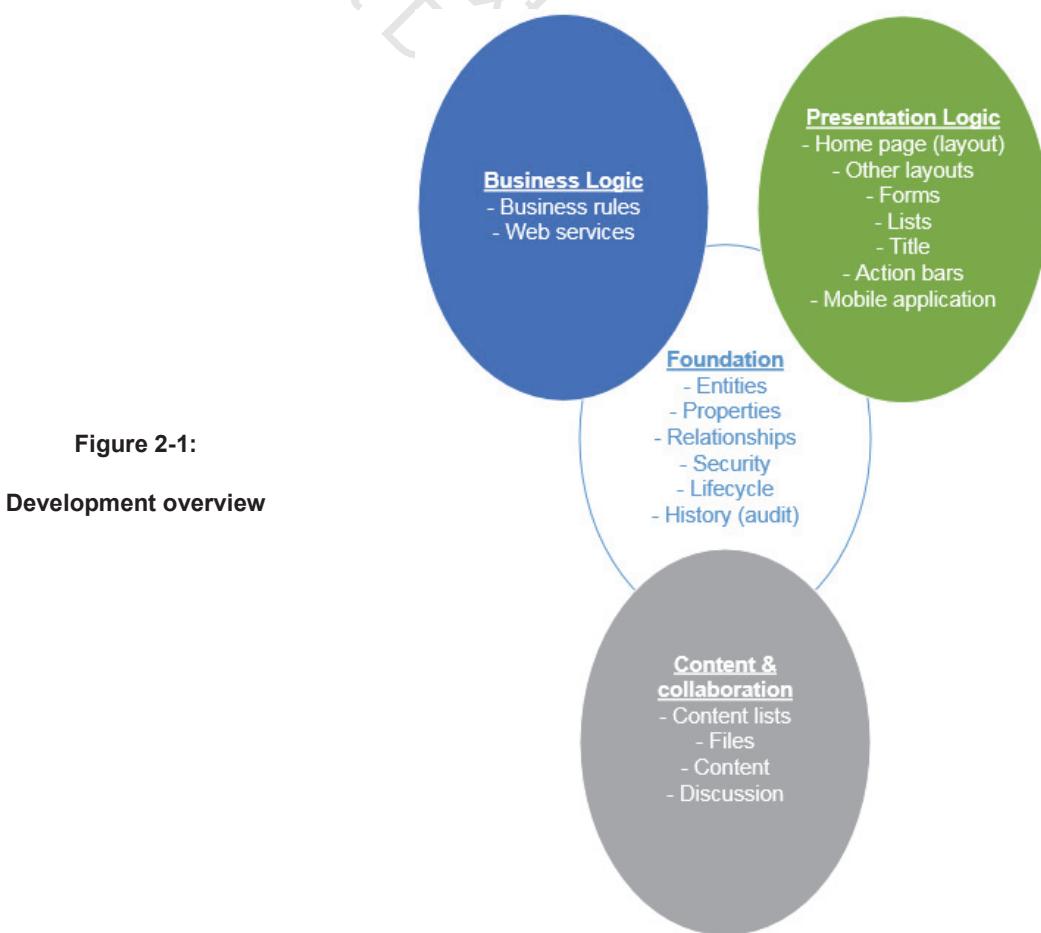


Figure 2-1:
Development overview

After creating a workspace and project, an application which employs entity models should, at a minimum, include the following:

1. Entities,
2. Properties,
3. Relationships,
4. Forms,
5. Lists, and
6. Item layouts.

After foundational elements are added to the application, the direction of the design and development phase can branch in other directions, as the preceding diagram suggests:

- Business logic phase: adding business rules or enabling web services
- Presentation logic phase: adding forms, layouts, action bars, etc.
- Content and collaboration phase (integration): adding discussion, files, content, etc.

The flexibility of the entity modeling environment permits design and development teams to progress as the application suits the business needs. During any phase, you can publish your changes and test them in Process Experience, as you will learn later in this course.

Multi-tenancy through organization management

Organizations

Users in Process Platform work in their own *organization(s)*. Organizations are used to separate application-specific content, where (organizational) users are given access to this content. There are two types of organizations:

System organization The System organization is used for maintenance and configuration of Process Platform. It is used exclusively by the system administrator.

Things that are managed in the system organization are typically:

- Process Platform licensing
- Other organizations
- Platform service groups like Process Platform Monitor, LDAP, and XML
- Shared application service groups
- Loading and unloading Process Platform application packages (CAP files)



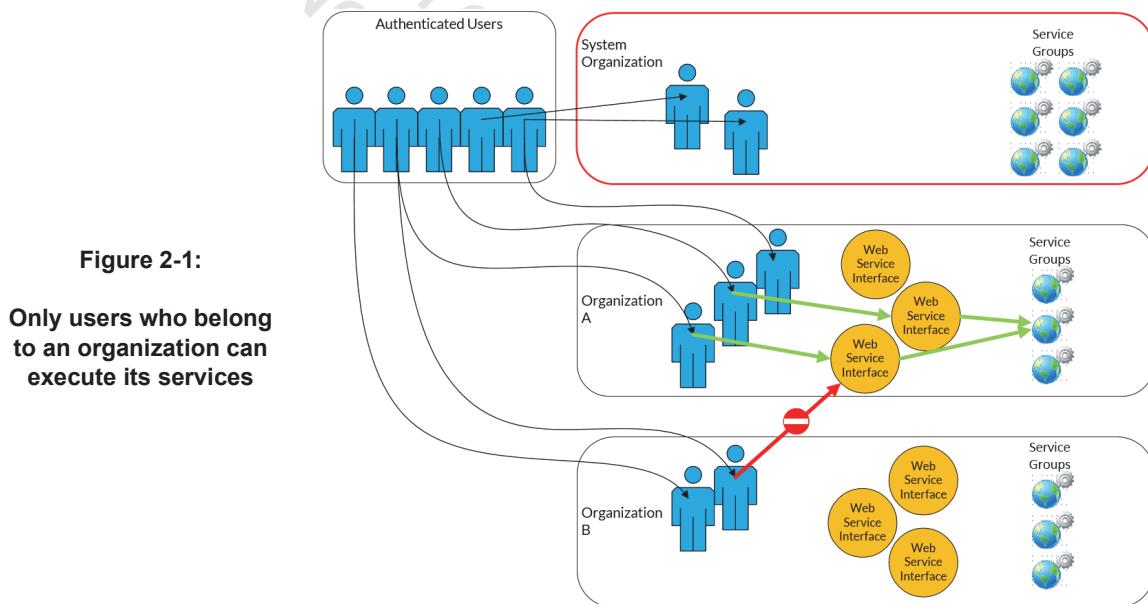
A service container is an object that executes a SOAP request. A service group is a group of related service containers and serves as the logical container to manage the separate service containers. You will learn about service containers and service groups in course 4-4913: Process Modeling for Process Platform.



Do not develop within the System Organization.

The System Organization should only be accessible to the system administrator. No one should ever develop in the System Organization. If anything happens to the System Organization, it may be impossible to recover from it. It is therefore recommended to develop in a normal organization, which your system administrator will create. Should anything go wrong, you can fix any other organization through the System Organization.

In the example below, organizational users cannot execute a service in another organization. A user can execute a web service operation in Organization A and B only if the user is authorized for both organizations. The administrator from an organization has to add a user to the organization before the user can execute Web service operations in that organization.



Process Platform shows:

- The organizations of the current user
- The roles and tasks assigned to the current user

The browser supports single sign-on through OpenText Directory Services (OTDS). This means that the user needs to only log in once: once they are logged in, they can access other organizations within Process Platform without needing to log in again. They can also access different components in Process Suite, such as Process Experience, without requiring a separate login.

In standalone web service calls, the URL contains the name of the organization. If no organization is specified, it will use the default organization of the user who is trying to log in. Authentication is required each time the method is invoked. Some applications allow caching of the user name and password, removing the need to provide credentials every time the call is made.

Development organizations Every other organization in Process Platform is a development organization. It is an organization created by the system administrator in which users and teams will collaborate in order to build solutions relevant to their organization. The system organization is included with Process Platform by default. Development organizations must be created by the system administrator after Process Platform is installed and configured.

Development organizations are specific to the business using them. They can be built along any logic which is relevant to the business, such as by department, project initiative, or geography. Users can belong to many organizations, as attributed by the system administrator. Users can play different roles in each organization, granting them access to specific artifacts on the user start page based upon their role in the organization. In this course, you are granted business analyst rights in the Warehouse organization. All your work in this course will be performed in the Warehouse organization.

Identity packages Whereas user management and authentication is managed in OTDS, identity package is used to manage Process Experience users, groups, and roles. Identity package is automatically installed with Process Suite and includes functionality to:

- manage the data for an organizational model,
- allow users with privileges to create certain organizational constructs,
- assign users to organizations, and
- build user and organizational hierarchies.

The identity package is a complete entity model with pre-constructed entities, properties, relationships, user interfaces, and other building blocks. The OTDS push connector is used to synchronize user information from OTDS to Process Suite. Run-time user management is handled in Process Experience through a pre-built identity package home page.



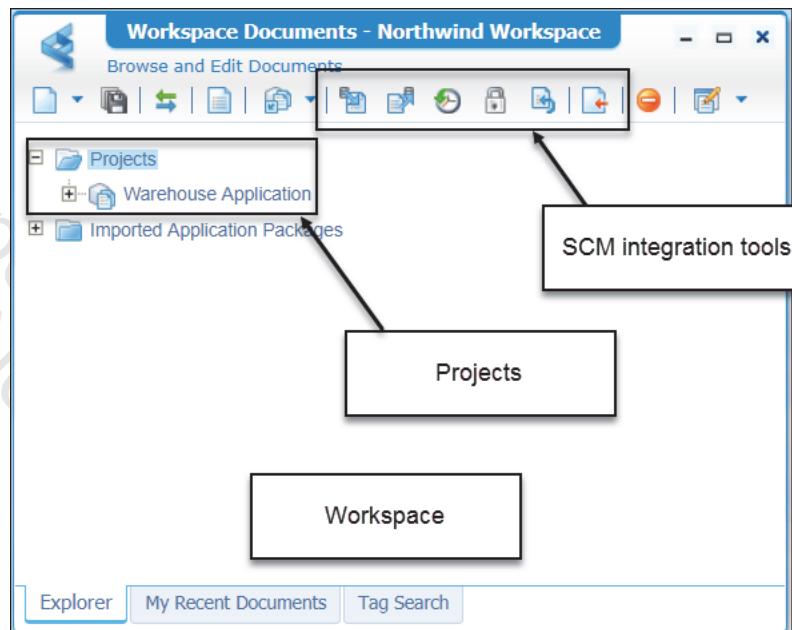
You will work more with identity packages later and in course 4-4913: Process Modeling for Process Platform.

Collaborative workspace

A collaborative workspace (CWS) is an isolated environment where a solution designer or group can develop their Process Platform content.

In the figure below, you can see an example of a workspace with one or more projects. In this case, the My Application Project.

Figure 2-2:
My Application Project in
a collaborative
workspace



In an organization, you can define several workspaces, however only one can be active at a time. A *workspace* is an isolated development environment in Process Platform. A workspace can be associated to a SCM system (e.g., SVN). Usually, a workspace connects to the URL of the source repository, where the documents for the solution are persisted. Within a workspace, one or more projects can be created.

CWS is a development environment used to build applications in Process Platform. A Process Platform application typically exists of entity models, business process models, user interfaces, services, etc. CWS provides a single design-time view of the development content for your applications and provides the functionality to test your content as well as validate and package your application.

The collaborative workspace supports content sharing among several solution designers, teams, and organizations. Furthermore, it supports association of a workspace with a SCM system, thereby supporting version control. Finally CWS provides a uniform look and feel of all the applications and editors.

A *project* is a container for all types of Process Platform documents that are delivered as a single CAP file. In fact, a project and an application are used synonymously. The project is one application on which you are working and should contain all components (entities, processes, forms, roles, etc.) that form the application you are building. You can have references to components in other applications for reuse. Projects are validated, published, and packaged into CAPs which are subsequently installed, tested, and deployed. A project can be used by multiple solutions within the same workspace.

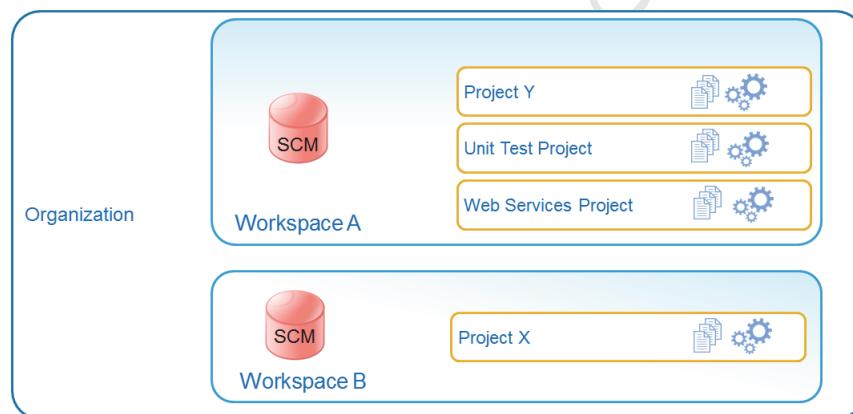
In CWS, a project corresponds to the application you are building. An application in this context is meant as a deliverable to users. For example, “expense claim” is an application to process expense claims.

Within a project, teams will create many types of “documents”, such as entities, business process models, services, user interfaces, etc. These together form an application. When the application is complete (i.e., all relevant “documents” have been developed and tested), you can package this project/application for deployment. As the whole content of the workspace is stored in SCM system, it is advisable to have only one project in your workspace. You can however add multiple projects that are related to your application, such as the test application you built to automate testing, for example. You can also include components and/or documents you want to keep with your application but not release to a customer.



Restrict your content to just one application per workspace. For example, the application itself, the test application, and documentation which you do not intend to release. Don’t mix content between applications in a single workspace.

Figure 2-3:
Workspaces and projects



- Types of content in projects** Each project can contain three different types of content:
- Process Platform documents (e.g., entity models, process models, rules, etc.)
 - Run-time references to existing Process Platform application content
 - External content (e.g., files, CSS, JavaScript, JAR files, HTML, images, etc.)

During development you have to decide on a certain folder structure. Process Platform groups its documents by document type. Other types of grouping could be by related content, or a grouping relevant to your business operation.

Workspace Documents have different views for displaying content. The default view is the Explorer tab. This view allows developers more control over the structure in a project like the folder structures and the location of documents within the workspace. In this course, you will use the Explorer tab as you will setup solutions, projects, and the deployment structure.

The My Recent Documents tab is preferred by business users as it does not deal with project, folder structures, etc., but only displays documents like processes and forms that have recently been used so they are easy to find and open.

Entities are a type of document in Process Platform, just like business process models. When you create an entity model in Process Platform, it must be added to a new or existing workspace. If the workspace is shared with a SCM system, then other team members will have access to it.



From this point onward in this course, all your development work should be performed as the *analyst@training.local* user in the Warehouse organization.

Solution overview

In this course, you will build content-rich entity models for Process Platform.

You are a business analyst for the Northwind trading company. Northwind takes orders from and delivers products to customers worldwide. In this scenario, you will build entity models to represent the Northwind warehouse system. You will need to develop content for:

- Products
- Orders
- Line items for orders

In some of the more advanced exercises, you may also build content for customers, suppliers, and other entities.

Solution workspace and project	Before you can begin creating entities for your solution, you must first create the workspace and project in which the solution will be contained.
---------------------------------------	--



Create a collaborative workspace (CWS)

1. *In the browser, click the Process Platform shortcut for the Warehouse organization in the Favorites bar.*
2. *Log in to Process Platform with the user name **analyst@training.local** and password **opentext** if you are not logged in already.*
3. *In the upper right corner of the user start page, confirm that you are using the Warehouse organization.*

If you are in any organization other than Warehouse, select Warehouse from the Organizations drop-down. Wait for the user start page to refresh when you switch organizations.

4. *If you did not do so in the previous exercise, create shortcuts for Workspace Documents and New Document in the shortcut bar.*

You will use these shortcuts frequently in this course.



Create a shortcut by dragging an artifact from the palette and dropping it onto the shortcut bar.

-
5. *Click the Workspace Documents artifact.*

The first time the Workspace Document Explorer is started it takes a while to load all the required components.

6. *Click the insert (+) button to create a new workspace.*
7. *Provide the following details:*
 - a. *Name: Northwind Workspace*
 - b. *Description: Workspace for Northwind applications*
 - c. *Source Control Management type: SVN for team development*

8. Click **Next**.



In a production environment, it is recommended that you create workspaces in an SCM system. In this way, workspaces can be shared with other team members.

In your course environment, an SVN repository has been pre-built in the local folder. In a true production environment, this would likely be on a centrally located server in the corporate intranet. However, this environment cannot be achieved in an isolated course environment because the virtual machine is completely self-contained.

The SVN URL used for a workspace cannot be changed after the workspace has been created. Therefore you should not use an IP address in the SVN URL.

9. Complete the source control repository properties:
 - a. URL: file:///C:/SVNRepository/bpm
 - b. Username: sysadmin
 - c. Password: opentext
10. Click **Test Connection** and ensure that you are able to connect to the local repository.
11. Click **Next**.
12. Fill in the properties of the project, providing the following values:
 - a. Project Name: Warehouse Application
 - b. Package Owner: Northwind
 - c. Product Name: Warehouse Application
13. Click **Next**.
14. Review the summary and click **Create**.
15. Click **Close** when the workspace has been created successfully.



The project name is only to visualize the project level in the workspace folder tree.

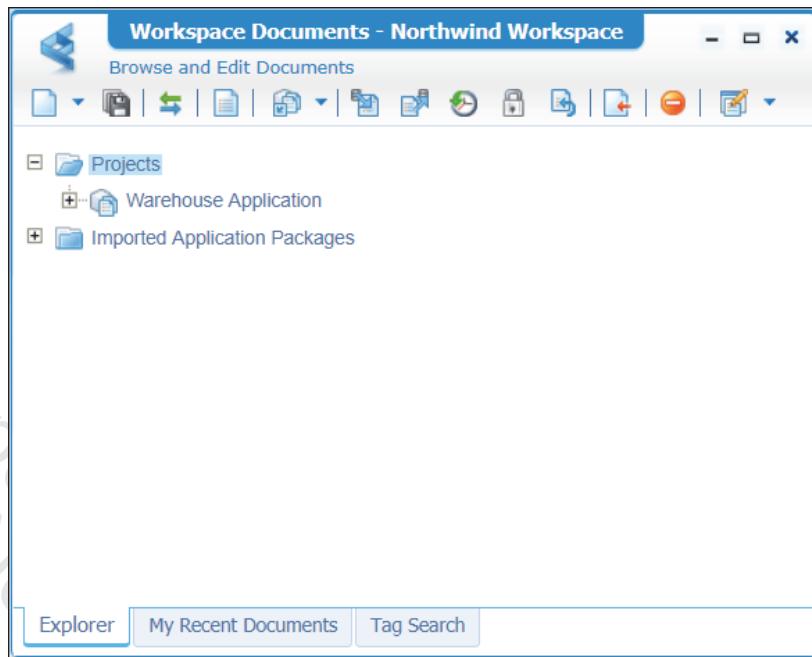


Figure 2-4:

Warehouse Application
project in the Northwind
Workspace



Create a folder structure for your project

1. Right-click the *Warehouse Application* folder and select *New > Folder*.
2. Type “*Entities*” for the name.

This folder will contain application specific entities. Other development teams may build more folders into this project to contain documents such as business process diagrams, case models, business rules, and so on.



In Process Platform, the folder in which a document is placed can be significant. In course 4-4913: Process Modeling for Process Platform, you will learn how to create a folder structure and set the starting points for qualified folder names.

Once you have created a workspace and project, you are ready to begin developing entities. The folder is only necessary to easily organize your work.

Building entities

In Process Platform, building a solution can be performed in different methods or directions. These directions are typically data-first or process-first (i.e., data-driven or process-driven). In the process-driven design direction, development teams work with business analysts to characterize business processes and/or cases, usually from business use cases. In the data-driven design approach, however, the solution designer will build entity models to capture relevant business objects without requiring any development experience. Process Platform is a flexible solution: teams can build parts of the solution in different directions simultaneously and in parallel. The flexibility and openness of Process Platform allows organizations to choose which approach to start with.

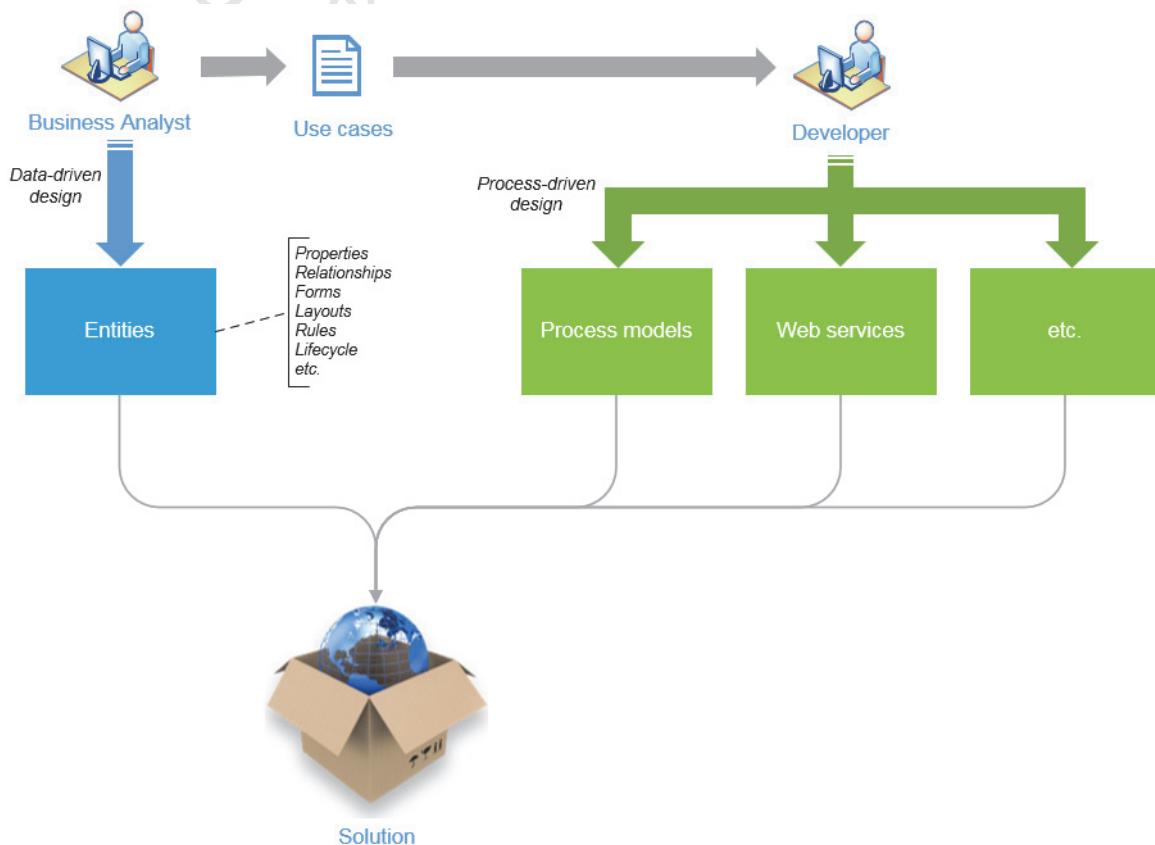


Figure 2-5: Data- and process-driven design

Entity modeling In Process Platform, an *entity* is a representation of a business object used in data-driven design. As you will learn in the upcoming chapters, entities will have several properties relevant to it. When you are building your solution from an entity, you will use these properties to help create other functional components, such as lists, forms, layouts, and more. You should consider whether each piece of information in the business object is a property of an entity, or an entity itself. In general, this depends upon the information and whether that information will be unique for each entity. Some information may even be included in a global design to be shared between several entities. An important part of modeling an entity is planning how entities will be related. Later in this course, you will design entity relationships.



It is a recommended practice to prepare a model or diagram of the entities, their properties, and relationships before you begin building them in Process Platform.

In the solution scenario which you will build for this course, you will build entities to support the business objects Product, Order, OrderLine and their respective children. A product and an order will each have a unique identifier (e.g., a product number and an order number), but they will also have shared information: each line of an order (i.e., the OrderLine) will refer to an Order and a Product, as depicted in the diagram below.

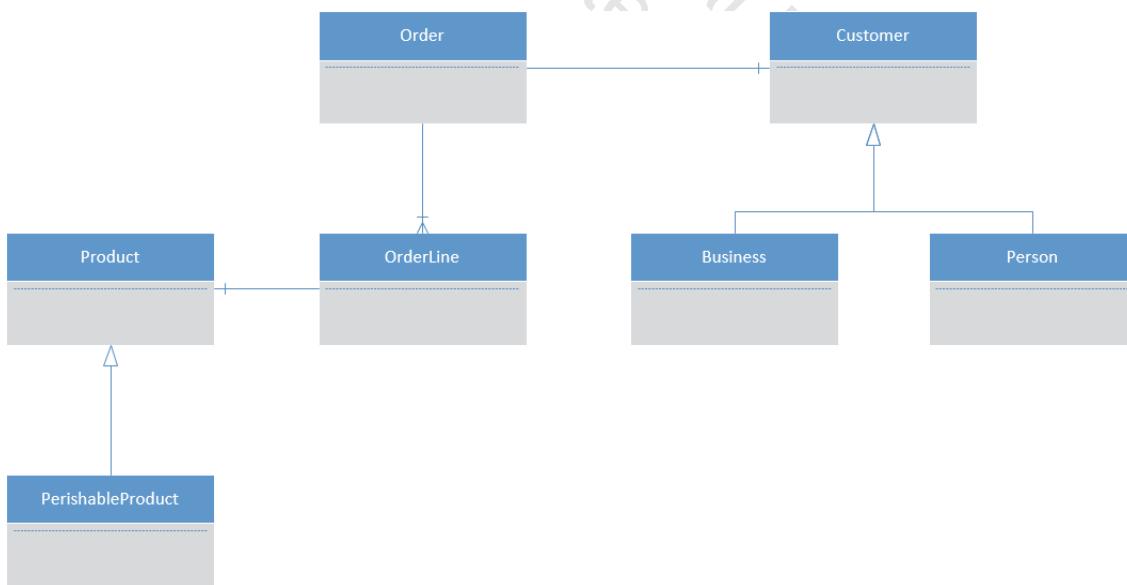


Figure 2-6: Relationship of business objects in solution scenario

In the preceding diagram, the Customer is represented with an association to the Order entity. Without a Customer object, this could mean several things for defining the Order entity:

- Customer is a text field. If the Customer were a text field, then end users would need to type the Customer name in every time an order is placed. This may result in one customer having several different entities, and difficult consolidating all their orders together. It would also mean that other Customer fields, such as phone number and address, would necessitate changes to the design.
- Customer is a drop-down field. In order to prevent disparate entries for a single Customer, you could restrict the entries by making the Customer a drop-down field. This would alleviate any misspellings and make consolidating orders easier, but does not resolve the problem of adding phone numbers, addresses and other Customer-specific properties. Furthermore, if new Customers are added, it would require changing the model, which is not flexible or open.

Ideally, the solution in this circumstance is to create a Customer entity which contains all the relevant fields, has a unique Customer ID, and is related to Orders. This is the best, most logical solution in terms of building entities.

Building an entity in Process Platform

It is the recommended practice that you model your entity solution by producing diagrams similar to the one presented earlier before you begin building entities in Process Platform.

In Process Platform, an entity is a document which is added to an application project in a CWS. As previously discussed, Process Platform offers several different types of documents, such as business process diagrams, case models, business calendars, and so on. As you have learned, you must have a workspace with an active project first, before you add documents to it.

When you add an entity to your project, you must provide the entity with a valid name. Each entity will have two names: the Entity Name will serve as the unique ID of the entity in the Project. The Display Name is the way users will see the entity. It is a good practice to make the display name something which is simple, user friendly, and easy to recognize.



The name of an entity may contain any alphanumeric character and an underscore, but no other special characters and cannot begin with a number. An entity name can be no longer than 128 characters.

Identity and primary key Once you create an entity, Process Platform will automatically assign it a *primary key*. This primary key is called an *Identity* building block. A primary key is a field which helps to uniquely identify each and every instance of an entity. Primary keys are unique: they cannot have any duplicates within the system.

When you are designing entities, all entities will have an Identity building block. This is to keep all entities and their instances unique.

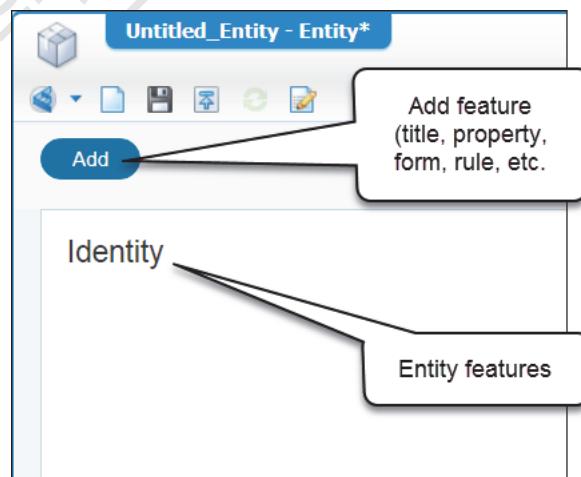
In practice, however, there are four distinct parts to the Identity:

- Id
- ItemId
- EntityType
- ItemStatus

These parts will be discussed later.

Entity Designer The Entity Designer is a simple interface in which builders can create and work with entities.

Figure 2-7:
Entity Designer



The entity's features are captured in a list in the editor. Features are grouped by type. The Add button is a generic button which a builder can use to add features or “building blocks” to the entity. There are many building blocks which can be added to an entity, which will be covered in this course, arranged in functional groups.

Types of entity building blocks (by functional group)

- **Basic or Foundation** building blocks:
 - **Property**: a characteristic or attribute of an entity.
 - **Relationship**: describes the connection between two entities.
 - **History**: adds audit capability to an entity, granting you the ability to track changes to all the instances of the entity (its history log).
 - **Identity**: a unique identifier, or primary key, for the entity. This building block is added and set automatically by the entity design tool.
 - **Security**: grants or restricts access rights to manage control of entities based on a user's security role.



Identity is not the same as identity package.

An *identity* is the feature of an entity which uniquely identifies the entity from other entities, the name for which functions as a primary key in the underlying database.

An *identity package* is a construct for organizing and arranging users and groups synchronized between Process Platform and OTDS.

You can also include in the foundation group those building blocks which manage process flows:

- **Lifecycle**: provides a collaborative process of assessment, planning, facilitation, coordination, evaluation, and advocacy for options and services to meet an entity's case management requirements. The lifecycle helps you manage the wide variety of business operations that are unstructured, ad hoc, dynamic, and not automated.
- **Activity flow**: defines a business process based on a grid structure so business users can manage the processes on their own.
- **Assignee**: specifies the target for e-mail notifications when a case is designated to an entity. Includes the e-mail template to use for the notification.
- **Tracking**: adds information to an entity to track when and by whom the item was created and last modified (i.e., audit).
- **Business logic** building blocks:
 - **Deadline**: assigns a schedule for completing a case.
 - **Rule**: specifies business logic or actions to be performed.
 - **Web Service**: expose web service operations on the entity. This permits the entity to work with other components in Process Platform, such as business process models.

- **Presentation** building blocks:
 - **Action Bar**: an aspect of a user interface which contains buttons which allow the user to upload, download, or perform other actions.
 - **Form**: a user interface in Process Experience for end users to work with the instance details.
 - **Layout**: specifies an overall presentation of the entity instance. Many layouts can be used to capture different states of an instance.
 - **List**: organize work tasks in Process Experience.
 - **Mobile App**: identify elements of your entity-based solution which will form part of a mobile micro-app. These elements will be published to a mobile application which can be deployed in OpenText AppWorks.
 - **Title**: adds a title that can be used to help identify entity instances.



OpenText AppWorks must be purchased separately and properly installed and configured before the Mobile App building block can be used. For the purposes of this course, the Mobile App building block will not be used.

- **Content and collaboration** building blocks:
 - **Business workspace**: enables end-users to perform all the document management operations supported by the business workspace and folder browser UI widgets of Content Server.
 - **Content**: allows users in Process Experience to add or remove attachments to/from an item.
 - **Discussion**: attach a message board to an entity. This will allow end users in Process Experience to collaborate in the entity instance.
 - **Email and Email template**: provides email functionality in your applications accompanied with the templates to send emails.
 - **Inbox**: displays items from a user's Process Platform inbox in a Process Experience list.

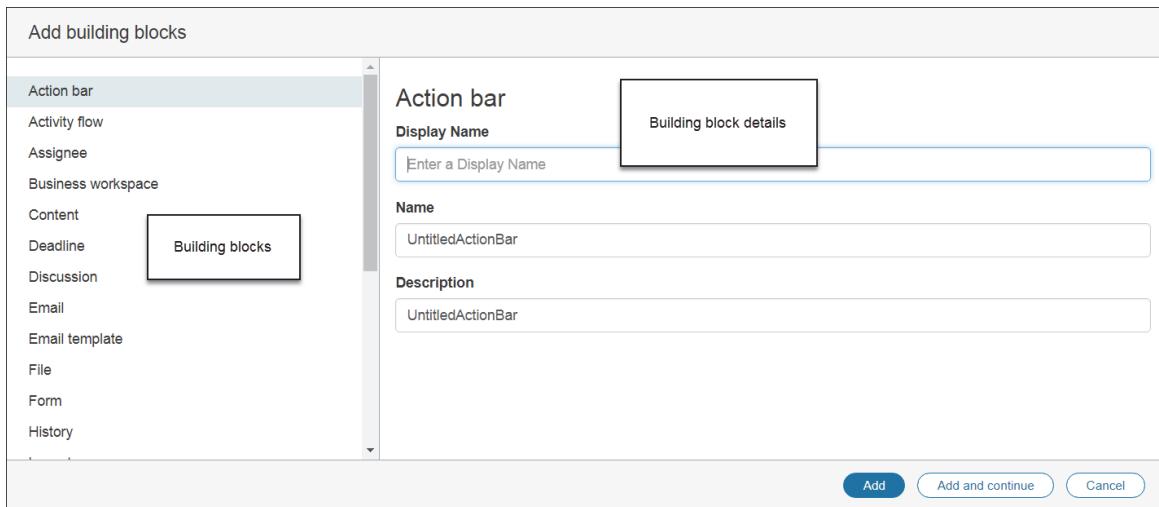


Figure 2-8: Add building block dialog for an entity

As you select a building block to add, the right side of the dialog will change in order to accurately reflect the relevant details of the feature which will be added. In the preceding diagram, an Action Bar will be added to the entity. On the right, relevant attributes for an Action Bar include its name, display name, and a description.

There are a number of function buttons which are also available in the entity toolbar:

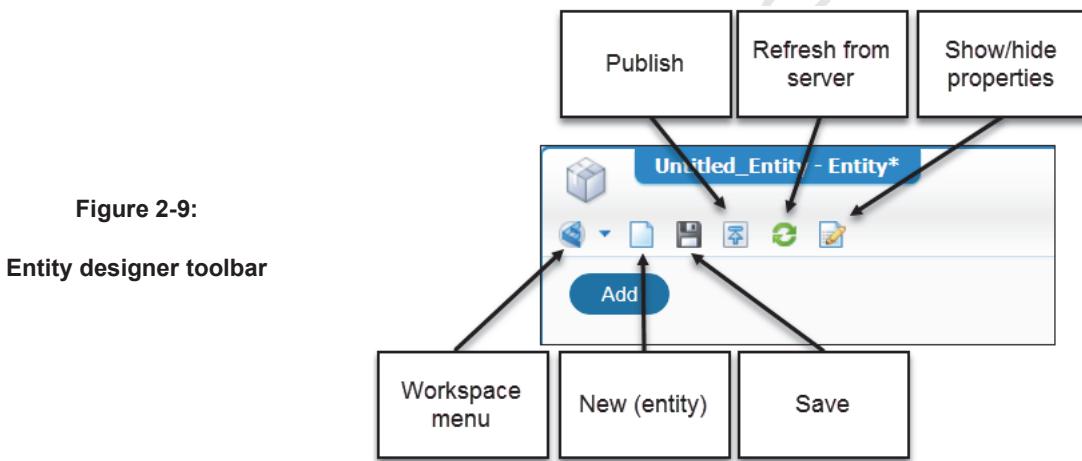


Figure 2-9:

- The **Workspace Menu** is a common tool for all documents in Process Platform. With the Quick Access Menu, you can create any new document, open documents, validate and publish models, determine which documents use your entity (Used By), and list the Properties of the document. You will use this menu more later in this course.
- The **New** button creates a new entity. Each entity will open in its own window.

- **Refresh** will force the document to refresh from the server. If you have been editing the entity document elsewhere in Process Platform, you will need to refresh it regularly in order to keep your document current and avoid data conflicts.



Refresh is not the same as checking out from a Source Control Management (SCM) system.

- **Show/Hide Properties** will toggle a display of a property and its attributes on the right side of the Entity Designer. By default, properties are hidden when an entity is created.
- **Publish** takes the development-time artifacts and publishes them to the server as runtime components. You must publish your changes before you are able to test them in the runtime.



Build a Product entity

1. Log in to Process Platform as the **analyst@training.local** user, if you are not logged in already.
2. Click **Workspace Documents** from the Shortcut bar and open the **Northwind Workspace** if it is not already open.
3. In the list of Projects, expand the **Warehouse Application** project.
4. Right-click the **Entities** folder and select **New > Other**.
5. In the New Document dialog, type **entity** in the search bar.

Typing entity in the Search bar will filter all the different types of documents.

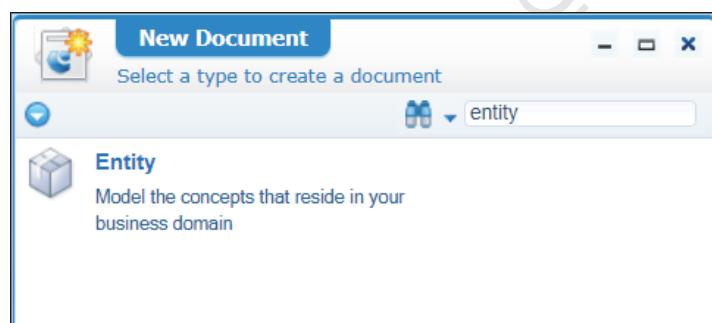


Figure 2-10:

Filter New Document dialog for entity type



The first time you select **New** from the right-click context menu, you must select **Other** in order to find the entity document for which you're looking. Each subsequent time you want to add an entity to the project, it will be available to you under the **New** content menu.

6. Click **Entity**.

The Entity Designer appears. Remember that an Identity is added by default to every new entity.

7. Click **Show/Hide entity properties**.

8. In the **Entity properties** section, set the **Display name** and **Name** to **Product**.



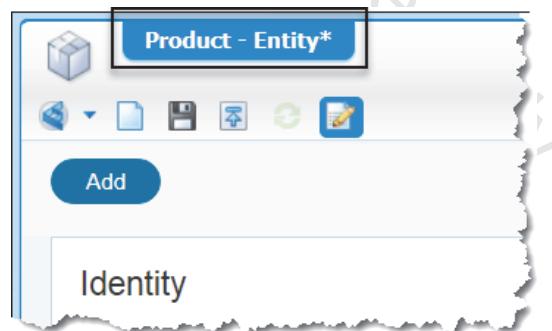
You may observe that, for every subsequent entity which you build, as you begin typing the display name, Process Platform offers you a drop-down of previous entries (e.g., Product). This is done for translation purposes; Process Platform has stored this name, allowing you to select it instead of re-typing. When your solution is translated into other languages, Process Platform can easily find all instances of the same translatable text and replace it with its matching translation. Display name properties in Process Platform can be translated: Name fields are not.

9. Click **Save**.



It is important to save your work regularly. If there are unsaved changes to a document, Process Platform marks them with an asterisk in the title bar.

Figure 2-11:
Unsaved changes
marked with an asterisk



The “Available for use by others” check box allows entities to be shared between projects. That is, you can create an Employee entity in the Warehouse Application and then make it available to other projects, such as a Human Resources Application.

You will learn about the “Supertype” and “Abstract” fields later in this chapter.

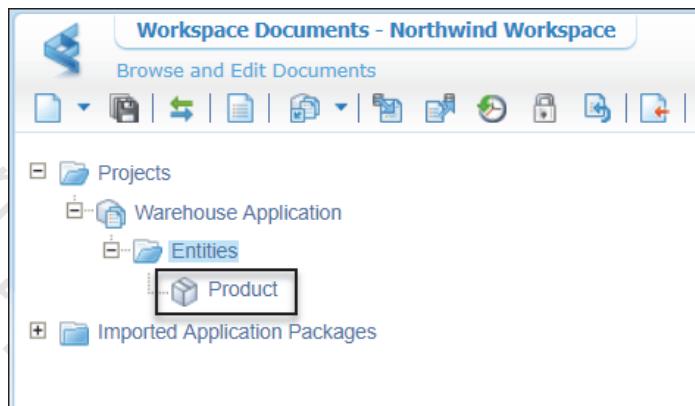
10. Click *Identity* in the left section.

The Identity's properties are opened in the right section of the window. Every entity has a mandatory Identity with a default name: Identity. This name cannot be edited.

11. Close the entity.

The Product entity is listed inside the Entities folder in your project.

Figure 2-12:
Product entity in the Entities folder



Creating vs. importing entities

When designing a solution, you can create your own entities "from scratch" or import entities from another system through an established EIS connection. It is often a preferred practice to first import entities from another application and then create ad hoc entities as required.

After importing entities from an external system, you can decorate them with entity building blocks as you might a new entity, such as the inclusion of lists, forms, layouts, and so on. However, since these entities are not generated from Process Platform, you cannot change their underlying structure (i.e., properties, relationships).

In order to import entities from an external system, you must first establish an EIS connection to that system. This requires modeling the metadata and configuring the connection: an undertaking which may require some initial effort.

In this course, you will learn the basics of entities from the ground up. Later in this course, you will learn how to import entities. It is important that you familiarize yourself with the Process Platform environment and entity modeling capabilities first before you attempt importing entities.

Validating a solution

The entity that you just created is in development, but is not available in the run time. Before any solution in Process Platform can be used in the Process Experience run time environment, it must go through a two-stage process: validation and publication.

The validation part of a solution is to ensure that all the essential properties and attributes of your documents are complete. Validation is run on an individual document, a folder of documents, or on a whole project. When running validation on a folder or project, Process Platform validates all the documents in that folder or project. Only those documents that have been validated are ready for publication to the executable run time. In a later exercise, you will learn how to publish your entities to the Process Experience run time.

There are many ways in which a designer can validate documents or projects:

1. From the Workspace Documents window. Select the document, folder, or project, right-click to call its context menu, and click Validate.

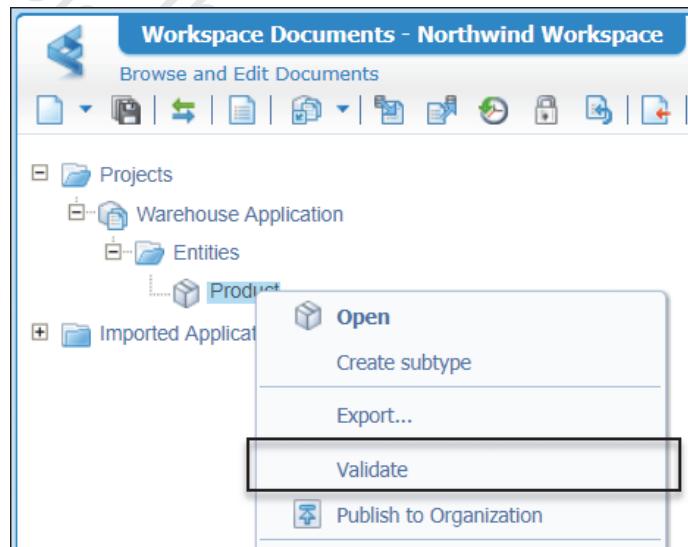
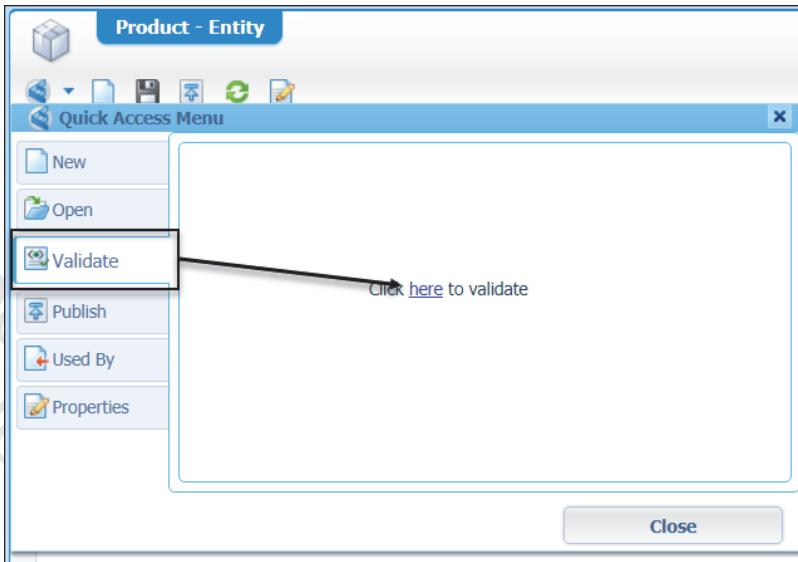


Figure 2-13:

Validation on documents, folders, or projects from Workspace Documents window

2. From the Quick Menu of the document window. With the entity document open, click the Workspace Menu icon in the toolbar and select Validate. Click the hyperlink to validate.

Figure 2-14:
Validation from workspace menu



3. Some documents, such as business process models, offer a context menu in the editor. With the document window open, right-click anywhere in the editor and select Validate. The Entity Designer does not offer a context menu in the editor.



Validate and commit the Warehouse Application project

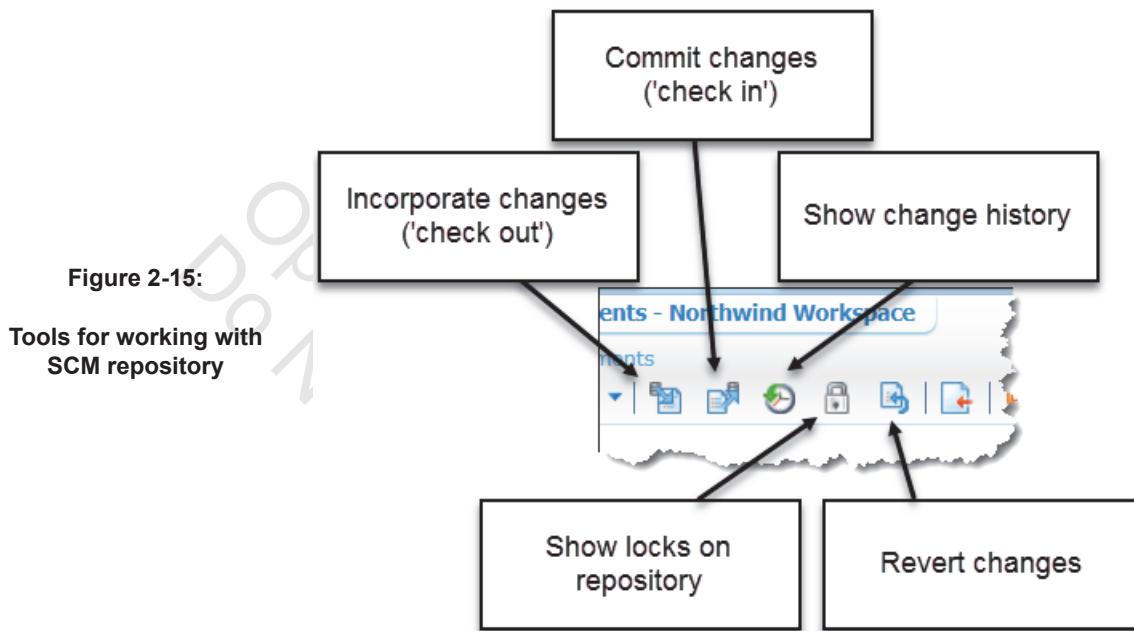
1. Open the Workspace Documents application if it is not already open.
 2. Right-click the Warehouse Application project.
 3. Select Validate.
- Validation should complete successfully.
4. Click Close.



It is a recommended practice to validate your project and documents regularly, particularly after creating or changing a document.

At specific points while you're building an application in a SCM system, you will need to make your changes available to other members of your team. This process is commonly referred to as *committal* or "checking in".

Since projects may be part of an SCM repository, you will periodically need to check your changes in to the repository for others, and consequently check out the project from the repository so you may collect changes other team members have made.



- **Incorporate changes from others** ('check out'): compares the latest version in the repository with the local version and if the repository version is newer, overwrites the local version with the changes from the repository.
 - **Make changes available to others** ('check in'): creates a new version in the repository and adds changes from your local copy to the version.
 - **Show change history**: reads the repository and displays the list of versions, the ID of the developer who committed them, and the date/time of committal.
 - **Show locks on repository**: projects which are locked in the repository may not be changed (versioned/checked in) until the lock has been released.
 - **Revert local changes**: undo all local changes and revert to the version which was last checked out. This is not the same as a new check out.
5. Commit your changes to the repository. Click Make Changes Available to Others () in the workspace toolbar.

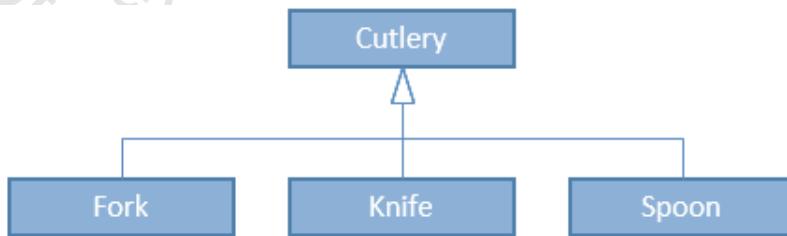
Changes will be synchronized with the repository.

6. In the committal dialog, enter a comment: "My first entity".
7. Click **Make Available**.
8. Click **Close** once the changes have been committed.
9. If you wish to review the changes in the repository, click **Show Change History**.

Entity types and subtypes

In some circumstances, entities may share many of the same properties and behaviors, and yet differ in specific and unique ways. By and large, the two entities could be either synonymous or hyponymous with one another. For example *fork* and *knife* are synonymous entities (alike but different) and *cutlery* and *spoon* are hyponymous entities (one is general, the latter more specific). These form logical, hierarchical structures, between parent, child, and sibling objects.

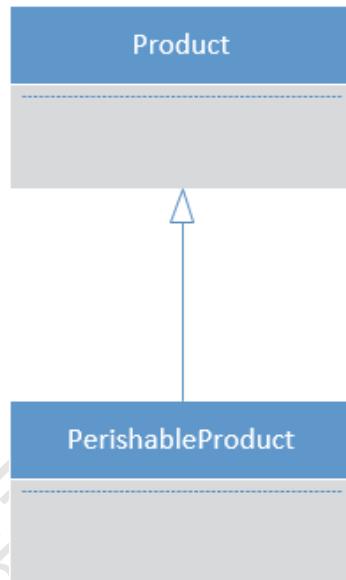
Figure 2-16:
A simple object hierarchy



You can model entities in Process Platform in similar hierarchical structures. The advantage to building a hierarchy is that you can push common aspects of an entity into a parent (the *supertype*) and model only the specific aspects in a child (the *subtype*). These are processes referred to as *generalization* and *specialization*. Furthermore, you can restrict which instances an end user can create in Process Experience by designating a supertype as *abstract*: when setting a table, you select a fork, knife, or spoon, not 'a cutlery'. Abstract supertypes are generalizations which contain all the common properties of its specialized subtypes. However, end users will never create instances of abstract supertypes, an instance is always of a subtype. The common ancestor provides a convenient and efficient model for capturing like building blocks, such as properties, forms, layouts, or rules.

For example, consider the entity which you just built: Product. A company may sell physical items, which possess stock keeping units (SKU), but they may also sell special types of products, such as perishable products. It may be argued that, for reporting purposes, Product is a superclass and PerishableProduct is a subclass, as shown in the diagram below:

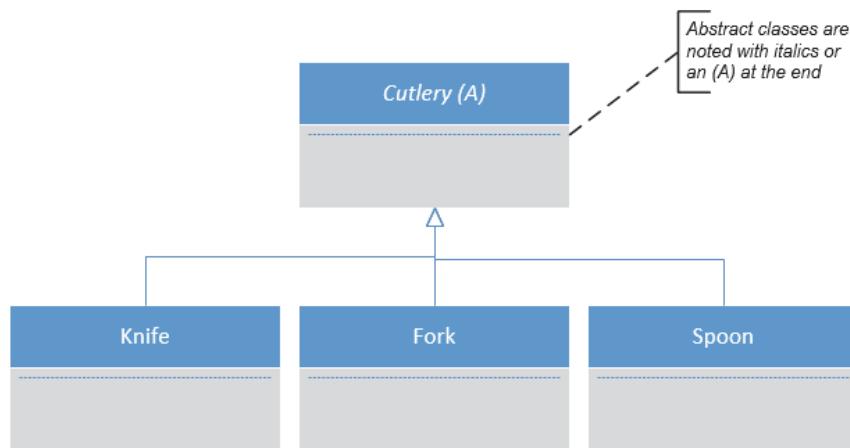
Figure 2-17:
PerishableProduct
inherited from Product



Abstract supertypes In the example above, the company sells products, some of which are perishable. In this example, all products are physical entities which are measured, audited, and sold.

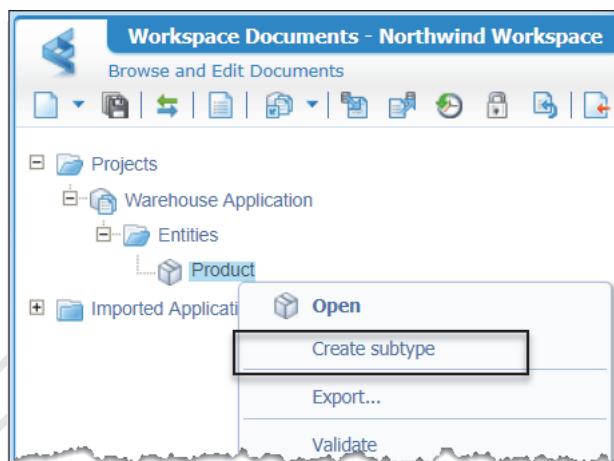
If you consider the class diagram earlier in this chapter which presented subtyping, there was an entity, Cutlery, which was the parent to fork, knife, and spoon. In this example, Cutlery is an *abstract supertype*; that is to say, one will buy a fork, knife, or spoon, all of which is considered cutlery, but one cannot buy a generic piece of cutlery.

Figure 2-18:
An object hierarchy with
an abstract supertype



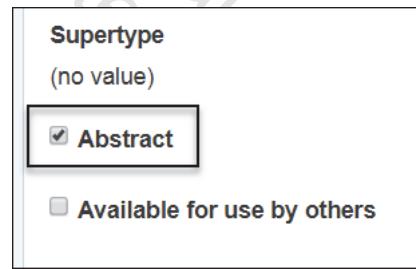
Creating an entity subtype In order to create a supertype/subtype hierarchy, you must first create the parent, then create the child/children. Subtypes are created exclusively in the workspace by right-clicking the parent entity and selecting Create subtype. In the example using Product, PerishableProduct is a subtype of Product..

Figure 2-19:
Create a subtype from a parent entity



If the parent is intended to be abstract, you must open the entity for edit, show its properties and select the Abstract check box. In this example, Product is not abstract.

Figure 2-20:
Setting a supertype to be abstract



Create a subtype of Product

1. In the Northwind workspace, right-click the Product entity and select Create subtype.

The entity editor dialog appears.

2. Show the entity properties if they are not already present.
3. Set the Display name and Name to PerishableProduct.
4. Save your changes.

In the save dialog, make sure the name and description are PerishableProduct and use the browse icon to set the Location to Warehouse Application/Entities. Click OK.

Figure 2-21:
Save dialog



5. Close the editor.



Despite PerishableProduct being a child of Product, the two entities will be displayed with equal precedence in the workspace. This is because the workspace portrays the physical structure (on the disk) and not the logical structure. Product and PerishableProduct are the same type of document in the Entities folder. If you open the PerishableProduct entity and examine its properties, however, you will notice that Product has been named the supertype.

Summary

Having completed this chapter, you should be able to:

- Create a workspace and project
- Define entities in an enterprise-level solution
- Describe entities and their role in a solution
- Create simple entities
- Run validation on a solution with entities

Open Text Internal Use Only
Do Not Distribute

Test your knowledge

1. What makes a workspace “collaborative”?

2. What is the common name given to objects you build in Process Platform (for example, entities, business process models, case models, business rules, etc.)?

3. What are the two types of organizations in Process Platform? What are the differences between them?

4. Which property or building block is automatically added to a new entity? What is the significance of this property?

Answers to test your knowledge

1. What makes a workspace “collaborative”?

Each workspace is stored on Process Platform server according to its organization. If a Source Control Management (SCM) system is installed, the workspace can be checked into the SCM server and checked out by other team members who have access to it. In this way, other users can contribute to the contents of the workspace in a global initiative and share work with their teammates anywhere in the world.

2. What is the common name given to objects you build in Process Platform (for example, entities, business process models, case models, business rules, etc.)?

Documents.

3. What are the two types of organizations in Process Platform? What are the differences between them?

System and Development.

The System Organization is included by default with every Process Platform installation. It is the single organization which a system administrator will use to manage information relevant to the entire Process Platform system. It is not intended for development, nor is it intended for any user other than the system administrator.

Development organizations are created by the Process Platform system administrator in order to perform development work. These organizations are created specific to each business. Users are granted access to these organizations by the system administrator to perform certain duties (e.g., developer, deployer, analyst, etc.) relevant to the organization.

4. Which property or building block is automatically added to a new entity? What is the significance of this property?

Identity.

The Identity building block functions as a primary key to help uniquely identify each instance of the entity. Examples of an Identity may be Product Number for Products, Order ID for Orders, Social Security Number for People, Employee ID for Employees, etc.

Exercises

1. Create the Order entity.

Earlier in this chapter, you were provided a data diagram portraying the relationship between three business objects: Product, Order, and OrderLine. You have created the Product entity, now, create the Order entity. You will create OrderLine in another chapter.

2. Create a Customer entity hierarchy.

Customer is a supertype which contains two subtypes: Business and Person.

3. Validate the Warehouse Application after you save your entities and commit your changes to the repository with the description: "Chapter 2 - Basic Entities".



Tips, tricks, and traps

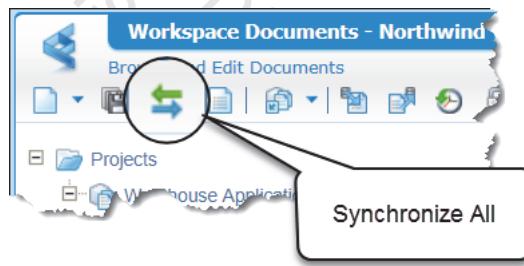
1. Creating a physical copy of your work

Committing your work to an SVN repository is invaluable: not only does it share your work with other team members who are connected to the same repository, but it also creates a physical copy of your work so that it can be retrieved in case of a system failure.

Process Platform is a database-driven environment. This means that whenever you work inside a collaborative workspace (CWS), your work is saved in the tables of an underlying database. This database is installed and configured with Process Suite. Committing your work to the SVN repository creates a physical copy of your work from the underlying database and posts that physical copy. It is possible to create a physical copy of your work without posting to a SVN repository.

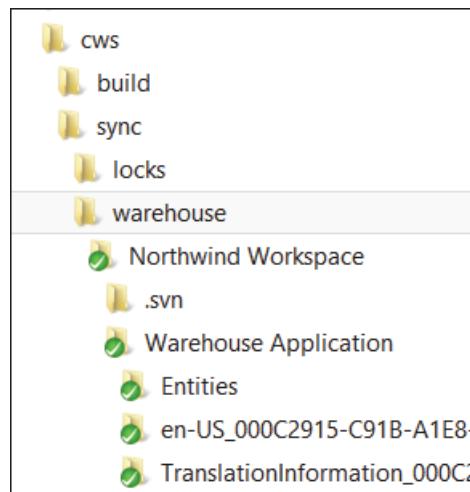
- In the Collaborative Workspace, click Synchronize All. Synchronize All will create a physical copy of your work and store it in a local directory. Usually, this directory is:

[ProcessSuiteInstall]\[InstanceName]\cws\sync\[organization]



In your solution material, your work will be stored in the following instance directory:

C:\Program Files\OpenText\ProcessPlatform\Development



3. Entity properties and relationships

Objectives

On completion of this chapter, participants should be able to:

- Describe the properties of an entity
- Add properties to an entity in Process Platform
- Define an entity's relationships
- Add a relationship to an entity
- Add a title to an entity

Overview

In the previous chapter, you learned how to model business objects into entities in Process Platform. Each entity has a unique identifier - an identity building block. In this chapter, you will examine more building blocks: property, relationship, and title, the significance of them to entities, how to add a property, a relationship, and a title to an entity, and strategies for designing a data-driven solution which includes entities, properties, and relationships.

What you will build in this chapter

- Property building block: you will add the Property building block to your entities. Each entity may contain several properties.
- Relationship building block: you will connect two entities to one another by adding the Relationship building block. Relationships may be described as peer or parent/child.
- Title building block

Timing

Lecture: 90-105 minutes

Exercises: 75-90 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform 16.3 Entity Modeling Guide

Building blocks

In the previous chapter, you were introduced to a diagram which demonstrated the relationship of three business objects to one another: Order, Product, and OrderLine. These three business objects were modeled as entities. As you learned, each entity in Process Platform has an identity: a characteristic which will help uniquely identify each instance of the entity.

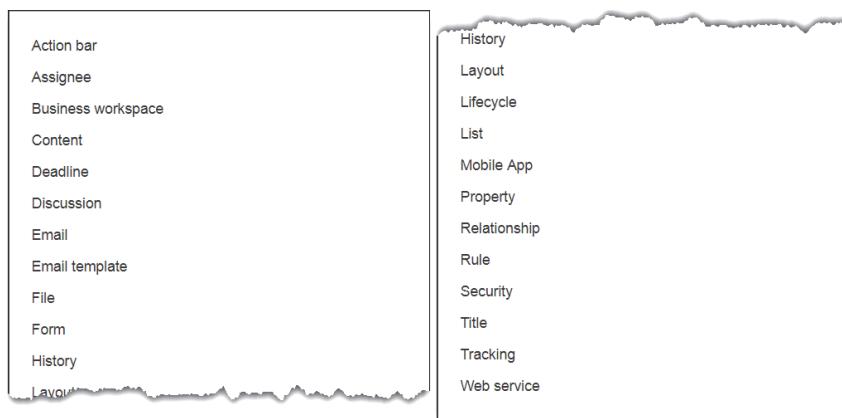
As the diagram in the previous chapter presented, each entity also had other characteristics which were non-identifying, but still relevant to the entity, such as price is to Product or order date is to an Order. An identity must be unique for every instance of an entity, but these properties do not need to be: it is possible, and quite likely, that there will be several Orders with the same order date, or many Products with the same price. Identity and property are *building blocks*: features which can be added to an entity in order to give it more substance and relevance in a solution.

As well, in the previous chapter, you learned that identity is a mandatory building block which is automatically added to every new entity you create. Property is another such building block which is added to an entity, but they are not added automatically: you must add the property building blocks to your entities manually. Each entity may have several properties: there is no theoretical limit to the number of properties an entity may have, but there is a logical limit.

There are several other types of entity building blocks which you will learn about in this course, but these were divided into different categories in the previous chapter:

- Foundation blocks (identity, property, relationship, history, etc.)
- Process flow blocks (lifecycle, activity flow, assignee, etc.)
- Business logic blocks (deadline, rule, web service)
- Presentation blocks (action bar, form, layout, list, etc.)
- Content and collaboration blocks (content, discussion, email, etc.).

Figure 3-1:
**Building blocks to add to
an entity**



Property building block

The property building block is a structural building block which builders can use to provide an essential framework to an entity. A property provides greater detail and descriptive attributes to an entity, but do not necessarily identify the entity instances in a unique way. For example, with an Employee entity, the Employee ID is often unique, and frequently an identity building block. First name, last name, birth date, and telephone extension are not identifying, and can be added as properties to the entity.

Property types Each property building block has several details, including a name, a display name, and a type. The name should be unique for the entity: no entity may have two properties with the same name. The display name will be used in user interfaces and for interacting with the users. The type describes the type of data which is stored for the property. There are several acceptable data types:

- **Boolean.** This is a true/false type of value. The value is stored in the database as either true or false, but the values may be displayed to the end users as Yes/No, On/Off, Active/Inactive, or any such minimal pair. You may also offer a “no value” option, in which neither true nor false is offered. This allows you to make the field optional, or defer its value until a later date. You may choose one value as a default value.

Figure 3-2:
Boolean property type

Display Name	Value	Default
Active	true	<input checked="" type="radio"/>
Inactive	false	<input type="radio"/>
N/A	none	<input type="radio"/>

- **Currency.** The user may enter a numeric value in a currency type which is selected from a list of available international currencies.
- **Date.** Allow the user to store a date value for the property. The format of the date (e.g., day-month-year or month-day-year) is decided by the Process Platform settings.

- **Date and Time.** The user may store a date and time values. The format of the date and the time (e.g., 12-hour or 24-hour format) is decided by the Process Platform settings.
- **Decimal.** A numeric value which includes a decimal point. You may restrict the number of digits which appear before and after the decimal mark. The type of decimal mark is decided by the Process Platform settings. This data type may be commonly used for currency values.

Figure 3-3:
Decimal property type

Property Details

Display Name

Name

Property Type

Max number of digits left of the decimal mark

Max number of digits right of the decimal mark

- **Duration.** An arbitrary date value which you can use to specify relative dates and times. For example, when expressing a deadline or timer.
- **Float.** A numeric value which may include a decimal point. Unlike the Decimal type, you cannot restrict the digits before or after the decimal mark. The type of decimal mark is decided by the Process Platform settings.



A float typically represents a 64-bit floating point numerical value with 16 decimal digit precision. Floats normally have a range between -10,308 and 10,308. It is not recommended to use a Float to capture currency values, because they have a tendency to round values.

- **Image.** Allow the user to store an image, in the form of a recognized image file.

- **Integer.** A numeric value which never includes a decimal mark. You can set the minimum, maximum, and default values of the integer. By default, the minimum value of the integer type is 0 and the maximum is 2,147,483,647 with a default value of 0. These default values prevent negative values from being entered.
- **Enumerated Integer.** A numeric value which never includes a decimal mark and may only be one value from a list. You must create the list of acceptable values to be displayed to the user. Each value may have a display name. One value should be marked as a default value. Furthermore, you may choose to sort the available options to end users according to the list of display names.

Figure 3-4:
Setting values for
enumerated integer
property type

Default Value	Display Name
<input checked="" type="radio"/> null	None
<input type="radio"/> 1	Canada
<input type="radio"/> 2	USA
<input type="radio"/> 3	Mexico & Caribbean
<input type="radio"/> 4	South America

- **Text.** Any alphanumeric text of a prescribed length. By default, the maximum number of allowable characters is 64, but you may choose to change this maximum length.
- **Long Text.** Any alphanumeric text which does not need to be limited by length. This is a useful data type for full text descriptions.



It is a recommended practice to use the Text type as frequently as possible and use Long Text only when the length of the entry should not be restricted. Each data type helps to define the columns of the database table where the instance values will be stored, and using Text is a more efficient means of data storage and retrieval.

- **Enumerated Text.** Alphanumeric values which are presented in a list to users. One of the values is a 'null' value, which means no value will be stored. You may also choose to set one of the values as the default value. Like the enumerated integer, you may also choose to sort the list for your end users. Each value in the list has a default length of 64 characters, but this setting is variable.

Figure 3-5:
Setting values for
enumerated text property
type

The screenshot shows a configuration interface for an 'AirportCode' property. The 'Property Type' is set to 'Enumerated Text'. The 'Value Length' is set to 3. A checked checkbox indicates 'Sort list by display name to end users.' Below this, there is a table with three columns: 'Default', 'Value', and 'Display Name'. The table contains four rows. The first row has a selected radio button in the 'Default' column, and the 'Value' column contains 'null' with a tooltip 'None'. The other three rows have empty 'Default' columns and contain 'CDG' (display name 'Paris-Charles de Gaulle'), 'JFK' (display name 'New York-John F Kennedy'), and 'ORD' (display name 'Chicago-O'Hare') respectively. There are also up and down arrow buttons to sort the list.

Default	Value	Display Name
<input checked="" type="radio"/>	null	None
<input type="radio"/>	CDG	Paris-Charles de Gaulle
<input type="radio"/>	JFK	New York-John F Kennedy
<input type="radio"/>	ORD	Chicago-O'Hare

Primary and secondary attributes Many properties, when they are added to an entity, have mandatory attributes which must be entered when the property is added. These are first-step, or 'primary', attributes. The most common primary attributes are the name and display name for the property. For example, when adding a property which uses the text data type, you must include the name, display name, and length (measured in number of characters) of the text field.

Once a property has been added to an entity, however, you may add additional or 'secondary' attributes. These attributes are optional and only become visible once the property has been added. With the text field, for example, once the property has been saved with a name, display name, and length, you can include several optional secondary attributes, such as the default value, a tooltip to display when the user hovers over the property in a form, and a default pattern for which to display the text (for example, as a telephone number).

Static and dynamic enumerations There are two types of enumerations offered as a data type: integers and text. With an enumerated integer, a user chooses an option from a predetermined list which is stored as a number in the database. With enumerated text, the user chooses an option from a predetermined list which is stored as text in the database. Enumerations are useful for storing simple values for complex, arbitrary lists (e.g., select employees by name, store by employee number, or select by name of state and store by two-character state code).

Such enumerations can be static or dynamic. A static enumeration is one in which the values are established when the property is added, and are rarely changed, if ever. The list of states or provinces, for example, is a common static enumeration.

Dynamic enumerations, are ones in which the values may fluctuate and change frequently. These values could potentially change from one instance to the next. In this case, you could retrieve the values from some externally located source. When you establish a dynamic enumeration, you do not need to build the enumeration at design time. Instead, you choose a business process which will fetch the values for you. The values depend upon the context, so you can pass the values from the entity to the retrieval process.



You will not build business processes in this course. If you want to learn how to build robust business processes which can use web services to retrieve values, you can register for 4-4913: Process Modeling for Process Platform.

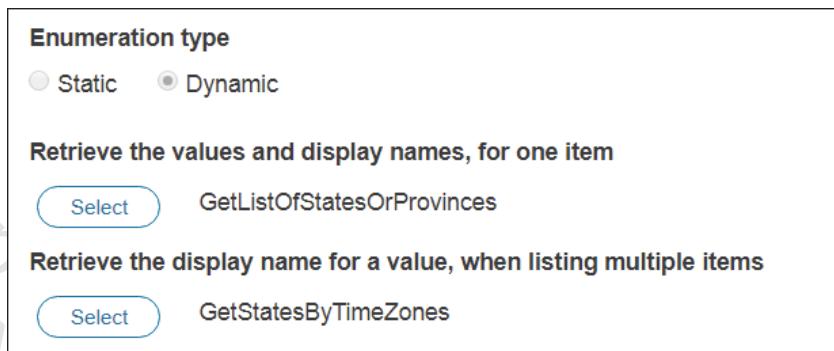
There are two attributes which you must specify when choosing a dynamic enumeration:

- Retrieve the values and display names for one item.

This named business process will retrieve a table of values and correlated display names based upon the context of the entity and present them as a list to the user. The user makes the selection based upon the display name, but the value is stored.

- Retrieve the display name for a value, when listing multiple items.
This named business process will accept a list of values as input and produce a list of display names which match the values provided. This is a useful option for filtering from a large list.

Figure 3-6:
Business processes for dynamic enumerations

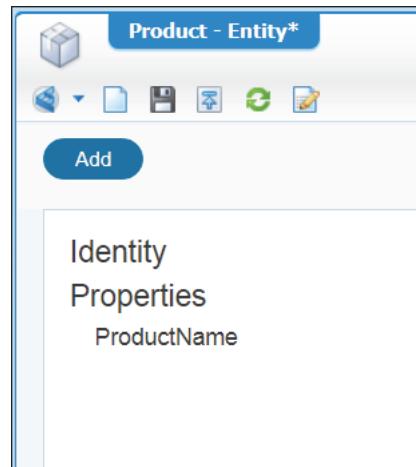


Add property building blocks to an entity

1. Log in to Process Platform Warehouse organization as user **analyst@training.local** and open the Northwind Workspace from Workspace Documents if you have not already.
2. Expand the Warehouse Application project and Entities folder.
3. Double-click the Product entity to open it in the editor.
4. Click Add.
5. Select Property from the building blocks on the left.
6. Set the Display Name to **Name**.
7. Set the Name to **ProductName**.
8. Set the Property Type to **Text**.
9. Set the Length to **64**.

10. Click **Add**.

**Product entity with
ProductName property**



11. Add another Property (Add > Property).
12. Set the Display Name to **Category**.
13. Set the Name to **CategoryID**.
14. Set the Property Type to **Enumerated Integer**.
15. Select the **Sort list by display name** check box.
16. The initial value in the enumerated integer list is a null value. Set the Display Name of the null value to **N/A**.



When using static enumerated integer and enumerated text types, one value must always be the null value ('none'). This value cannot be changed or removed, but you may set the display name to something relevant for your users.

17. Click **+** to add a new value.
18. Set the display name to **Beverages** and the value to 1.
19. Include the following additional values and Display Names:

Display Name	Value
Condiments	2
Confections	3
Dairy Products	4
Grains/Cereals	5
Meat/Poultry	6
Produce	7
Seafood	8



The value which will be stored will be an integer, but the end user will choose the value based upon the Display Name.

20. Click *Add and continue*.

The Add and continue option will allow you to add many properties at the same time.

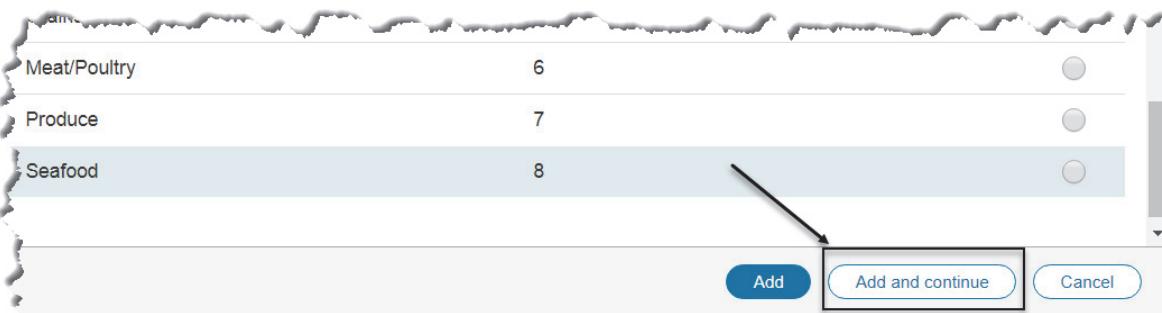


Figure 3-8: Add and continue

21. Add the following Property building blocks:

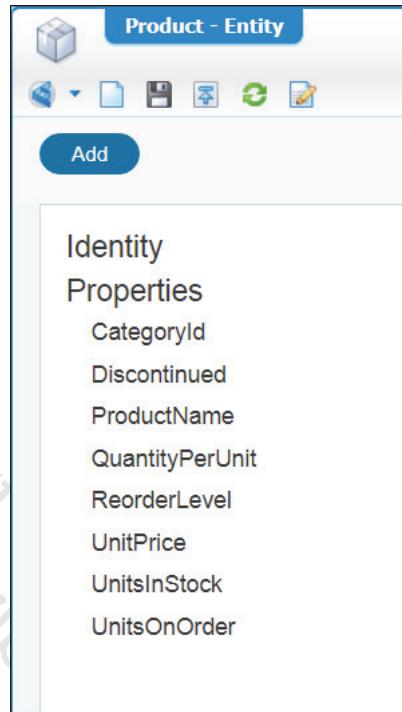
Display Name	Name	Property Type	Length	Currency
Shipping Units	QuantityPerUnit	Text	64	
Price per Unit	UnitPrice	Currency		CAD
In Stock	UnitsInStock	Integer		
On Order	UnitsOnOrder	Integer		
Reorder Level	ReorderLevel	Integer		
Discontinued	Discontinued	Boolean		

22. After creating the Property of Boolean type, set the Display Name of the true value to **Yes**, the false value to **No**, and the “none” value to **N/A**.
23. Set the false value of the Boolean property type to be **Default**.
24. Click *Add* to add the Boolean property.
25. Save and close the Product entity.
26. Validate the entity in the workspace.

Changing property building blocks

Once you add property building blocks to an entity, they are all captured on the entity editor and categorized in the properties category by their name (not their display name).

Figure 3-9:
List of properties on entity editor

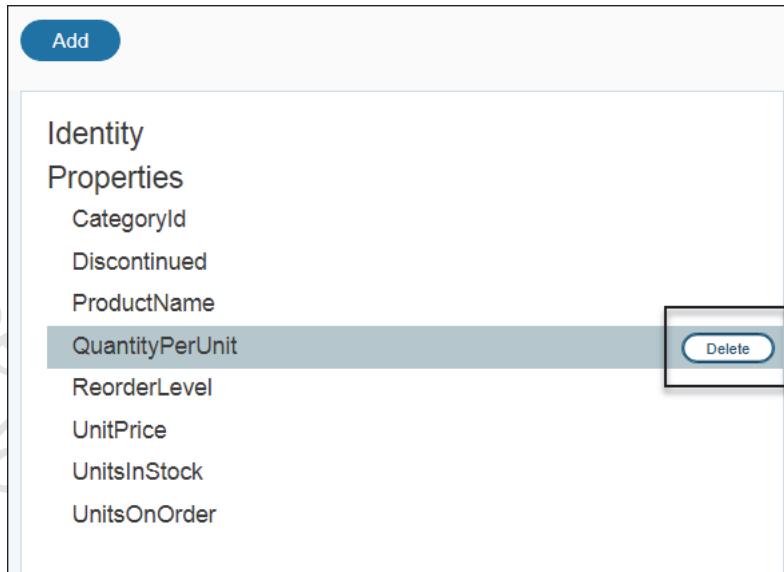


Delete a property You may edit or delete the property building blocks from the entity at any time. To delete a property building block from an entity, open the entity in the editor and hover your mouse to the right of the property name in the list.

A delete button will appear. Click the delete button to remove the property from the entity.

Figure 3-10:

Delete a property



Be careful when deleting a property building block. Once you confirm the delete, the property is removed. If you want to undo the deletion, close the entity without saving your changes. If the entity has been published and deployed before you delete the property, you should re-publish and re-deploy the updated solution as soon as possible.

Editing a property There are some attributes of a property which may be set, and others which may not be changed. For example, once you add a property building block to an entity, you cannot change the name or the property data type. You may, however, change the Display Name of the property or any field of the property (i.e., display names of boolean values or enumerated text values).



In order to change the name or data type of a property, you must first delete the property from the entity, then add it as a new property with the new values.

Each property has at least four additional secondary attributes which are added once the property is created:

- Allow participants to change the value. This field determines if property value is read-only or not. Options are ‘Anytime’, ‘Never’, and ‘Only when first created’.
- When users work on this property at the same time, report conflicts. It is possible that two users will be working on the same instance of an entity at the same time. If this occurs, it has the potential to create a data conflict (i.e., User A enters a different value than User B in the same property). If this occurs, you can report the conflict:
 - When a change is made by the other user
 - When changed to the same or a different value
 - When changed to a different value only
 - Never

Conflicts are only reported if the database is updated with new values while a form is displayed elsewhere, and before the changes are saved.

- Available for use by others. This field is a simple check box which, by default, is not selected. Every entity you create, and every Property of that entity, may only be used by documents in the current project. However, you may create entities which can be used in different projects. For example, you may create an Employee entity in the Warehouse Application which can be re-used in a Human Resources Application. The Human Resources Application may be in a completely different project or workspace. However, if you select the ‘Available for use by others’ field, the entity can be shared with other projects. Selecting individual Property building blocks within the entity shares just those Properties with other projects, provided that the entity is also shared.
- Tooltip. This is arbitrary text which will appear when the user hovers over the field in a user interface, such as a form.

Furthermore, there are other secondary attributes which are set for certain data types. For example, for properties with the currency, decimal, or float data type, you can set the minimum, maximum, and default values.

Intrinsic validation	When you create a property for an entity, Process Platform automatically performs a basic set of validations for the different property types. This validation is required to ensure that data entered by the user follows the proper format (e.g., numeric property types do not support text). This form of validation, called <i>intrinsic validation</i> , is performed automatically in Process Platform, is locale-specific (depending upon the user’s globalization settings), is not optional, and cannot be configured.
-----------------------------	--

There are other types of validation which may be performed on values in a property. For example, you may want to enable validation so that the number of products order cannot exceed the number of products in stock. This type of validation is business logic validation and is performed with a rule building block.

Relationship building blocks

As you add property building blocks to your entities, you may begin to notice that some properties begin to “overlap” between entities. For example, earlier in this chapter, you created a property with the enumerated integer type in order to capture a CategoryID. Category could represent a table in another database, with CategoryID functioning as its primary key. In the previous chapter, you created an entity for Order, which represents a customer’s order. Each order may be composed of several products (i.e., a quantity of products may be sold in an order). These represent the line items in an order. Each order line has a reference to the original order and to the product ordered. These cross-references can be characterized with relationship building blocks.

Relationship types and directions

There are two basic relationship types for an entity: peer and child.

Peer relationship In a peer relationship, two separate and independent entities form a reference to one another to satisfy a business purpose. The two entities will have instances of their own which may act on their own, and the reference between them is not immediately necessary. For example, in the scenario presented here, customers will place orders, but you do not need to create an order every time you create a new customer. Similarly, you do not need to create an order every time you create a new product.

Parent-child relationship

In a parent-child relationship, one instance of an entity is the parent and another instance is a child. The child instance is dependent upon the parent, and cannot exist without a parent. When placing an order, for example, there are several line items to the order. You characterized these line items with the OrderLines entity. You cannot have OrderLines without having an Order, so these two entities are in a parent-child relationship: Order is the parent and OrderLines is the child.



In the OrderLines entity, you will also have a reference to a product. This is a peer relationship to the Product entity.

Reflexivity

Reflexive relationships are a special type of peer relationship. A peer relationship presupposes that there are two independent entities. A reflexive relationship contains only one entity which refers back upon itself. A typical example of a reflexive relationship is between a manager and employee. Both of them are instances of the Employee entity, but an employee’s manager is a reference back upon another, different instance of an Employee entity.

Relationship direction Relationships can also express direction: either unidirectional (one-way) or bidirectional (two-way). In a one-way relationship, only one of the entities enforces the reference, or “knows” about the other entity. For example, the relationship between a Product and an OrderLine could be modeled as a one-way relationship: when adding items in a shopping cart, each line item (OrderLine) must refer to one Product. However, when adding a new Product, you never refer to the OrderLines in which it is or will be included.



You may be interested in reporting all the orders and order lines in which a product is sold, but this is for reporting purposes, and is handled in Process Intelligence.

Conversely, in a bidirectional, or two-way, relationship, both entities are aware of each other. The employee/manager reflexive relationship is a good example of a bidirectional relationship: when you add an employee, you can specify the manager, or when you add a manager, you can add a set of employees. In our example, you must enforce a Customer for every Order, and your business may choose to monitor all Orders for each Customer.

In Process Platform, relationships are built one at a time with Relationship building blocks. Each relationship is assumed to be a one-way relationship until it is “paired” with a matching relationship in its target entity. Once the two relationships are made aware of one another, or “paired”, bidirectionality is enforced. Later in this chapter, you will pair relationships in order to enforce bidirectionality.

Multiplicity

Multiplicity represents the number of instances existing on each side of a relationship. For example, a Customer may place many Orders, but each Order has just one Customer. The Process Platform engine enforces all relationship multiplicity settings. Relationship multiplicity can be expressed either:

- one-to-one (“To One”): the relationship supports one and only one related instance. For example, each item (i.e., OrderLine) in a shopping cart is related to one and only one product.
- one-to-many (“To Many”): the relationship support many related instances. For example, a customer may place many orders.



It is a good strategy when designing relationships to create the relationships in ascending multiplicity (“To One” to “To Many”).

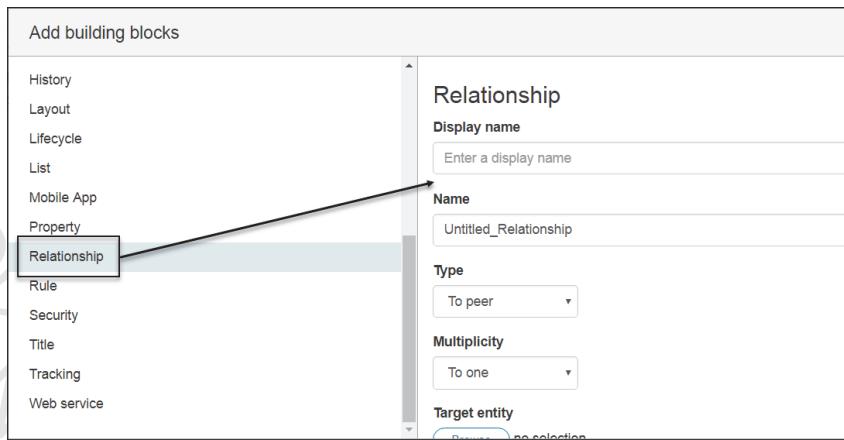


Parent-child types of relationships have an implicit multiplicity of “To Many” (i.e., a parent has many children). You cannot specify a parent-child relationship type as “To One”.

Adding relationships in Process Platform

Relationships are modeled using the Relationship building block in Process Platform. Relationship building blocks are available from the Add menu in the entity editor.

Figure 3-11:
Relationship details in Add dialog



Relationship names Every relationship building block must have a unique name, just like each property building block had a name. The name can represent the reference between source and target entities, or can express the nature of the relationship. For example, you could name the manager/employee relationship: “direct reports” in order to express the one-to-many relationship a manager would have with employees, or “reports to” to represent the one-to-one relationship an employee has with a manager.

As is the case with other building block attributes, you can set a display name for the relationship. In parent-child relationships, you can set display names for both the parent and the child.

Type In the Type field, you will notice that there is only the “To peer” and “To child” types. These describe the peer and parent-child types of relationships. To indicate a reflexive relationship, use the “To peer” type and indicate the same entity as the target.

When you indicate a “To peer” type of relationship, you can set the Multiplicity field and you must choose the Target entity to which the source refers. Click the Browse button and choose the target entity from the list of available entities in your project.

Figure 3-12:
Target entity for “To Peer” type



Process Platform offers many pre-built entities, such as phone number, language, email address, and many others, in order to make construction easy.

In the Target entity selection dialog, you can also click the New button to create an ad-hoc entity while designing your relationship.

The other Type you can choose is the “To child” type. When you select “To child” as your relationship type, the multiplicity is automatically set to many.

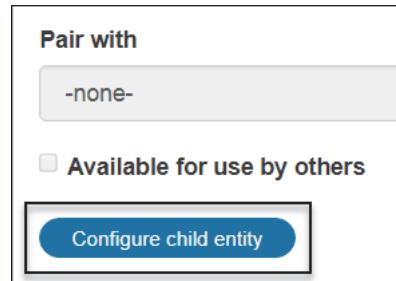


The child entity is always created ad hoc - you cannot create two entities separately and then connect them later in a parent/child relationship.

The name of the child entity is always set to the name of the relationship.

After you create the parent-child relationship, an entity placeholder will be set aside for the child entity. In the entity editor (of the parent), you will notice a button enabled for you to configure the child entity. This button will trigger another entity editor window for you to work with the child entity.

Figure 3-13:
Configure child entity
button





Child entities cannot be viewed, edited, or accessed from the Workspace Documents application. They are only accessible through the parent entity document.

Pair with Entity relationships will sometimes come in pairs: “to one” in the source entity and “to many” in the target entity, for example. In the relationship building block tool, the Type and Multiplicity fields help define a single direction of that relationship. You can build the other direction of the relationship in the target entity. Once both sides of the relationship have been established, and the Target entity has been named in the Relationship building block editor, use the Pair With field to twin the two relationships together to form a bidirectional (two-way) relationship.



The many-to-many relationship is the most rare and difficult type of relationship to model. In a many-to-many relationship, many instances of one object are related to many instances of a second object. For example, a book may have many authors and an author may write many books. This must be carefully modeled with entity modeling.



Create a parent-child relationship in the Order entity

1. From the Workspace Documents application in the Warehouse organization of Process Platform, open the Northwind Workspace if it is not already open (you must be logged in as user **analyst@training.local**).
2. Expand the Warehouse Application and the Entities folder.
3. Double-click the Order entity to open it in the editor.
4. Click Add.
5. Choose the Relationship building block.
6. Set the Display Name to **Order Line** and Name to **childOrderLine**.
7. Set the Type to **To child**.

The Multiplicity is automatically set to many.

8. Set the Child display name to **Line Item**.
9. Click **Add**.

The Relationship is added under the list of Relationship building blocks in the editor.

10. Save the Order entity.
11. Click **Configure child entity**.

The Order Line entity opens in a separate entity editor window.

12. Save the *childOrderLine* entity and close it.
13. Save the *Order* entity.

Despite OrderLine being added as a child entity to Order, the child entity does not appear in the collaborative workspace. Be mindful that the CWS only displays parent entities: it does not display child entities.



As well, be aware that two entities in a parent-child relationship are not the same thing as a super-/subtype entity hierarchy. In a super-/subtype hierarchy, the subtype is logically similar to the supertype and inherits the supertype's properties. Entities in a parent-child relationship may be related, but do not necessarily need to be logically similar. Likewise, they do not inherit properties.

Title building block

As you develop more entities and user interfaces in which to edit them (i.e., forms and layouts), it will be important to distinguish one entity from another. The title building block adds a title to an entity so that its items (i.e., entity instances) may be easily distinguished from other items. Some other building blocks, such as history, will even use the title as a simple, effective, user-accessible means to identify an item.

The only purpose of the title building block is to distinguish entity instances from one another. An entity may have only one title building block.

Create by The only relevant field of the title is the name: the distinguishing feature, written as text. Another field, Create By, permits you to choose whether the title is created by the user, or by the system. By allowing the system to create the title, you reduce the amount of work by automatically generating a title for the item.



Changing the Create By option changes the behavior and properties of an entity. You should not change the option if you have deployed and created instances from the entity.

Title specification If you are using the Create By system option for a title building block, you can have the system create a specific title. Add this as text in the Title Specification field. The Title Specification may also create system-generated dynamic messages: message based upon variable fields, such as a person's name or an order's unique identifier. These messages must be added to the Title Specification using the entity scripting language.



You will be using the scripting language in a later chapter.



Add a title to the Order entity

1. Open the Order entity from the Warehouse Application if it is not already open.
2. Click Add.
3. Select Title.
4. Leave the default values selected.
5. Click Add.
6. Select the Title building block from the entity editor.
7. Change the Create By option to **System**.
8. Set the Title Specification to **Order**.
9. Save and close the Order entity.
10. Validate the Warehouse Application.

Summary

Having completed this chapter, you should be able to:

- Describe the properties of an entity
- Add properties to an entity in Process Platform
- Define an entity's relationships
- Add a relationship to an entity
- Add a title to an entity

Exercises

1. Add title building blocks to the following entities:

Entity	Create By	Title Specification
Business	System	Business
Person	System	Person
Product	System	Product

2. Add property building blocks to the following entities according to the respective tables. Leave the default values unless otherwise noted:

Customer:

Display Name	Name	Property Type	Other
Client Code	ClientCode	Text	Length = 5 Change value = when first created
Address	Address	Text	
City	City	Text	
Region	Region	Text	
Postal Code	PostalCode	Text	Length = 12
Country	Country	Text	Length = 24
Phone	Phone	Text	Length = 24
Fax	Fax	Text	Length = 24

Business:

Display Name	Name	Property Type	Other
Company	CompanyName	Text	
Contact	ContactName	Text	
Logo	CompanyLogo	Image	

Person:

Display Name	Name	Property Type	Other
First name	FirstName	Text	
Last name	LastName	Text	
Home phone	HomePhone	Text	Length = 12

PerishableProduct:

Display Name	Name	Property Type	Other
Keep Refrigerated	Refrigerated	Boolean	True = Yes False = No none = N/A
Temperature	Temperature	Decimal	Digits left = 3 Digits right = 1 Default = 72
Best Before	BestBefore	Date	

Order:

Display Name	Name	Property Type	Other
Employee	EmployeeID	Enumerated Integer	Values: N/A - none Davolio, N - 1 Fuller, A - 2 Leverling, J - 3 Peacock, M - 4 Buchanan, S - 5 Suyama, M - 6 King, R - 7 Callahan, L - 8 Dodsworth, A - 9 Sort list by display name
Order Date	OrderDate	Date and time	Property value can be set only when first created
Requested Date	RequestedDate	Date	
Ship Via	ShipVia	Enumerated Integer	Values: N/A - none Speedy Express - 1 United Package - 2 Federal Shipping -3 Sort list by display name
Freight	FreightCost	Currency	Currency = CAD
Shipped Date	ShippedDate	Date and time	

3. Open the Order entity. Select the childOrderLine relationship and click the Configure child entity button. Add Property building blocks to the child entity, OrderLine, according to the following table:

Order Line:

Display Name	Name	Property Type	Other
Price	SalePrice	Currency	Currency = CAD
Quantity	Quantity	Integer	Default and min value 1
Discount	Discount	Decimal	1 place left and 2 places right of decimal mark Min value 0, max value 1, default 0

4. Save and validate your changes to the entities.
5. Open the OrderLine entity (Order > childOrderLine relationship > Configure child entity). Create a one-way peer relationship from the OrderLine entity to the Product entity. Use the following values:

Display Name	Product Ordered
Name	oneLineItemToProduct
Type	To peer
Multiplicity	To one
Target Entity	Product
Pair with	(none)

6. Create a two-way peer relationship from the Order entity to the Customer entity and the inverse. Use the following values:

	Order	Customer
Display Name	Bill To	Transaction
Name	OrderToCustomer	CustomerToOrder
Type	To peer	To peer
Multiplicity	To one	To one
Target Entity	Customer	Order

When setting the Pair with field, pair the latter (CustomerToOrder). Once you establish the relationship and pairing in the second entity, the pairing in the former entity will be automatically established.

7. Save, close, and validate the Warehouse Application when you are complete.

8. In Workspace Document, click the button Make Changes Available to Others in order to commit your changes to the repository. Commit your changes with the description “Chapter 3 - Properties, Relationships, and Titles”.

Open Text Internal Use Only
Do Not Distribute

4. Lists in Process Experience

Objectives

On completion of this chapter, participants should be able to:

- Navigate Process Experience
- Configure a list from an entity for Process Experience
- Add an action bar to an entity
- Configure a list with an action bar
- Publish and administer your solution in Process Experience

Overview

Process Experience is a configurable user interface in which users can view, track, manage, resolve, and close cases. Additionally, managers can administer and organize work into lists. Process Experience brings together processes, application instances, and content from other participants in Process Suite, such as Process Platform. Process Experience offers a single user interface for interacting with entity instance data.

In Process Experience, providers manage the case details which were submitted by their customers. With Process Platform, your organization can create customized user interfaces to access data and organize it in meaningful ways for Process Experience. Builders can also create home pages, data forms, user interface layouts, and lists that enable end users to work with the data in an organized fashion. Discussions allow users of Process Experience to communicate in order to enhance this collaboration.

After building entities with all of their related building blocks, you must publish your entity-based solution so that end users may access it in Process Experience. In the final section of this chapter, you will publish your solution and test it in Process Experience.

What you will build in this chapter

- A Process Experience theme. The theme is a set of colors, fonts, backgrounds, and images which are used to display for the end user interface: Process Experience.
- List building block: you will add the List building block to your entities. The list will provide you with the opportunity to view instances of your entities in lists of results. Lists are configurable and can filter instances based upon established criteria.
- Action bar building block: a configurable set of buttons for each entity which end users will be able to click in the runtime.
- You will also publish your solution to Process Experience. This will allow you to view your solution as your end users will see it.

Timing

Lecture: 60-75 minutes

Exercises: 45-60 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform 16.3 Entity Modeling Guide

About Process Experience

Process Experience is a browser-based web site secured with a unique user ID and password. The user and password information is maintained by OpenText Directory Services (OTDS). Once you are logged in, the default view is the Process Experience home page layout. The Process Experience home page layout contains panels which you will use to work with, configure, and create instance data for your run time entities. The functions which are available on your home page layout depend upon your user and role access.



Users and the roles to which they belong are managed in identity packages and synchronized with OpenText Directory Services. User profiles are not managed in Process Experience.

OTDS provides seamless connectivity between Process Suite participant applications, such as Content Server, and a central directory service. Some organizations prefer to use single sign-on (SSO). SSO is also managed by OTDS. When SSO is enabled and a user logs in to Process Platform, they are also automatically and transparently logged in to other applications, such as Process Experience or Content Server. This authentication facility is supported for web browser access (under the control of the HTTP or web application server), and for applications that access through the OpenText Application Programming Interface.



You will learn how to work with users and groups in identity packages later in this course.

To access Process Experience, your administrator must provide you with:

- A URL for accessing your system. To make it easier for future access, create a bookmark or make the URL a favorite in your web browser.
- Your unique user name and password combination. Most user names are not case-sensitive as part of the system security, but passwords are. Check with your administrator if you're uncertain.

Process Experience uses your login information to:

- Determine the information which you can access in Process Experience,
- Determine the actions which you're permitted to perform on that information (i.e., your permissions),
- Determine the types of information you can add (your privileges),
- Provide you with access to your Process Experience home page, and
- Track information for auditing purposes.



Figures and examples illustrate the default English localized installation of Process Experience, however, the user interface can be displayed in other locales.



Log in to Process Experience

1. Open a new browser window (Google Chrome or Internet Explorer).
2. From the Favorites bar, click Process Suite 16.3 > Warehouse Org > Process Experience.

The full URL is: <http://localhost:81/home/warehouse/app/processExperience/web/perform>



The URL breaks down to:

`http://<server>:<port>/home/<org>/app/processExperience/web/perform`

If this is the first time you've opened Process Experience, you may need to wait several seconds while the screen renders.

3. If you are logged in to Process Platform as the user **analyst@training.local**, you will be automatically logged in to Process Experience as the same user. If not, log in as user **analyst@training.local** with password **opentext**.



By virtue of the current configuration, OTDS permits single sign-on (SSO), which will automatically log you into any other Process Suite application (Process Platform, Process Experience, etc.).

Navigate the Process Experience desktop

All of the components with which you work in Process Experience are managed and accessed through Process Experience pages. These pages can also be customized. A user may have several page layouts, but only one default, “home” page layout. *Panels* are the primary components of a page. Each panel provides certain functionality, such as work items awaiting your attention. The Process Experience home page provides a consistent user interface for accessing data, regardless of where it resides on an external system.

Desktop composition

The Process Experience home page is divided into two distinct sections: the header area and the panel area.

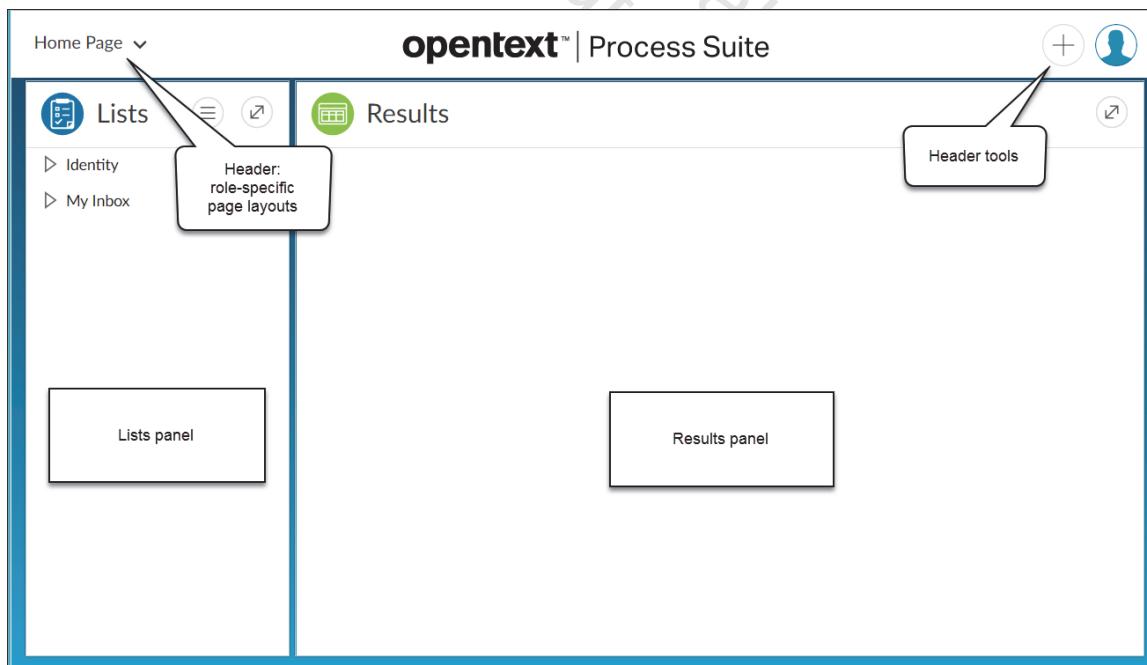
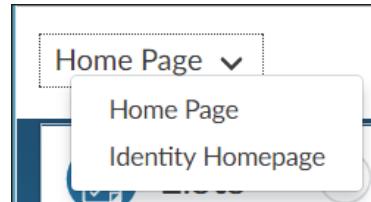


Figure 4-1: Process Experience header and panel areas

In the header area, you will find:

- A drop-down list which displays the pages for objects to which you have access. The first time you open Process Experience, a default layout, called Home Page, is displayed. The other default layout is Identity Homepage, which allows you to work with identity packages and OTDS. If you select a different layout, it will be remembered as your new default home page.

Figure 4-2:
Page layouts available
under the Home Page
button



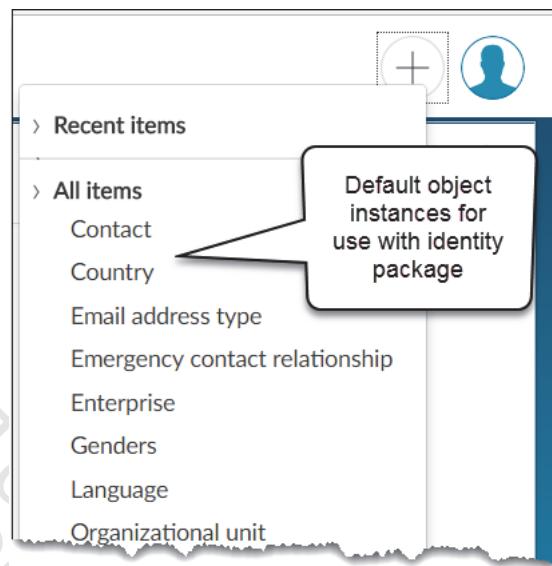
Later in this course, you will create page layouts and work with the identity package.

- A *Create* button (+) to create new object instances. These could be instances of entities, processes, cases, or more. In this course, you will create instances of entities.

This list is divided in to two halves: the bottom half is a list of all the instances which you are permitted to create. By default, this also includes object instances for use with the identity package. The upper half is an automatically generated list of the items which you have recently selected. Only the 10 most recent items are stored in this list.

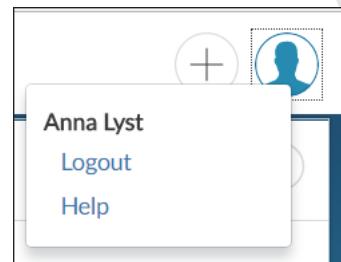
This recent items are arranged in reverse chronology descending from most recent item selected.

Figure 4-3:
Create new list



- Your user avatar which includes:
 - Your user profile picture
 - Your user name
 - A link from which you can log out of Process Experience. You will consequently be logged out of every other system to which OTDS is connected.
 - A Help link from which you can access the help system.

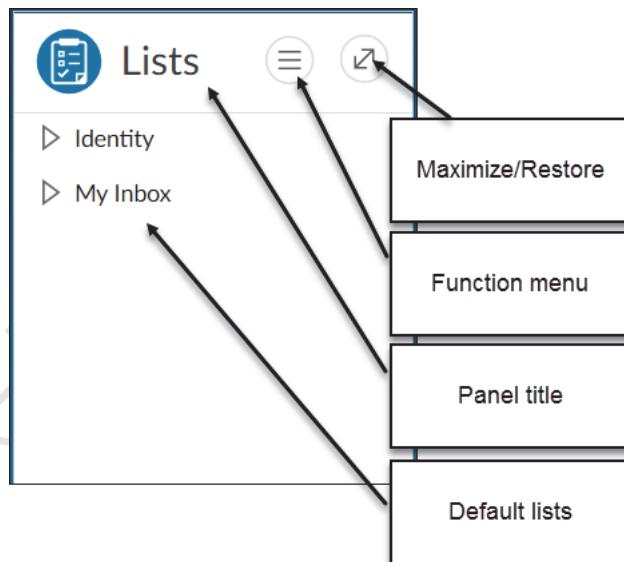
Figure 4-4:
User avatar



The panel area of the Process Experience page contains any number of Process Experience panels. Each panel has the same basic design:

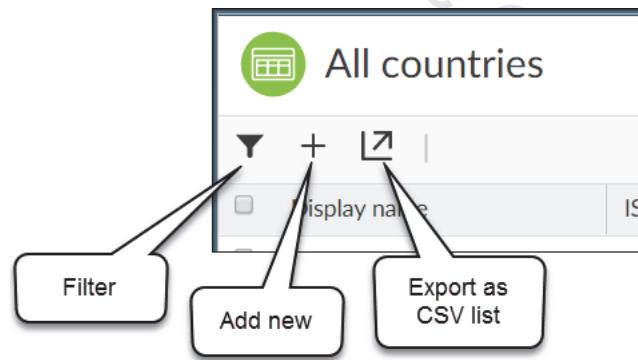
- Panel name
- Function menu to work with the panel
- Maximize/Restore button

Figure 4-5:
Anatomy of a Process Experience panel



In addition, some panels may have additional functionality, such as a filter, a search bar, a create button, or a button to export results as a list of comma-separated values.

Figure 4-6:
Other panel buttons



Filtering behaves differently depending upon the panel and the type of filter available. A results panel which has been configured for filtering consists of a hidden sub-panel which contains the set of columns available for the panel. The sub-panel is shown or hidden using the Show/Hide Filter option in the results panel.

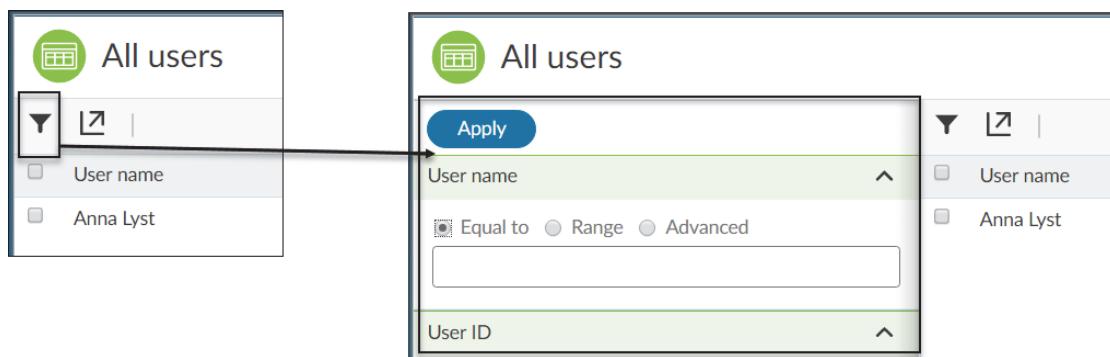


Figure 4-7: Show or hide filter sub-panel

This type of filter is most often found on results panels. A text field allows the user to limit the list items in the panel to those which contain the user text. This type of filter is often found when there are long lists of items to select.

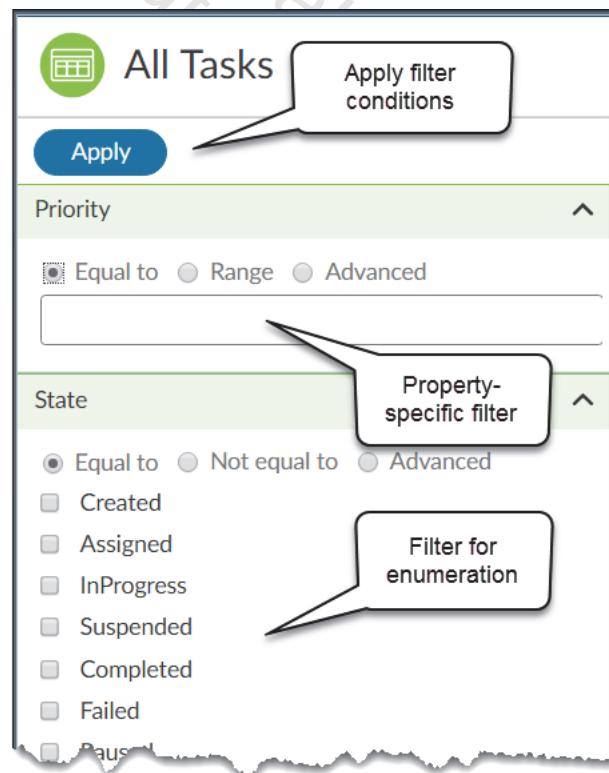


Figure 4-8:

Examples of different types of filter functions

The filter function which is available for the results panel is sensitive to the different data types of the entities. For example, if the entity's property is set to contain text data, then the filter function contains string-type filters. If the entity property contains date or time data, then filter function contains date-time-type filters. If the entity property contains date data, a calendar tool is provided which allows the user to easily choose a specific date. In the example above, the State property was an enumerated text type, therefore, the filter presents a list of check boxes from which to select.

The function menu in a panel contains different selections, depending upon the type of panel or where it is used. For example, panels which feature search or filter results often provide menu options for selecting display columns.

Working with panels

There may be any number of panels on a page. By default, there are usually two panels on the Process Experience home page. When one of these panels is maximized using the Maximize/Restore button, all the other panels are minimized to a list of buttons in the footer area of the page.

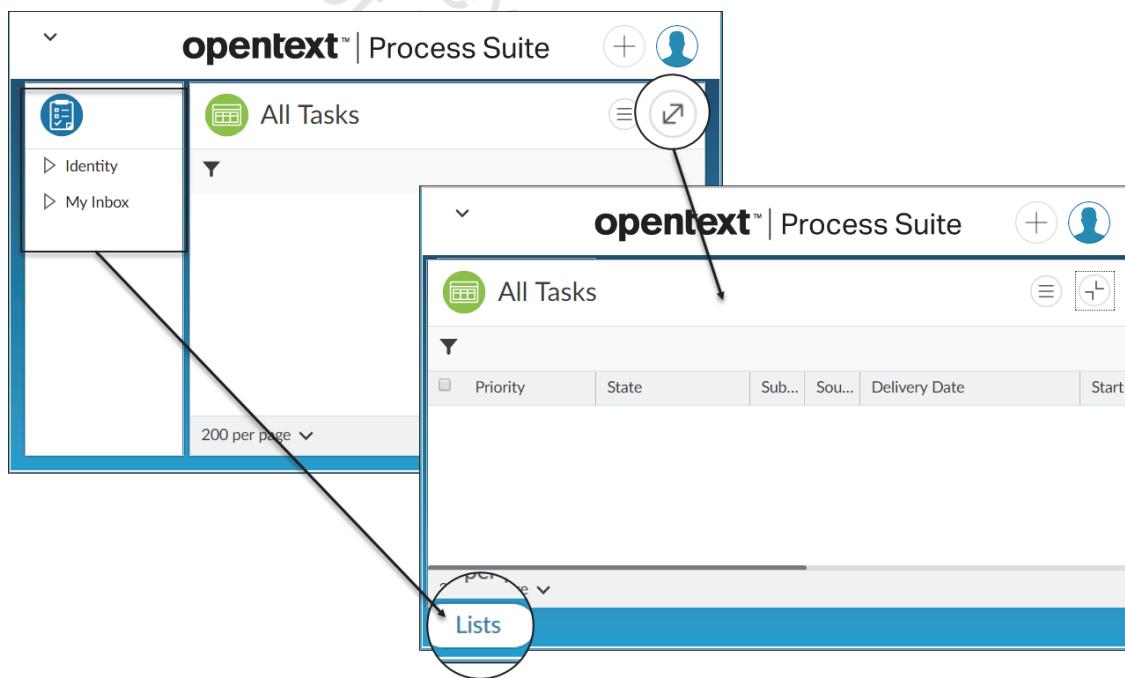


Figure 4-9: Lists panel minimized to footer when Results panel maximized

Maximize/Restore To restore the layout, click the Maximize/Restore button again. Alternatively, when one panel is maximized, you can click the button in the footer associated with the minimized panel to swap the maximized panels.



Not all panels can be maximized: the panel must have a Maximize/Restore button, and there must be more than one panel on the page. If there is only one panel on a page, it is always maximized and there is no Maximize/Restore button available.

Resizing a panel Between each panel is a border which you can drag to change the horizontal or vertical size of your panels on the desktop.



Panels are resized in increments of 5% of the screen height or width. The minimum height and width for any panel is 5% of the total screen size.

Personalized size preferences are retained even if you refresh the page. If a manager or administrator changes the page layout, these personalization settings are reset to the default values specified by the page builder.

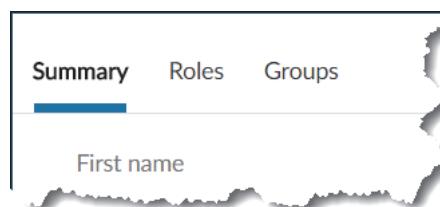


Panel names are truncated when a panel is resized. You can hover over the truncated name to see the full panel name displayed as a tooltip.

Tabbed panels Depending upon how the page layout was configured, panels can be shown as tabs. For tabbed panels, the list shows the name of the active tabbed panel only. You can select a different panel by clicking the corresponding tab. Although long tab names may be truncated, you can hover over the name to see the full name as a tooltip. If tabs cannot be adequately fit in the window, they are displayed as a drop-down list instead. Tabbed panels can be maximized and restored the same way in which other panels can.

Figure 4-10:

Tabbed panel



Sub-panels Some panels may have sub-panels, which can be used to preview content. If a panel has sub-panels, the Maximize/Restore button is available for each sub-panel. When one sub-panel is maximized, the other sub-panels are hidden.

If you maximize a sub-panel, the list of available panels is displayed within the parent. For instance, if you maximize a sub-panel within a Preview panel, the list of hidden sub-panels is displayed at the bottom of the Preview panel. The list contains only the sub-panels that can be maximized within the respective parent.



To simplify content on mobile devices, the display of layouts is optimized for windows whose width is less than 968 pixels. The layout display responds automatically to the mobile device you're using. For example, if you sometimes work on a smartphone where the display is less than 768 pixels, the layout is optimized in the following ways:

- Columns are displayed as rows from left to right and top to bottom.
- Each panel occupies as much height as needed with vertical scrolling for pages.
- Panel headers are displayed and panels defined as tabs are shown in a drop-down list in the panel's header.

Types of panels

All panels fall into basic types. Each type of panel has its own functions or buttons which are available:

Action panel An Action panel contains action buttons which are relevant for the item you are working on. These buttons may be Edit, Delete, and Modify. Later in this course, you will learn how to add buttons dynamically to this panel with the rule building block.

Create panel A Create panel is a common panel used to create an instance of an entity. These panels use Forms from the entity to display relevant fields for creating the instance. A Form is the default way of displaying the instance of an entity. You will create forms later in this course.

Results panel A Results panel displays the results of a search, filter, or list. For some Results panels, you may also preview the instance.

Preview panel A Preview panel displays a preview of the instance or instances which are commonly selected from the Results panel.

List panel A list panel arranges your lists. Lists organize entity instances into groupings logical to the business. Only those lists which you have authority to access are displayed.

External panel This is a special panel which displays data from an external system.

Web panel This is a simple panel which displays content for a web site. A designer can configure the capabilities of a web panel.

Process Experience themes Businesses have the ability to change the look of Process Experience to suit their own needs. An easy way to accomplish this is to change the Process Experience theme. The *theme* is a collection of properties, such as colors, gradients, and a logo, which are used to create a specific look for Process Experience pages. Themes are stored as CSS files (Cascading Style Sheets), a standard for describing web pages. Process Platform offers a simple, intuitive editor for creating and applying themes to the Process Experience page.

In the design time for Process Platform, a theme is a document, which can be created and added to a project in a workspace. To create a theme for Process Experience, it's a simple matter of creating and configuring a new Theme document.

In the document, you can add a corporate logo to the header, change the background colors, change the font style and size, and/or change the look of the buttons on the page.



Create a theme for the Northwind application

1. *Return to Process Platform and the Northwind Workspace.*
2. *Right-click the Warehouse Application project and select New > Folder.*
3. *Set the name of the folder to Themes.*
4. *Right-click the Themes folder and select New > Other.*
5. *In the New Document dialog, select Theme.*

You can use the search filter to narrow down the documents by name.

6. *In the Header section, click Browse image.*
7. *Navigate to C:\Support Files and select Northwind Banner.gif.*
8. *Click Open.*



A header image should be 55 x 385 pixels.

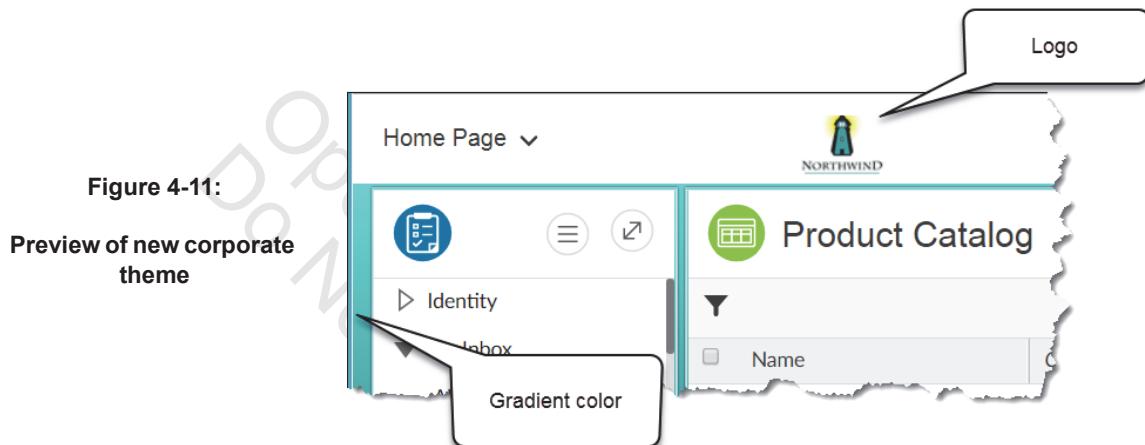
-
9. *For the Background, select Gradient.*
 10. *Set the Top color to be #65dfdf and the Bottom color to be #429c9c.*
 11. *Set the Font name to Helvetica.*
 12. *Set the Panels Border color to #c9f9f9.*
 13. *Set the Text color for the Global action bar to #429c9c and the Background color to #f7f7aa.*

14. Save the theme with the name **Northwind Theme**.

Make sure the location is *Warehouse Application/Themes*.

15. Click *Preview* in the theme editor header.

You should be presented with the Process Experience home page and its new theme and logo.



The theme will not be used until it is published to Process Experience (later this chapter).

16. Close the preview window.

17. Close the theme editor.

List building block

In this course, you will build entities and user interface objects to work with instances of these entities in Process Experience. In the chapters to come, you will build forms to display instance data and layouts to capture entity instances in their various states. The first user interface object you will build for Process Experience is a list.

As previously stated, a *list* is a business-relevant categorization of entity instances. Examples of lists include open order requisitions, a catalogue of products, or a list of customer contacts. Lists are arranged in the lists panel on the Process Experience home page. End users can use well-designed lists to find entity instances easily. Lists can display all instances of an entity, or they can filter the instances in a business-relevant manner (e.g., only those orders which have not been shipped). Lists are added as building blocks to an entity.

Add a list building block to an entity in the same way you added a property or relationship. Each list will have a unique name and a display name which is visible to the end users.

Figure 4-12:

Details for list building block

The screenshot shows a configuration dialog for a 'List' building block. The fields are as follows:

- Display Name:** Enter a Display Name
- Name:** Untitled_List
- Default Category:** (empty)
- Display Count:** Never (radio button is not selected), 200 (radio button is selected)
- Results per page:** All (radio button is selected)

Display name The name of the list which is visible to end users. This is how the list will be portrayed on the Process Experience desktop.

Name A unique name which will identify the list in the Process Platform database.

Default category Lists can be optionally arranged into arbitrary categories. Categorizing lists is especially relevant if you have many active lists. Lists are created in the default category for an end user who is accessing the system for the first time, but the end user can rearrange the lists based on their own personal requirements.

Display count When the number of instances is displayed on the Process Experience desktop, you may also choose to display the total number of instances. The valid options for the Display Count are:

- Never. Never show the total number of instances.
- Only in results. The total number of instances is shown only in the Results panel.
- In lists and results. The total number of instances is shown on the Results panel and with lists.

Results per page The number of instances to show on a single page of Process Experience. Results can be broken into sets and arranged across several pages (the default is 200 instances per page), or you can choose to display all instances on a single page.



If you choose to display all instances and there are many, the Process Experience desktop will take a long time for the end user to load. It is recommended that the “All” option is selected only when working with a small subset of all entity instances (e.g., open orders, discontinued products, etc.)

Visible to end user You must enable this check box if you want the list to be visible in Process Experience.



Create a list for the product entity in Process Platform

1. Log in to Process Platform as the **analyst@training.local** user and open the Northwind Workspace if you have not already.
2. Expand the Warehouse Application and Entities folder.
3. Double-click the Product entity to open it in the editor.
4. Click **Add**.
5. Select **List**.
6. Set the Display Name to **Product Catalog**.
7. Set the Name to **ActiveProducts**.
8. Set the Default Category to **Products**.
9. Set the Display Count to **Only in results**.
10. Set the Results per page to **200**.
11. Make sure that **Visible to end user** is selected.
12. Click **Add**.

Properties and filters on lists

Once the list has been created, you have the option to add an action bar, a layout, additional properties, or apply filters to it. An action bar is a building block which contains buttons that perform certain actions. A layout is a building block which describes the default panels to display with the list items. A filter will select some of the instances for the list, such as only those orders that are incomplete or only those customers in your region.



You will build action bars later in this chapter.

You will build layouts in the following chapter.

By applying a filter to a list, you are limiting the number of instances in the list as well as limiting the number of instances to which the end user will have access. For example, you may want to restrict the end users from accessing discontinued products. Setting a filter can only be done in the entity editor: once the filter has been set, it cannot be removed in Process Experience. The end user, however, may apply additional filters on the set of instances in the list.

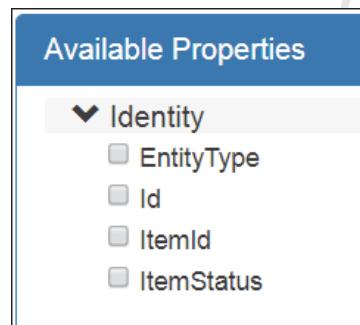


An end user may not be able to apply further filters to the instances in the list if the security setting prevents the user from interacting with the filtered field.

Identity attributes The identity building block was added automatically when the entity was created. As you learned, the identity building block functions as a primary key for the entity instance. You may add identity as a feature to your list. The identity building block includes several features which may be added to your list:

- Id: an integer which represents the sequential value of the item instance (the “business ID”).
- ItemId: a data code which represents the item in the table of entity instances (the “system ID”).
- EntityType: a data code which represents the entity in the table of entities.
- ItemStatus: an integer which represents the current state of the item (0 = draft).

Figure 4-13:
Identity features
available for lists



List properties After a list is created, list properties and filters are both accessible through a button labeled Configure in the editor. In list configuration, the properties option allows you to choose which property building blocks of the entity will be displayed in Process Experience. The property and identity building blocks will be displayed under a list of available properties. From the left side of the list editor, you can select individual properties to include in the display. As properties are selected from the list, they are added to the right, to be shown in the results panel on Process Experience. On the right side of the list editor, you can indicate how each property will be displayed in Process Experience.

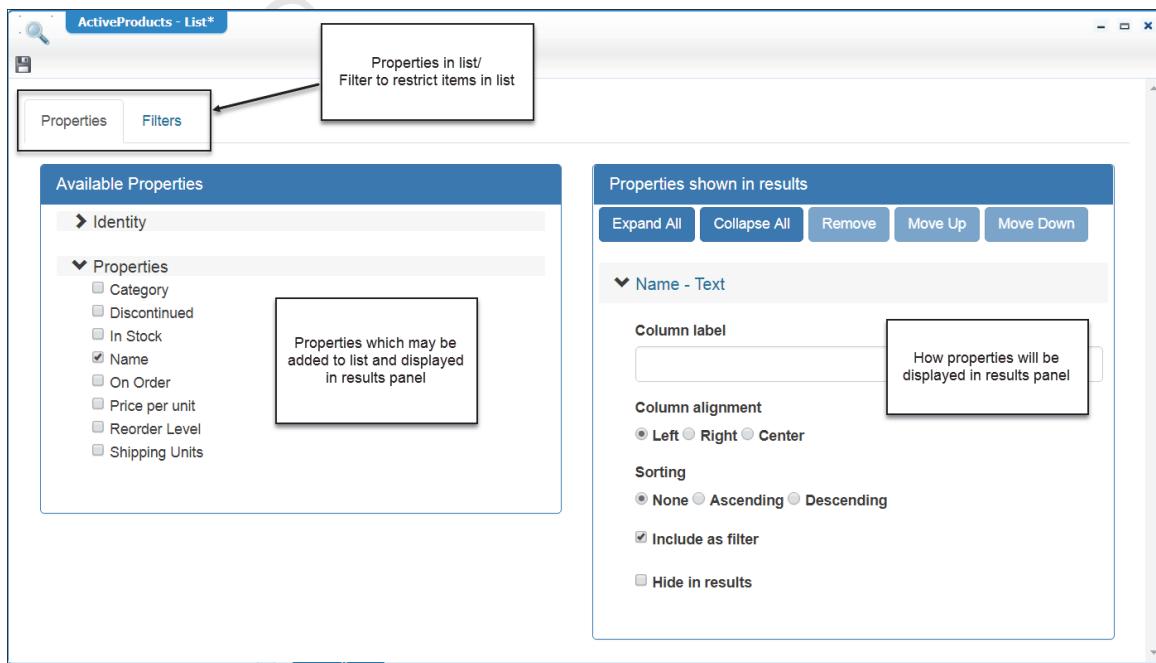
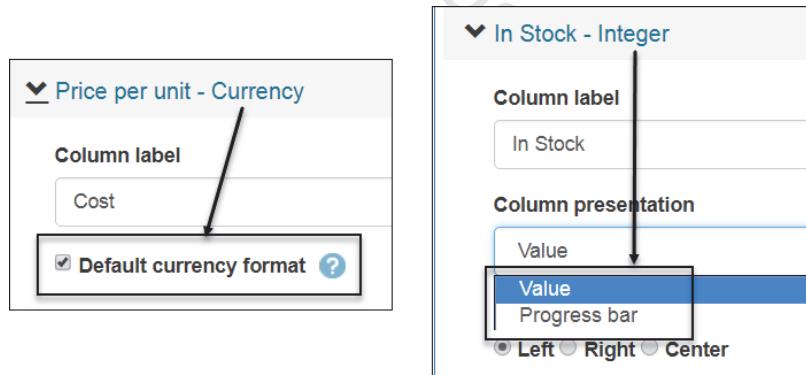


Figure 4-14: List properties in the list editor

Options for displaying properties in a list include:

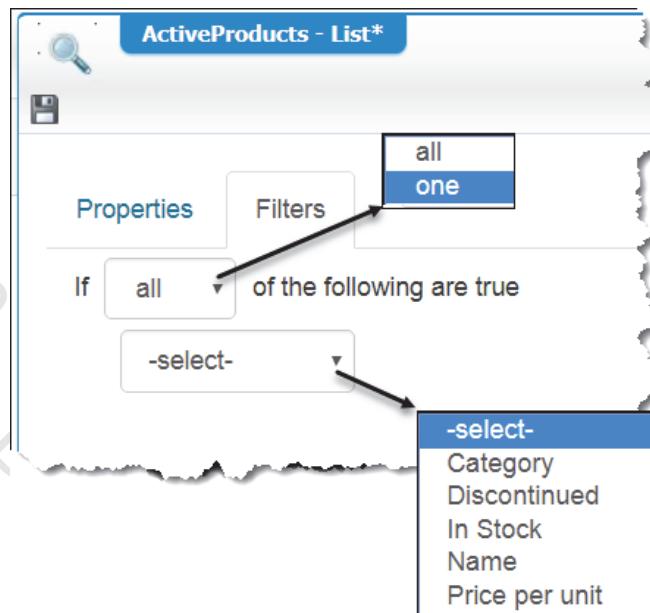
- Position of the property in the display. Use the Move Up and Move Down buttons to place the property in position relative to the other Properties selected for the list. Expand All or Collapse All will allow you to visualize all the selected properties easier for editing purposes. Remove functions the same as de-selecting the property from the left pane: it removes it from the Results panel.
- Column label. A heading to display in the column for the property's value. This label may be different from the property's display name.
- Column alignment. Values for the column may be aligned left, right, or center. It is common to use left alignment for text and right alignment for numbers, especially currency.
- Sorting. Values in the columns may be sorted in ascending or descending order, or not sorted at all. If several columns are sorted, they are sorted in priority of list placement.
- Include as filter. If you want the user to be able to further filter results based upon this property's value, select this check box.
- Hide in results. The property will be present in the list, but will not be visible to the user.
- Secondary features. Some properties, such as those with numeric data types or currency, will offer secondary features which may be set with the property. For example, currency types include an option which allows the user to display the values in default currency format. Some properties with numeric values may allow the designer to display the values as a numeric value or as a progress bar.

Figure 4-15:
Some secondary
features on list
properties based on data
type



List filters The other tab in the list editor is for applying filters on the list. When defining filters, build conditions which must be evaluated to either true or false. Designers may build conditions upon any property which will be included as part of the list. You may choose to apply the filter(s) if any or all of your conditions are true.

Figure 4-16:
Inclusive and exclusive
options for applying filter
on list properties



To establish a filter, select one of the list properties. Based upon the property type, the Filter editor will provide you with logical operators for building the filter, then set the value for the matching operand:

- Equal to/Not equal to: used to determine if the value is an exact match of the operand. This operator is available for text, numeric, and Boolean types. If the property is a Boolean, the operand will either be true or false.
- Empty/Not empty: used to determine if a value is present or not.
- Between: used to specify a range of values. Only those values which fall in that range will be accepted by the filter.
- Contains: used to determine if the operand value can be found in the property's value. Only those values which can be found in the property will be accepted by the filter. This operator is only available for text types.
- Any of: used to specify a list of acceptable values. Only those values which are part of the list will be accepted by the filter.
- Greater/Less Than/Equal To: used to specify a lower or upper limit to the acceptable range of values. This operator is only available for numeric values.

- Within: used exclusively for date types, this operator will evaluate the date operand to see if it falls within a described date range.



List filters use an expression language to create subsets of entity instances. An expression is formatted according to the following syntax:

`$ (expression)`

Expression language is a powerful tool for building and applying filters and requires some rudimentary understanding of programming principles. In this course, you will use the builder tools as much as possible instead of the expression language.

You can add as many such filters to the list as you choose by clicking the + button. Remove a filter from the list by clicking the minus (-) button. If you need to add or nest more complex evaluation statements (that is, more “if one/all the following are true...”), click the evaluation button ().



Before you begin this example, you must have created the list from the previous example.



Add properties and filters to a list

- Open the Product entity from the Warehouse Application if it is not open already.
- Select the ActiveProducts list from the list of building blocks.
- Click **Configure**.
- On the Properties tab, select Name, Price per Unit, In Stock, Category, and Discontinued in that order.
If you make a mistake with the order, select the correct property from the right panel and use the Move Up/Move Down buttons as appropriate.
- Set the Column label to **Name**.
- Set the Column alignment for Name to **Left**.
- Set the Sorting for Name to **Ascending**.
- Select the **Include as filter** check box for Name.

9. Set the other following details for the remaining Properties:

Property	Column label	Alignment	Sorting	Include as filter	Hide in results
Price per Unit	Cost	Right	None	True	False
In Stock	On Hand	Left	None	False	False
Category		Left	None	True	False
Discontinued		Left	None	False	True



If you leave the column label blank, Process Experience will use the display name of the property by default.

10. Select the *Filters* tab.

11. Set the evaluation criteria to **one**.



Even though you will only build one condition in this filter, it is often more efficient to choose the ‘one’ criteria: if you choose ‘all’, the evaluation engine will check all of the criteria. If you choose the ‘one’ criteria, the evaluation engine stops after evaluating the first true condition.

12. From the select drop-down, select **Discontinued**.

Although you set Discontinued to be hidden from the results, it is still part of the list.

13. Set the operator for Discontinued to **equal to**.

14. Set the operand for Discontinued to **False**.

The list will only select those products which are not discontinued (i.e., active products).

15. Click Save.

16. Close the ActiveProducts list.

17. Save the Product entity.

18. Close the editor.

19. Validate the Warehouse Application.

Action bar building block

An action bar is an optional building block which may be added and configured with decorative building blocks in an entity. On top of standard building block properties, such as a display name and a name, an action bar contains buttons which allow users to work with an entity, such as opening an entity instance, deleting an instance, uploading or downloading files, or displaying the audit history of an entity instance. Buttons which may be added to an action bar are either static or dynamic in nature:

Static action bar buttons Static action bar buttons depend upon the building blocks which are added to the entity. For example, since every entity has an identity building block, there are two buttons which are available with every action bar: open and delete. Open allows users in Process Experience to open the entity instance, normally from a list. Delete allows users in Process Experience to delete the entity instance.



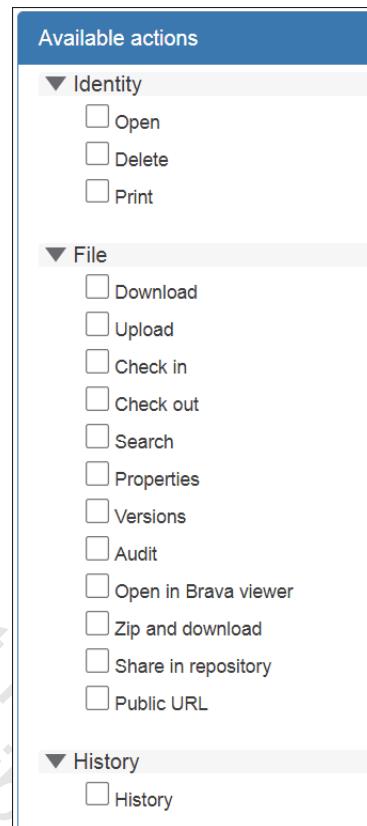
You may choose to control who has access to the delete capability based upon their role. You may configure this option using security. You will learn about security in an upcoming chapter.

If you have added the file building block, other static buttons related to file handling will be available, such as Upload and Download. These buttons allow Process Experience users to work with a file from an entity instance. Some options are only available if there is a file already attached to the entity instance.

If you have added the history building block, a History button will be available. The History button will allow the Process Experience user to review the history of the entity instance. The entity instance's history will be a complete audit trail for the item which may be reviewed. This option will be useful for managing sensitive client issues or highly regulated products.

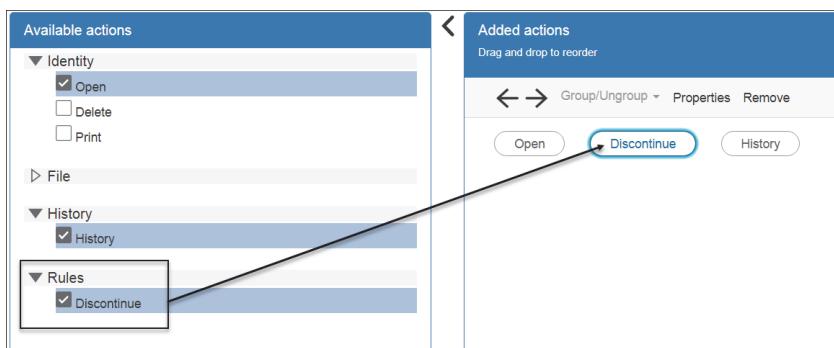


You will add the file and history building blocks to entities in another chapter.

**Figure 4-17:**

Static buttons available when designing action bars

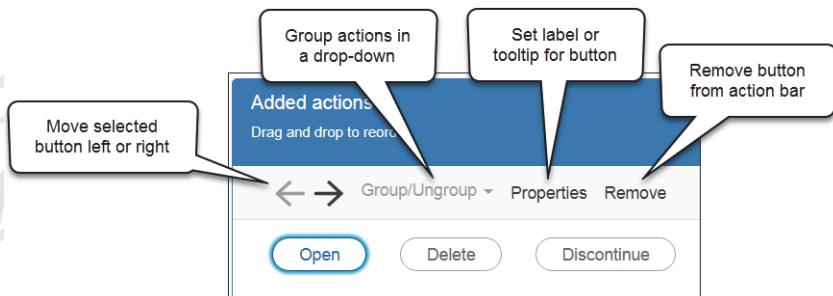
Dynamic action bar buttons Some buttons may be created dynamically and added to an action bar, using the rule building block. As you will learn in an upcoming chapter, you can build a rule with the rule building block to evaluate a certain condition and then perform actions, such as setting properties of entities or starting a process. As an entity modeler, you can create a rule which will perform one of these actions, and then create a button for an action bar which will execute your rule. The name on the button will be the display name used for the rule.

**Figure 4-18:**

Rule-based action bar buttons

Configuring and editing action bar buttons Once buttons have been added to an action bar, you can choose to re-arrange and edit the buttons as they are placed. When you select the available actions to add to the action bar, buttons which correspond to the actions will be automatically added. You can either drag the buttons to re-arrange their order, or use the convenient arrow buttons to select and move the buttons in the bar. If you click the Remove button, the button will be removed from the action bar and the action will be de-selected from the list of available actions. This has the same effect if you were to de-select the available action from the list itself.

Figure 4-19:
Positioning buttons on an action bar



Build an action bar for product

1. Open the Product entity from the Warehouse Application.
2. Click **Add**.
3. Select **Action bar**.
4. Set the Display Name to **Default**.
5. Set the Name to **abarDefault**.
6. Set the Description to **Default action bar**.
7. Click **Add**.
8. Select the **abarDefault** action bar from the list of action bar building blocks in the entity.
9. Click **Configure**.
10. Under the Identity actions, select **Open** and **Print**.
11. Click **Save**.
12. Close the action bar editor.
13. Save and close the entity.
14. Validate the Warehouse Application project.



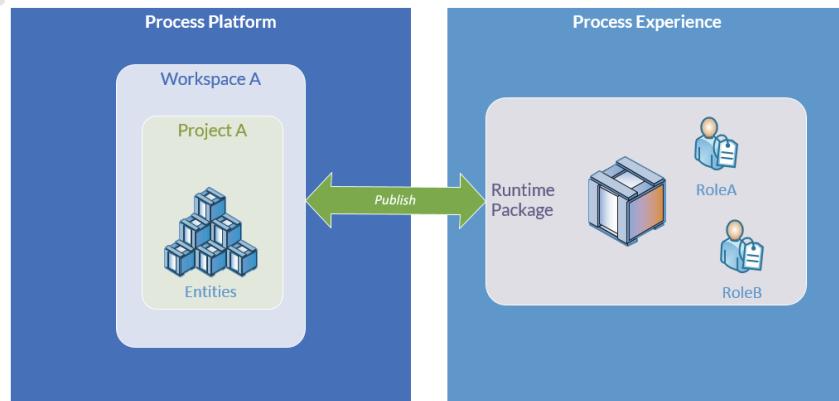
Add an action bar to a list

1. From the Warehouse Application project, open the Product entity.
2. Select the **ActiveProducts** list.
3. In the right panel, change the Action Bar field to **Default**.
4. Save and validate the entity.

Publishing a solution to Process Experience

In order to use an entity-based solution in Process Experience, the application must be published from the collaborative workspace in Process Platform to Process Experience. The act of publishing takes your modeled solution, generates run time code, and makes that code available for Process Experience.

Figure 4-20:
Publishing solution to
Process Experience



Publishing and CAP files There are two methods of publishing solutions from Process Platform to Process Experience. The 'simple' method involves projects in the collaborative workspace which contain only entities. When your solution project contains only entities (and other documents, such as roles or homepage layouts), publishing the project creates a run time package which is automatically deployed to Process Experience. If your application project contains other documents, however, such as process models, you must create a deployable CAP file.

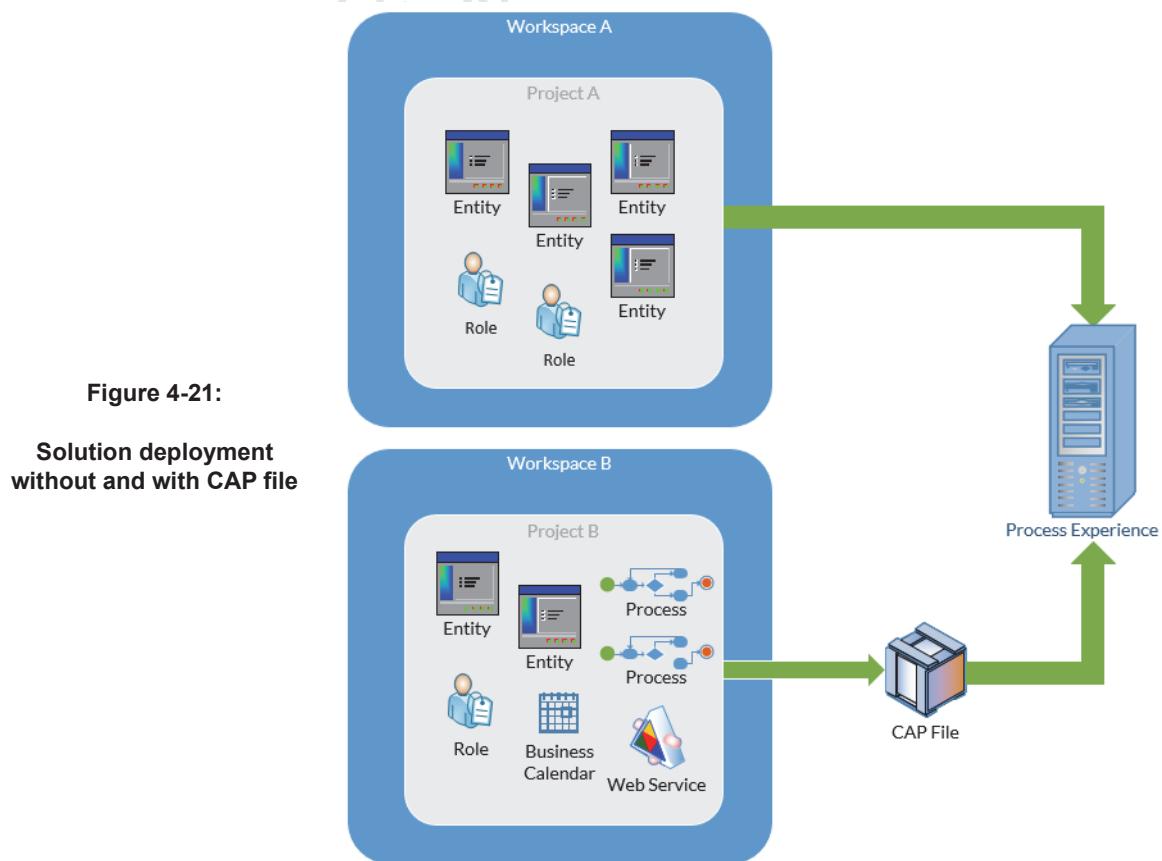


CAP file: Cordys Application Package file.

The projects with which you work in Process Platform's collaborative workspace may contain many design documents. Entities are a type of design document. Other documents which development teams may include in a project are business process model documents, homepage layouts, role documents, business calendars, metadata design documents, web service interface documents, and so on. Projects which contain complex documents cannot be simply deployed to Process Experience; they must be wrapped in a CAP file package and that CAP file must be deployed by a system administrator to the Process Experience run time.



In course 4-4913, you will learn to publish and generate CAP files for deployment.



In either case, publishing is a one-step process. Simply select the project from the workspace, right-click the project, and select Publish to Organization.



A project must be validated before it can be published.

Process Experience Administration Console

In the entity designer, you can perform one-step publish with the publish (button).

After the entity-based solution is published to Process Experience, you must configure solution security: mapping elements of the solution to the roles who have access to it. This task is performed in the Process Experience Administration Console.



You will learn more about security and roles in an upcoming chapter. Solution security, however, is a complex issue and the security building block only addresses one aspect of the issue.

The Administration Console is a separate web page:

`http://<server>:<port>/home/<org>/app/admin/web/config`

In your solution, that will be:

`http://localhost:81/home/Warehouse/app/admin/web/config`



A shortcut has been provided for you in the Favorites bar for easy access.

The Administration Console contains a summary of the solutions, as workspaces, which have been deployed to the organization, and offers drill-downs into configurable elements of each solution.



Figure 4-22: Process Experience Administration Console

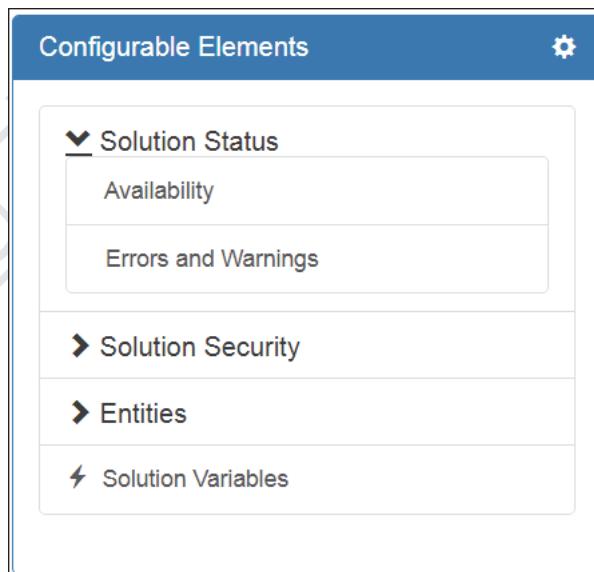
The deployed workspaces are on the left. By default, there should always be workspaces for Identity Components and Inbox Task Management.

The center grouping contains the set of configurable elements. The administrative user will select the workspace from the left and the configurable elements appear in the center. There are certain configurable elements for every workspace:

Solution status Solution status captures the current state of the deployed solution. There are two elements which are available:

- Availability: whether the deployed solution is available to users in Process Experience. A solution which is deployed is, by default, available. However, the administrator can choose to temporarily disable the solution. This is set in the Configuration Properties grouping on the right.
- Errors and Warnings: this element captures all the errors and warnings which have been generated by the system for the solution. The individual errors and warning are organized under their respective titles under the Properties grouping.

Figure 4-23:
Solution status
configurable elements



Solution security This section determines who has the ability to work with the overall solution (not individual entities, but the complete solution) by the user's role. The aspects with which the user can work with the solution are:

- Administer Solution: user roles able to use the Administration Console to work with the deployed solution and set the configurable elements.
- Build: user roles able to modify the definition of the deployed solution. This includes the ability to delete run time elements from the deployed solution.
- Include: determines the interaction between solutions. If one solution depends upon another, for Web service interactions, business process models, and so on, use this setting to refer one solution to another.
- Manage Solution: determine which users, by their role, are able to manage the configuration of a solution. This ability includes the control to manage different versions of a deployed solution.
- Manage Solution Security: determine which users, by their role, are able to modify a solution's security settings.

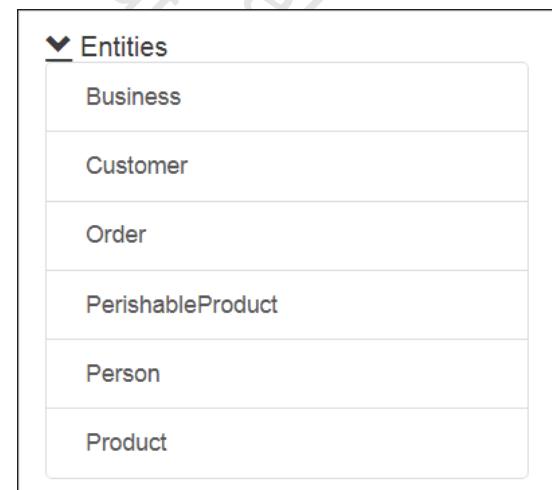
- Use Solution: determines which users, by their role, can use the solution in Process Experience. A user cannot access any entities or entity instances of the solution if they do not have this permission.

Figure 4-24:
Solution security
configurable elements



Entities These are the entities which were deployed with the solution. By default, just about every user has access to all the published entities. With the entities configurable element, however, you can associate specific entities to user roles, thereby controlling who has access.

Figure 4-25:
Entities configurable
element



Layout Security Later in this course, you will learn how to build a home page layout. A home page layout describes the panels which users will see on their home page when they log in to Process Experience.

When you add home page layouts to your solution, you can administer their security in this console. Use the console to assign security roles to the home page layout; this will dictate the groups of users who have the ability to use this home page layout.

History Logs	The History building block, which you will use later in this course, is an optional building block which may be added to an entity. It can maintain a record of every instance created, deleted, changed, or even accessed. This audit record is contained in a special document called a <i>history log</i> . The history logs are summarized in the administration console.
Solution variables	You can use the administration console to define solution variables that represent values that are subject to change, such as a discount rate. For example, assume that you need a rule that discounts a sale amount by a discount rate percentage that changes based on various factors. By specifying the discount rate as a variable, you can then use that variable (rather than a fixed rate) in the rule. When the discount rate changes, you simply change the value in the Administration Console rather than reconfiguring all the rules in which it is used. Another possible use for a solution variable could be as a part of the URL in a web content panel. The Name of a solution variable must exactly match the name of the element that it represents. For example if the variable represents a property called Discount, it must be named Discount.
	To add a solution variable, click the Create button under the Configuration Properties grouping. The variable will have a name unique to the solution and a value. That value is stored in the system as pure text, so you may need to do some manipulation in order to use it as another type of data.
Default entity roles	There are several roles which are normally published with a solution. On top of that, there are roles which are added by default when Process Platform, Process Experience, and the entity modeling module are installed. When the entity modeling module is installed in Process Platform, there are predefined roles that are also installed. These roles are used to set the initial security of the solution. When a solution is deployed to Process Experience, roles are assigned to permissions automatically. The default entity roles are:
Entity Runtime Administrator	The Entity Runtime Administrator role is defined for a user who will have access to administer a deployed solution which includes entities. By default, when a solution is deployed to Process Experience, the Entity Runtime Administrator is automatically granted access to the Administer Solution, Manage Solution, and Manage Solution Security configurable elements.
Entity Runtime Developer	This role is defined for a user who is responsible for developing and building aspects of the deployed solution which includes entities. By default, when a solution is deployed to Process Experience, the Entity Runtime Developer is automatically granted access to the Build Entity and Include Entity configurable elements.
Entity Runtime User	This role is intended for Process Experience end users who will be working with a solution which includes entities. By default, when the solution is deployed to Process Experience, the Entity Runtime User is automatically granted to the Use Solution element.



Administer Northwind workspace in Process Experience Administration Console

1. Save and close any open entities you may have.
2. Validate the Warehouse Application.
3. Right-click the Warehouse Application and select **Publish to Organization**.

Only those projects which can be validated will be published.

4. Open a new browser window.
5. Use the bookmarks to navigate to the Process Experience Administration Console:
`http://localhost:81/home/warehouse/app/admin/web/config`
6. Log in as user `analyst@training.local` if you need to.
You should not need to log in: you should automatically be logged in through the single sign-on aspects.
7. Select the `NorthwindWarehouseApplication` from the list of Workspaces.
There should not be any warnings.
8. Expand the Solution Security configurable element.
9. Select Use Solution.
10. On the right, under the Roles grouping, make sure the `Entity Runtime User...` and `everyoneInwarehouse` roles are checked.
You may need to scroll to the bottom of the role list to find `everyoneInwarehouse`.

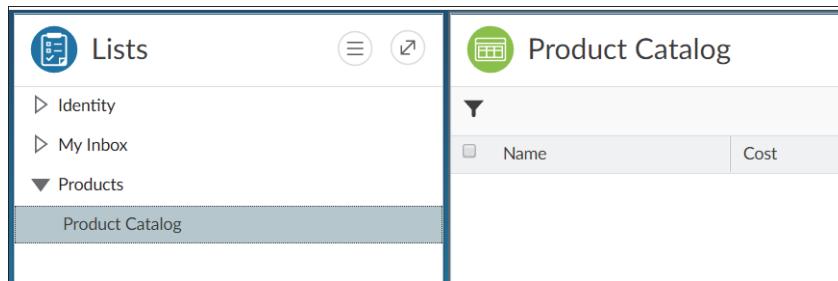


There's no need to save your settings when using the Process Experience Administration Console: your settings are applied automatically once selected.

11. Open another new browser window.
12. Navigate to the Process Experience home page:
`http://localhost:81/home/warehouse/app/processExperience/web/perform`
13. Log in as user `analyst@training.local` if you need to.
14. Click Home Page > Home Page.

15. Confirm that the Product Catalog list appears under the lists panel.

Figure 4-26:
Product Catalog list available in Process Experience



16. Click the Filter icon to confirm that you are able to filter on name, cost, and category.



You will not be able to create any new items - Customers, Products, or Orders - until you build forms to support them. You will build forms in the following chapter.

You will not be able to see the action bar, either. The action bar buttons (Open and Print) refer to items (i.e., instances of the entity). Because you have no items, you will not see the action bar.

Summary

Having completed this chapter, you should be able to:

- Navigate Process Experience
- Configure a list from an entity for Process Experience
- Add an action bar to an entity
- Configure a list with an action bar
- Publish and administer your solution in Process Experience

Exercises

1. Create action bars for the Customer and Order entities with the following properties for both:

Display Name: Default
Name: abarDefault
Description: Default action bar
Available actions: Open, Print

2. Create a list for the Customer entity with the following properties:

Display Name: Client List
Name: AllCustomers
Default Category: Customer
Display Count: Only in results
Results per page: 50
Visible to end user: True
Action Bar: Default

Configure the list to include the following properties:

Property	Column label	Alignment	Sorting	Include as filter	Hide in results
Id	ID	Left	None	False	False
Client Code	Client	Left	Ascending	True	False
Address		Left	None	False	False
Phone		Left	None	False	False
City		Left	Ascending	True	False
Country		Left	Ascending	True	False

3. Create a list for the Order entity with the following properties:

Display Name:	Order List
Name:	AllOrders
Default Category:	Order
Display Count:	Only in results
Results per page:	200
Visible to end user:	True
Action Bar:	Default

Properties to include in list:

Property	Column Label	Alignment	Sorting	Include as filter	Hide in results
Id	Order Number	Left	None	False	False
OrderToCustomer/ Client Code	Customer	Left	Ascending	True	False
Order Date	Ordered On	Left	Ascending	True	False
Employee	Sales	Left	None	True	False
Requested On	Required By	Left	Ascending	True	False

4. Create another list for the Order entity with the following properties:

Display Name:	Outstanding Orders
Name:	OutstandingOrders
Default Category:	Order
Display Count:	In lists and results
Results per page:	50
Visible to end user:	True
Action Bar:	Default

Properties to include in list:

Property	Column Label	Alignment	Sorting	Include as filter	Hide in results
Id	Order Number	Left	Ascending	False	False
OrderToCustomer/ Client Code	Customer	Left	Ascending	True	False
Order Date	Ordered On	Left	Ascending	True	False
Shipped Date	Shipped On	Left	None	False	True
Employee		Left	Ascending	True	False
Requested Date	Required By	Left	Ascending	True	False
Ship Via	Shipper	Left	None	True	False

Filters to include in the list:

- If **one** of the following is true...
- **Shipped On** is empty

5. Save, validate, and publish your project to the organization. Confirm that the lists appear in Process Experience.
6. Make your changes available to others in the SVN repository by committing your exercises with the name: "Chapter 4 - Lists, Action Bars".

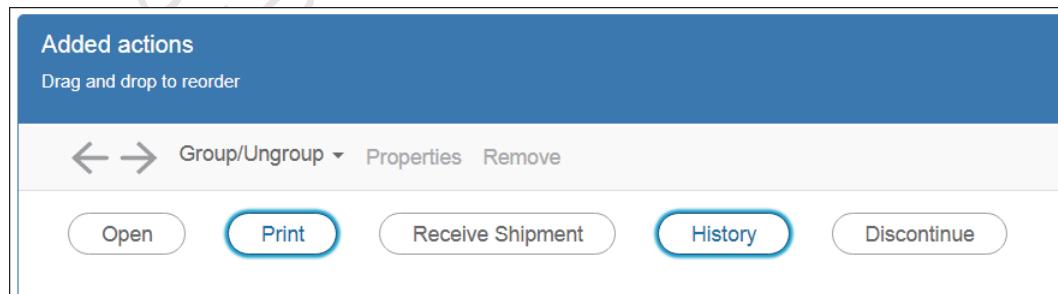


Tips, tricks, and traps

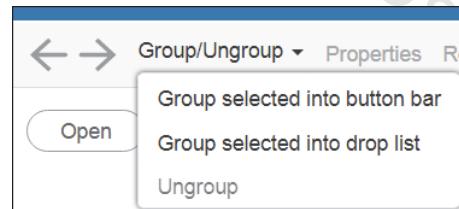
1. Group buttons on an action bar

An action bar with a lot of buttons can become cluttered, but you can group buttons together in custom sets.

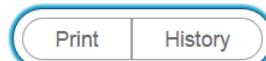
- Open and configure the action bar.
- On the right, in the Added actions section, select the first button you want to group, then hold down the CTRL key and click the other buttons you want to add. Each button to be added to the group will be highlighted.



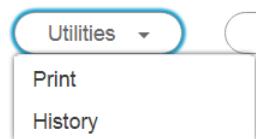
- Click **Group/Ungroup**. The selected buttons can be added into a button bar or into a drop list. If you selected a button group, you can also ungroup them.



- Button bar option:



- Drop list option with label:



Open Text Internal Use Only
Do Not Distribute

5. Process Experience user interfaces: forms and layouts

Objectives

On completion of this chapter, participants should be able to:

- Create a form for an entity
- Explain layout panels and their composition
- Create a layout for an entity
- Create a homepage layout
- Create a custom home page for Process Experience
- Add a discussion building block to an entity

Overview

Users in Process Experience will interact with instances of your entities by their *forms*. A form represents a visual interpretation of the data in an entity at a given state in time. Some aspects of the data will be editable based upon the user's role and level of authorization in the system. The form can present any property of the item to the end user. This extends to other building blocks as well, such as buttons specific to lifecycle and rule, or interactive controls for discussions.

Layouts are customized user interfaces for your solutions. End users may also choose to build their own, customized user interfaces by choosing which columns are displayed, or how search results are filtered. Layouts organize Process Experience panels in one of two ways:

- *Item layouts* display instances of entities, and contain panels that display various aspects of that instance (usually forms) at a particular moment in time, or at a particular phase in a lifecycle.
- *Homepage layouts* provide users with a functional landing page to run choreographed aspects of the application. Home page layouts (or application layouts) are not dependent upon the instance of an entity.

Layouts will contain a number of predefined panels arranged into specified areas, or zones, presenting users with the information they need in a way that is optimized for interacting with the tasks they are supposed to perform. The layout can be different depending upon the phase of the lifecycle.

What you will build in this chapter

- Form building block: forms are reusable user interfaces. You will build forms for all of your entities.
- Layout building block: layouts are customizable workspaces for users in Process Experience. A layout is composed of a number of panels, which could include forms.
- Homepage layout. Whereas layouts are built for individual entities, a homepage layout is built for a Process Experience home page. The layout is built analogously to other layouts: an assembly of relevant panels.
- Discussion building block: discussions may be added to entities in order to allow team members to collaborate in an ongoing dialog of topics and replies.

Timing

Lecture: 75-90 minutes

Exercises: 60-75 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform 16.3 Entity Modeling Guide

Forms

An *item* is an instance of an entity. Item data is presented to the user by means of forms. Forms are visual presentations of the instance data, as well as other building blocks, such as titles.

For each entity, you should consider creating two forms: one form named “Create” and another named “Default”. The Create form is used whenever an end user creates an item in Process Experience. For example, whenever Process Experience users want to create a customer or order item, they click the create button (+) on the Process Experience home page. If you have developed a Create form, options named Customer or Order will be available. Otherwise, users can create a layout (which will also be discussed this chapter) with the *Use to create new items* option enabled. In this circumstance, the whole layout will be used for instance creation.

The Default form will be used whenever a user opens an entity's instance for edit in Process Experience. It is possible that, during different phases of an entity's lifecycle, you may choose to present certain fields for edit. For this reason, forms for editing are normally attached to a layout. The layout will then be attached to the phase of the lifecycle. Layouts can also be attached to a list: for example, when you design a list, you can choose the layout which will be used for displaying the contents.

You can create as many forms for an entity as your business requires. Later, when you create a layout, you can include the form (or forms) you built as panel(s) on the layout. It is common to create many forms and layouts to display data in a specific way for specific users or groups of users and/or work scenarios.



End users who have permission to Create instances from entities must also have Update permission to allow them to modify the properties of the instance.

Create a form

The form is a building block of the entity modeling tool, available under the Add button. There are a limited number of features which must be defined when adding a form:

Form
Display Name
Enter a Display Name
Name
Untitled_Form
Description
Untitled_Form

Figure 5-1:

Form details when designing a new form

Display Name The name which is displayed to the users.

Name The name of the form as stored in the database. This name should be unique for the entity.



For each entity, you should have one form named “Create”, in which you will create instances, and one form named “Default”, in which you will edit or modify instances. You may add more forms if you add more layouts.

Description An optional description to explain the purpose and intent of the form.



Add a generic business customer form

1. Open the Northwind Workspace in Process Platform, if it is not already open.
2. Expand the Warehouse Application and Entities folder.
3. Double-click the **Business** entity to open it in the editor.
4. Click **Add**.
5. Select **Form**.
6. Set the Display Name to **Create Business Customer**.
7. Set the Name to **Create**.
8. Set the Description to **Default form for creating business customers**.
9. Click **Add**.

When a form is added, it is automatically assigned a system-generated URL. This URL can be copied and used in other Process Platform artifacts or in external artifacts, such as portals, as a simple means of directly accessing the form.

Figure 5-2:
Generated URL for simple access to form

Form Url
processExperience/web/perform/create/005056327662A1E7ABDFA5A37A7D80DE
Configure

Form configuration

Once the form has been created, there is a significant amount of configuration which must be performed in order to make the form relevant for users of Process Experience.

On the Form Details in the entity editor, you will notice a button labeled Configure. This button directs you to the form configuration editor.

The form configuration editor is divided into two sections: the Components panel and the Presentation panel. These panels will help you visually design the form.

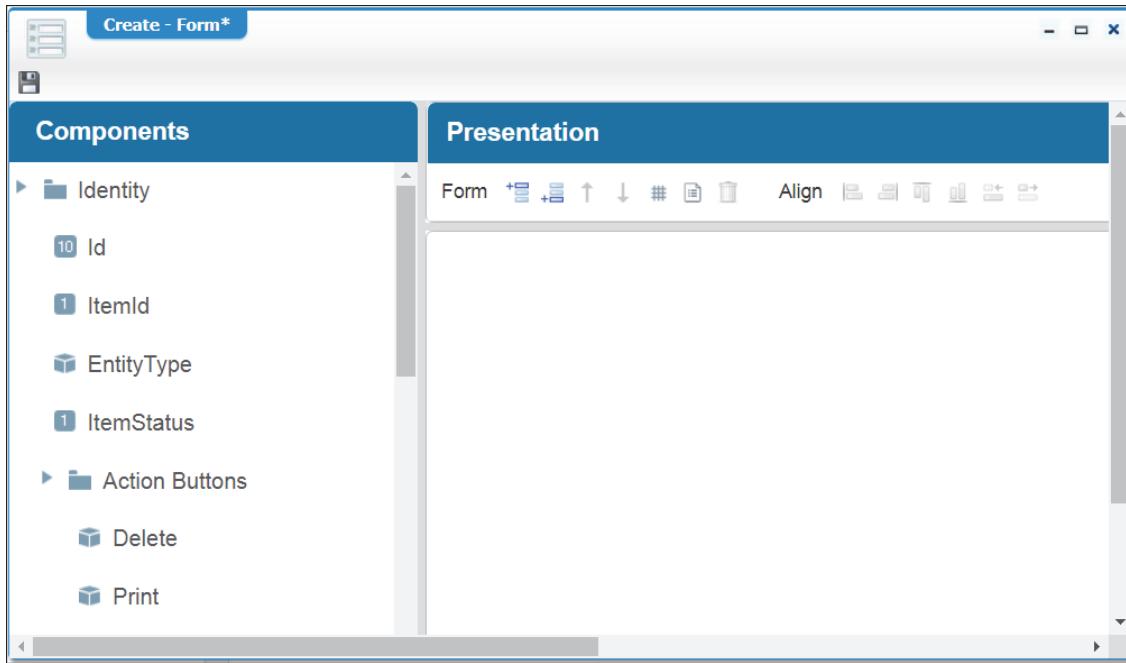


Figure 5-3: Form configuration editor

Components panel The Components panel contains all the relevant properties of the entity, such as:

- Identity attributes: these include
 - Id: the unique identifier of the entity instance, the business ID,
 - ItemId: the unique identifier of the metadata, the system ID,
 - EntityType: a unique code which represents the entity metadata,
 - ItemStatus: the system state in which the entity instance exists (such as created or obsolete)



The ItemStatus does not represent a business state for the object, rather a system state. You can capture the business state using a lifecycle.

- Related action buttons, such as deleting or printing an instance. These buttons can be placed directly on a form instead of in an action bar.

- Relationships: each relationship is captured by its multiplicity (either [0..1], [0..*] or [1..*]). Each relationship is itself a full recursive set of components containing identity, relationship, property building blocks, and so on,
- Title building blocks,
- Property building blocks, and
- Other forms (subforms)

In form configuration, you can drag any of these attributes and drop them onto the Presentation panel in order to control which properties will be visible and/or editable on the form.

When you drag a building block onto the presentation, the editor will automatically associate the property with a field and provide it with a label that matches the display name.

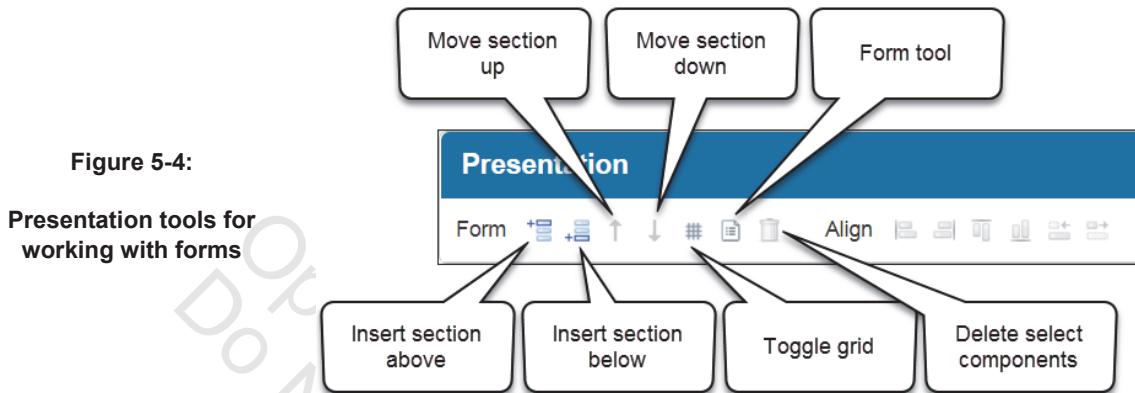
Underneath the properties in the Components panel, you will find another palette to help you design the form. This palette contains visual tools to help arrange and organize the fields on the form more efficiently. The palette includes:

- Container: groups components together in a titled box.
- Tab Container: categorizes containers into tabbed panes.
- Horizontal Line: add a decorative horizontal line to the form.
- Image: insert a graphic file.
- Text: add read-only text to the form. The text is not associated to any data value of the item.
- Hyperlink: a hyperlink to another web page.

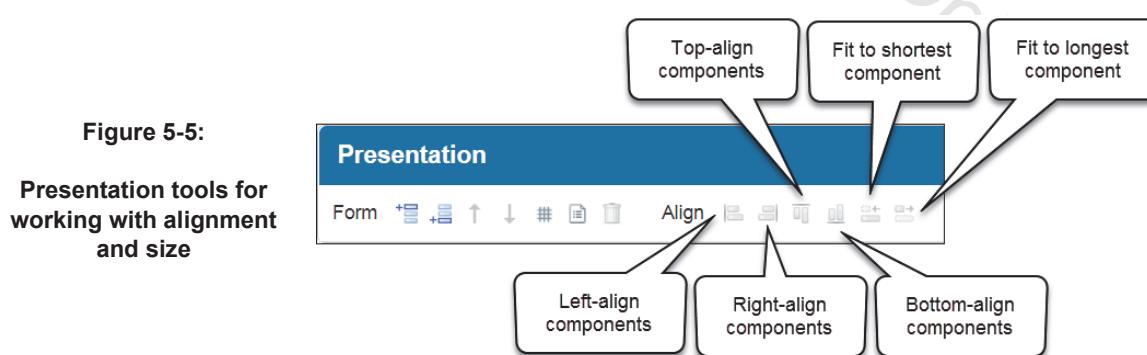


You can spawn another form using the hyperlink tool and the URL of a form.

Presentation panel The Presentation panel captures a WYSIWYG design of how the data for the item will appear to the user in Process Experience. There are a number of tools available on the editor to help you work with the components:



- Insert section above/below: a section is a canvas or abstract container for a group of components. Unlike other containers, a section does not have a visible border. These buttons allow you to divide your presentation workspace into cohesive pieces.
- Move section up/down: move a section and its contained components up or down in the presentation.
- Toggle (show/hide) grid: toggle the grid lines on or off in the presentation panel, allowing you to better visualize how property fields will line up relative to one another.
- Form: show the properties of the form.
- Delete selected components: delete one or more selected presentation components.

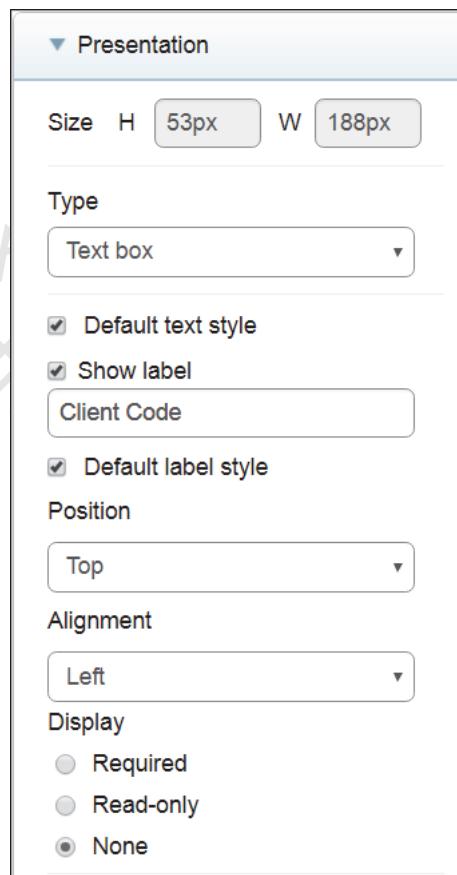


- Align left/right/top/bottom: align all the selected properties with their edges to the left, right, top, or bottom, respectively. You must select more than one property from the presentation panel first.
- Fit to shortest/longest: set all the property fields to the same size, either to the match the shortest or the longest property field, respectively. You must select more than one property from the presentation panel first.

Component properties in presentation Each component that you add to the presentation panel has certain properties which can be changed, such as the font, the type of field, and whether or not it is required.

Figure 5-6:

Detailed properties of a field in the presentation



You can set or change these values on the right side of the presentation panel when you select the component. The detailed properties include:

- Size: these values are set by default when a field is added to a presentation. You cannot set the specific values but you can change the size of the field on the presentation directly.

- Type: the type of field which will be used to display the item's value. These types are selected by default depending upon the data type of related property. For example:
 - Text fields can be displayed with text boxes, text only (i.e., read only), or multi-line text boxes (i.e., allowing multiple lines of text).
 - Number fields can be displayed with text boxes, text only, or progress bars.
 - Date and time fields can be displayed as text or as date/time boxes with calendar/clock selection tools.
 - Enumerated types can be displayed as text, radio buttons, check boxes, or drop-down lists.
- Default text style: set the text style to be the same as the rest of the components in the form or set the text style, including font, size, alignment, and color, manually. Text style can be set globally on the form by selecting the whole form (using the Form tool in the presentation toolbar).
- Show label and default label style: show or hide the display name of the field, which is used as the field's label. You can also choose to manually set the label style (font, size, alignment, color, etc.). By default, the label is the display name of the associated property.
- (Label) Position and Alignment: place the label above or to the left of the field. Align the label respective to its position.
- Required: whether or not a value is required for the field. The user will not be able to save the form if any of the required fields are incomplete.
- Read Only: whether or not the data in the field is editable.
- None: allow default settings for value (read-write).
- Default currency format: use the current locale settings to determine how currency will be displayed.
- Show on mobile devices: a toggle which will include the field in a mobile device presentation. For example, you may choose to include the Title building block in a standard Web presentation, but not on a mobile device. By default, this value is always true.
- Category: you may also choose to assemble fields of the presentation into arbitrary categories. Data fields which are assigned a category may be disabled, hidden, or shown in groups depending upon certain conditions.



Data fields can be dynamically shown or hidden depending upon certain established conditions. You will learn how to accomplish this when you examine the rule building block in the following chapter.



Build the presentation for a business customer form

1. Select the *Create form* from the **Business** entity.
2. Click **Configure**.
3. Click **Toggle grid**.
4. In the Components panel, drag the **Id** property and drop it in the Presentation panel.
5. Change the Type feature on the right side to **Text only**.
6. Under the Show label feature, change the label to **Business ID**.
7. Drag the Client Code property and drop it to the right of the ID field.
8. Set the Client Code field to **Required**.
9. Drag the Company property and drop it beneath the ID field.
10. Set the Display of Company to **Required**.
11. Drag the Contact field and place it to the right of Company.
12. Save and close the form.
13. Your form should resemble the following example.

Figure 5-7:
Create Business Customer sample form



When you're placing fields in the presentation, orange-colored guidelines will be presented to assist you when aligning the fields horizontally and vertically compared to other fields placed in the presentation.

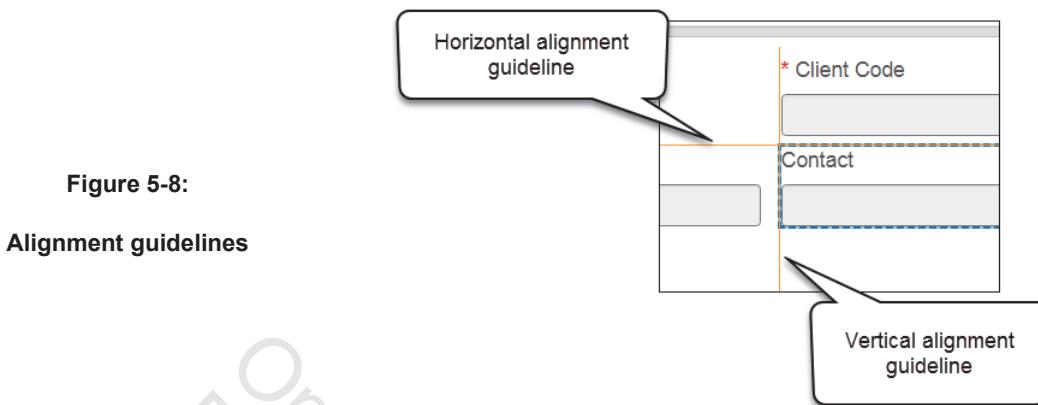


Figure 5-8:
Alignment guidelines

Forms and relationships

Some entities are connected to one another through relationship building blocks. In these circumstances, you may choose to have your users complete data for both instances in a single form, instead of dividing the work over several forms. By nature of the relationship building block, you may add fields from related items to the form of the primary entity. You may also add special containers and utilities which will help you maintain the relationship between the entities.

By adding a relationship building block to a form, you can include properties from the related entity on the form defined in the primary entity. By doing so, you can give the user the ability to see and modify the values of properties of related items in Process Experience. For example, when creating an order, a user would typically need to choose the customer. You accomplish this by creating a relationship from the order to the customer entities. You can then place the customer entity on an order entity form, so that users can access or modify both sets of properties in Process Experience directly from that form.

Relationship multiplicity As you studied in a previous chapter, there are several different types of relationships. Similarly, each relationship will have a defined multiplicity. The combination of relationship type and multiplicity is realized in different ways on a form.

To add a relationship to a form, simply drag the relationship from the list of components and drop it onto the presentation panel. Whenever a relationship is added to a form, Process Platform will assess the type and multiplicity of the relationship and offer different options for interacting with the related items. Typically, this will include a create button, allowing the user to create a related item in parallel with the primary item, but it could include other types of buttons, too, as you'll see below. Complex relationships (e.g., parent-child) are represented differently, as you will learn.

“To one” relationships A “To one” relationship expresses a multiplicity between the primary entity and one related entity. For example, an order will have only one related customer. Relationships of this multiplicity are represented in the form editor with a [0..1] identifier.

When a “To one” relationship is placed on a create form in the form designer, a create button is not included on the form in Process Experience. Instead, related items can be created from a default form or any other form defined in the entity, after the primary item is created and opened from a list. Related items can also be created using the create form defined in the related entity. However, an existing item of a related entity can be associated with the primary item on the primary item’s create form. The related item properties can be modified from the create form just as from any other form, depending upon the user’s permissions in Process Experience.

“To one” relationships may be displayed in a form in the following ways:

- Button(s) only: a search button (🔍) is displayed which will use a defined list to consult and select one from the available related items.
- Text box: to select an item, the user selects the relevant button. The text box is updated when the selection is made.
- Drop list: to select an item, the user selects an item from a drop-down list.
- List box: to select an item, the user selects an item from a list of items presented in a list field.
- Drop list series: When you choose this presentation, the browse button is converted to a container and the properties that you choose from the selected browse list are created within the container. You cannot drag properties into or out of the container. The order of the properties in the Properties list is the same order in which the related properties are shown in the Components pane. The presentation of individual properties inside the container cannot be changed.

To select an item, a user makes a selection from each drop list in succession. The next item only becomes available when the preceding one has been selected.



Items used to populate drop lists or list boxes, or as the subject of searches, are pulled from lists. A list needs to be specified as the value in the **Browse list** property. You built lists in the preceding chapter.

Dynamic loading The dynamic loading option specifies whether the list is loaded by default in Process Experience. This option is not selected by default. This option is available when Button(s) only or Text only is selected as the presentation type.

In Process Experience, dynamic loading behavior for the list works as follows:

1. When the list is browsed, initially no results are shown. The Filter panel is shown by default.
2. Results are retrieved only when the user enters the filter criteria.

Unlike the normal lists, the personalized filter settings are not displayed the next time the list is browsed.

“To many” relationships A “To many” relationship expresses a multiplicity between the primary entity and potentially several related entities. For example, each order could potentially have several line items (in a parent-child relationship), each customer can have several orders (in a peer relationship), or an employee can be a manager for several other employees (in a reflexive relationship).

“To many” relationships are either marked with a **[1..*]** indicator (for parent-child relationships) or **[0..*]** for peer and reflexive relationships.

When a “To many” relationship is placed on a create form in the form designer, create buttons are not included. Instead, related items can be created from, or existing items associated with, the primary item from a default or any other form defined in the entity, after the primary item is created and opened from a list. Related items can also be created using the create form defined for the related entity.

Repeating group and grid containers In a “To many” type of relationship, multiplicity is automatically enforced on the form. It is presumed that the primary (or parent) entity will have potentially several related items. In a parent-child type of relationship, it is presumed that there will be at least one child. For example, when you create an order, there will be at least one, but potentially several line items. Orders without items make no logical sense.

When you drag a “To many” relationship and drop it into the form designer, Process Platform automatically enforces the “to many” relationship with one of two types of presentation containers: a *repeating group container* or a *grid container*. By default, the repeating group container will allow you to display properties of each related entity (which you can drag from the Components panel and drop onto the container) in the same way you design a form. Furthermore, this type of container includes buttons for creating, working with, or deleting child instances.



You should only drop properties of a related instance onto the repeating group container. Similarly, you should only drop properties of the primary instance on the “main” presentation, but not the repeating group container. The repeating group container is only intended to capture repeating instance data.

The grid container behaves like a traditional table with each column representing a property from the underlying related item and each row an instance of an item.

Columns are added to the grid container by dragging them from the properties of the related entity and dropping them inside the grid. Columns are added to the grid container in the order in which they're dropped, but you can drag a grid column to reposition it before or after other columns.

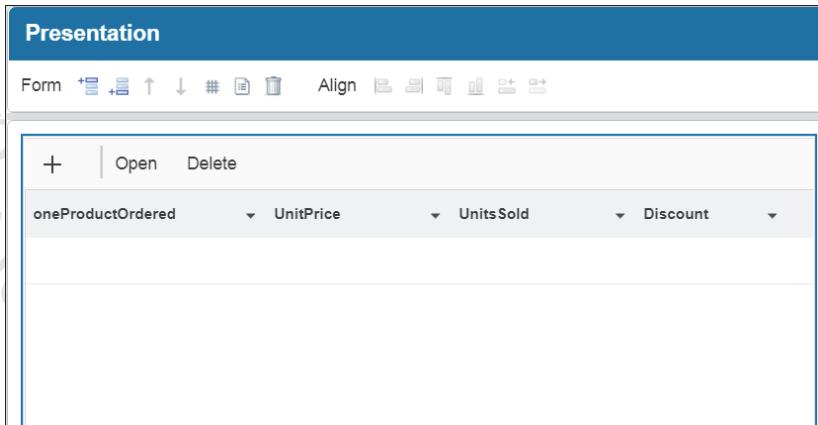


Figure 5-9:

Columns in a grid container

Functions available for relationships in forms Process Platform offers users the ability to interact with related items when their relationships are added to a form. When you design your form, you have the ability to control how users will be able to interact with the related items. These are listed as Actions in the presentation properties. There are a number of actions available:

- Create. When the Create action is enabled, the end user will have the ability to create an instance of a related item from within the primary item. The Create button is enabled by default in the repeating group and grid containers with parent-child relationships.
- Browse. When the Browse action is enabled, the end user will have the ability to browse through a set of existing related items and attach one to the primary item. This option is enabled by default in a peer relationship with “To one” multiplicity.

After you enable the Browse option, you must also indicate how the user will be able to browse the related items by associating the browse button with a list.

- Clear. When the Clear action is enabled, the end user will have the ability to disconnect (or “clear”) a related item from the primary item. The related item still exists on the system, but the relationship between the two instances will be severed.

- Delete. When the Delete option is enabled, the end user will have the ability to disconnect a related item from the primary item and delete that related item.



You may also set a category for fields in a form. The category attribute is a means for you to group certain fields together. You can then use rules to disable, hide, or show fields in a given category.



Build an order form with relationships

1. Open the Warehouse Application in the Northwind Workspace and expand the Entities folder.
2. Double-click the Order entity to open it in the editor.
3. Click **Add**.
4. Select **Form**.
5. Set the Display Name to **New Order**.
6. Set the Name to **Create**.
7. Set the Description to **Form for new orders**.
8. Click **Add**.
9. In the editor, select the form and click **Configure**.
10. Click **Toggle grid**.
11. Drag the **Id** field and drop it on the presentation.
12. Set the presentation of the **Id** field to **Text only**.
13. Change the Label of the **Id** field to “**Order Number**”.
14. Drag the **Order Date** property and drop it onto the presentation.



To quickly add several properties to a form, click the hyperlinks associated with each property in the Components panel. The properties will be added to the presentation stacked vertically.

15. Drag the **Requested Date** property and drop it onto the presentation.
16. Drag the **Employee** property and drop it onto the presentation.

By default, since the data type of the property is an enumerated integer, the presentation will be set to a radio button option.

17. Change the Type to **Drop list**.
18. Resize and align the fields in the Presentation panel.
19. Under the Other section on the left of the editor, drag a **Tab Container** and drop it next to the fields in the presentation.

20. Select a gray area in the header of the tab container, to the right of the new tab.
It may help if you make the tabbed container larger.
21. On the right, set the number of tabs to 3.
22. Select the first tab.
23. Set the tab's container heading to **Customer Info**.
24. From the Components section on the left, drag the [0..1] **Bill To** relationship and drop it in the tabbed container.



The container will be highlighted in green to show you that the relationship is being added to the interior of the container.

25. Under Presentation on the right side, change the type from **Button(s)** only to **Drop list**.

26. Set the Browse list to **Client List**.

This is the list which will be referenced for contents to the drop list.

27. Set the Property to **Client Code**.

This is the value which will be visible to the user in the drop list.

28. Set Required to true.

29. Further down in the right-hand properties pane, under the Actions section, make sure the Create option is selected.

This option, when enabled, allows users to create ad hoc Customer instances while creating an Order instance.

30. Expand the Bill To relationship on the left and expand Properties.

31. Drag Address, City, Postal Code, and Phone to the Customer Info tab.

32. Set the Read-only property to true on each of these four fields.

33. Select the second tabbed container.

34. Set the tab's container heading to **Shipping Info**.

35. Drag the following Order properties and drop them on the Shipping Info tab:

- a. Ship Via (set the type to **Drop list**)

- b. Freight

- c. Shipped Date (set the Display to **Read Only**)

36. Select the third tabbed container.

37. Set the tab's container heading to **Items**.

You will add the contents to this tab later in this chapter.

Your form could resemble the following:

Figure 5-10:
Sample order form with a “to-one” relationship

The screenshot shows a 'Presentation' window with a toolbar at the top. Below the toolbar, there's a main area containing several input fields and a sub-form. On the left, there are fields for 'Order number', 'Date ordered', 'Requested Date', and 'Employee' (with 'N/A' selected). To the right of these is a sub-form titled 'Customer Info' which contains fields for 'Client Code', 'Address', 'City', 'Postal Code', and 'Phone'. Above the sub-form are tabs for 'Customer Info', 'Shipping Info', and 'Items', with 'Customer Info' being the active tab. A '+' button is located next to the 'Client Code' field.



You can also use keyboard shortcuts to move or resize fields in a presentation. The arrow keys will move the selected field 5 pixels in a given direction. CTRL + arrow key will move the selected field just one pixel. Shift + arrow key will increase or decrease the width or height of a field, respectively. ALT + arrow key will align sets of selected components to the left, right, top, or bottom, respectively.

38. Save and close the form.
39. Save and close the Order entity.
40. Validate the Warehouse Application.

Sub-forms

In many cases, because an entity will have many properties, the forms which you build will have groups of fields which will often be repetitive. For example, in the create Order form which you just built, the customer information can be grouped into its own section and repeated across many different forms. So too can the Shipping Info information. It can be an arduous task to build several different forms with the same fields over and again, so a more elegant strategy is to create a *sub-form*.

A sub-form, or “nested form”, is created with the form building block, but an entity modeler can simply assemble sub-forms together to create new forms. In this way, if data for a form ever changes, you can make the presentation in just one place - the sub-form - and the changes will be propagated to all the forms which use that sub-form.

As you may recall, the Business and Person entities are sub-classed from Customer. When you work with abstract entities, however, you cannot create items: because they are abstract, they can never be realized as instances. Therefore, if you build a ‘Create’ form for an abstract entity, it will not be displayed in Process Experience.

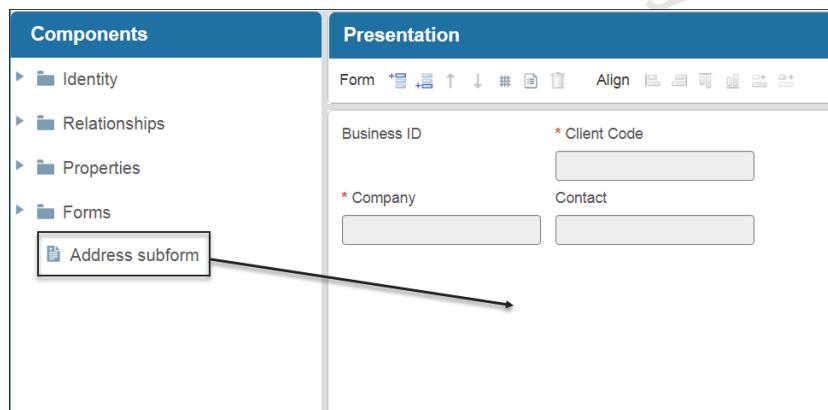
In terms of inheritance, subclasses borrow building blocks from the parent, including properties and forms. In the example below, you will create a sub-form in Customer which will contain just address information. Then you will add that sub-form to your Business entity. Because Person is also subclassed from Customer, the same sub-form can be used for Person, as well.



Build and use a sub-form for customers

1. Open the Warehouse Application if it is not already open.
2. Open the Customer entity.
3. Click **Add**.
4. Select **Form**.
5. Set the Display Name to **Address subform**.
6. Set the Name to **subformAddress**.
7. Set the Description to **Sub-form containing address info**.
8. Click **Add**.
9. Select the sub-form from the entity editor and click **Configure**.
10. Under the Components section on the left, expand **Properties**.
11. Drag the Address, City, Region, Country, and Postal Code properties and drop them onto the canvas.
12. Change the display name (i.e., label) of the Address field to **Street** in order to prevent any ambiguity.
13. Resize and align the fields on the canvas appropriately.
14. Save and close the sub-form. Save the Customer entity if required.
15. Open the Business entity.
16. In the entity editor, select the Create form and click **Configure**.
17. On the left side, under Components, scroll to the bottom and drag the Address sub-form into the editor.

Figure 5-11:
Add the Address
subform to the editor



18. Clear the **Show container heading** option.

19. Clear the **Default border** option.
20. Set the border style to **No border**.



Subforms are containers with their own headings and borders. You can choose to show or hide the heading, give the heading a different name, or change the border. There are many different styles of border available (including dashed, double-lined, inset, etc.), widths of the border wall, and colors.

21. Save and close the Create form.
22. Save and close the Business entity.
23. Validate your project.
24. Publish your project to the organization.
25. Open a browser window and navigate to Process Experience.
26. Log in as user analyst@training.local if you need to.
27. Click the Create (+) button in the upper right.
28. Select **Business**.

The Create form that you prepared for the Business entity should be displayed in a separate window.

29. Complete the form with some random values and click Create.



When you create an item in Process Experience, you also have the ability to open the new item (upon creation) or save the item and create another of the same type of item.

30. On the Process Experience home page, select Customer > Client List from the left.

Your new customer should be displayed in the Results panel.

31. Click Create (+) and select Order to create an order.
32. Set the date ordered and employee name.
33. On the Customer Info tab, use the drop-down to select the customer you just created.

If you added address values for your new customer item, they should be automatically populated into the read-only fields of the form.

34. On the Shipping Info tab, set some values for Ship Via and a cost for freight.
35. Click Create to finish building the new order. You do not need any values for the items yet.

36. Select the Order list. Does your new order appear in the list?
37. Click the new order or, alternatively, select it and click Open. Are you able to edit the order? Discuss.



For some fields, you may choose to restrict the options available depending upon the values of other fields. For example, you may want to restrict the states or provinces available based upon the country you have selected, or the cities based upon the state or province. There are two presentation types with a relationship you can use on your form to achieve this:

- you can choose a drop-list series as the presentation type (instead of a grid or repeating group series). A drop-list series, as the name suggests, is a series of drop-lists which allow the user to drill down on choices by choosing values from successive drop-lists.
- for other types, like drop list and text box, you can filter a browse list. Click the filter icon next to the browse list and use expression language to filter the list contents.



Build a sub-form for Order with grid container

1. Return to the Process Platform collaborative workspace.
2. Open the Order entity from the Warehouse Application.
3. Click **Add > Form**.
4. Set the display name to **Order Items Grid subform** and the name to **subformOrderItemsGrid**.
5. Set the Description to **Order Items sub-form with grid presentation**.
6. Click **Add**, then click **Configure**.
7. Under the Relationships section in the Components panel, drag the **[1..*] Order Line** relationship to the editor.

Since Order Line is in a child relationship with Order, Process Platform will automatically create a multi-item group container. By default, Process Platform uses a repeating group container.

8. Clear the **Show container heading** setting.
9. Change the type of container to **Grid**.
10. Under the Available actions section on the right, make sure the Open, Delete, and Create options are enabled for the container.
11. Under the Order Line item in the Components panel, locate the Relationships section.

These represent nested relationships.

12. *Drag the [0..1] Product Ordered relationship and drop it inside the Order Line grid container.*

You must drop the Product Ordered relationship inside the Order Line repeating group because there could be many products ordered, but only one product per line.

The relationship is added as a column to the grid.

13. *For the Product Ordered relationship column, set the Browse list to Product Catalog.*
14. *Set the Property to Name.*
15. *Make sure that Browse and Clear are the only available actions which are enabled.*

End users should be able to select or remove a product for the Order, but should not be able to create new products during an order.

16. *Drag the following properties and drop them inside the Order Line grid container:*
 - a. Order Line > Price (set to Read-only)
 - b. Order Line > Quantity
 - c. Order Line > Discount
17. *Resize and align the fields on the form to create an organized, ordered presentation. You can drag columns in the grid container to rearrange them.*
18. *Save and close the sub-form.*
19. *Open the Create form for order.*
20. *Click Configure.*
21. *Switch to the Items tab in the order form.*
22. *Drag the Order Items Grid subform and drop it into the Items tab.*
23. *Clear Show container heading.*
24. *Clear Default border and set the border type to No border.*
25. *Resize the tabbed panel container as required to make sure that scrolling is not necessary.*
26. *Save and close the Create form.*
27. *Save and close the Order entity.*
28. *Validate and publish your solution.*
29. *Switch to Process Experience. Refresh the browser to collect your published changes.*
30. *Click the '+' button and create an order.*
31. *Set the date ordered, employee, and client fields. Remember that you can create an ad hoc customer instance with the customer create button.*
32. *Select the Items tab.*

33. Click ‘+’ to add a new row to the grid.
34. Select the first column and click the drop-down arrow.
35. Click Browse.

The Product Catalog opens in a separate dialog.

You have not yet created a Create form for the Product entity, so there are no items for you to select.

36. Click Cancel.
37. Click Create: you will add items later.

Layouts

In Process Experience, Create forms are displayed in their own dialog. Once the item has been created, however, you will need to work with it, either to view, modify, or delete information. Layouts provide a means for working with instance data directly in Process Experience once it has been created. Furthermore, layouts provide a means for the Process Experience user to work with other aspects of your solution, such as other applications or tools, as well as Process Experience panels. As a general rule, users will create an item with a ‘Create’ form, otherwise, they will work with your entity through a layout.

Layouts provide you with a means for creating customized user interfaces on Process Experience. Process Experience users also have limited ability to personalize their layouts, by choosing the result columns displayed or by resizing panels, for example. A layout organizes panels (which include forms) into a single Process Experience page. There are two types of layouts:

- Item layouts: Item layouts work with the instance of a specific entity, much in the same way a form does. The item layout may contain a number of panels which display various aspects of an item.

You must create at least one item layout that includes a form (in a panel) in order to make the entity instance visible and/or editable when opening an item in Process Experience. You may create different layouts (with different forms) to represent the item in different stages of its lifecycle. The form you use in the item layout can be any form of your design.



The item data which is available for edit is often different from the data used to create the item. Therefore, it is recommended that the form you use to edit an item in a layout is different from the Create form.

- Homepage layouts: a Homepage layout provides a single page for users to launch applications or tools in Process Experience. Process Experience includes a default home page, and an identity home page for working with the identity package. Using the homepage layout, you may add more home pages. From the home page, Process Experience users can launch other applications. A user can switch between any homepage layout they want, as long as they have the authority to use it. Homepage layouts are not related to any specific entity instance and, as such, are not created in the entity designer window.

A layout can include any number of panels that you arrange on a single Process Experience page. These panels can include pre-designed forms and lists. If you combine multiple panels into a single area, or zone, the user interface will automatically present tabs to the user as a means of selecting which panel to use. Layouts are designed to present users with the information they need in a way that is optimized for the interactions they are expected to perform. Many applications do not include a homepage layout, relying on the system-provided default layout to serve their needs. Similarly, many entities rely on the default item layout you create. Typically, however, you could build a few basic homepage layouts, and one item layout for each entity: the layouts will be optimized for a different task or type of user.

For example, consider the warehouse application you have been building. In this example, a customer service representative (or CSR) may receive the order and enter the default details. The warehouse, which is fulfilling the order, may need a completely different presentation of this data in order to fulfill the order. In this example, the two different roles - CSR and warehouse fulfillment - dictate the type of layout each user will require. The implication, too, is that the item exists at different stages in its lifecycle - Creation and Fulfillment - which necessitate different layouts. A senior CSR, which is handling customer complaints, may require a complete end-to-end view of the entire order, from handling to fulfillment, in order to address any customer complaints or lost shipments. This would require a different layout.



You can restrict who has access to the data, and whether it is visible on the form, using rules and security building blocks, which you will learn in the following chapter.

Building a layout Like a form, building a layout is performed in two stages: providing basic information and then configuring the presentation. The basic information includes a display name, a unique name, and a description. You also have the following options when adding basic information for a layout:

- Preview: in preview mode, you can choose to display the layout in a preview panel.
- Full: in full layout mode, you can display the entire layout in a separate page.
- Use to create new items: in this mode, users are able to create new items from within the layout. When this box is checked, the layout will appear as an option under the Process Experience Create (+) menu.



You can use a layout with 'Use to create new items' enabled as an alternative to a form with the Create name or as a second method to create an instance.

- Item URL: each layout will be given a unique URL so that you may include the layout in other applications, such as portlets, when working with items.



Build an item layout with basic information

1. From the Warehouse Application in the Northwind Workspace, expand the Entities folder and double-click the Business entity to open it in the editor.
2. Click **Add**.
3. Select **Layout**.
4. Set the Display Name to **Business Contact**.
5. Set the Name to **workWithBusiness**.
6. Set the Description to **Default layout for working with business entities**.
7. Set **Full Layout** to true.
8. Click **Add**.



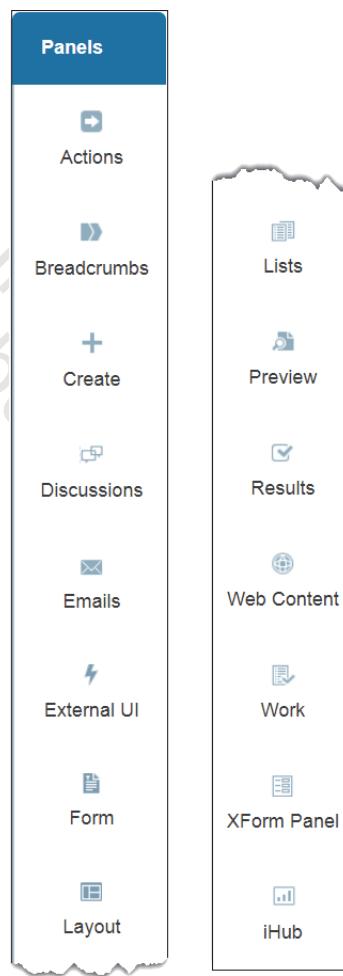
Remember, in order for you to view or edit items, you will typically require at least one layout for each entity you create.

Building the presentation of a layout

Once the basic information has been added, you can set the presentation of the layout, as was the case with forms. Similarly, there is a presentation aspect for layouts, just as was the case with forms. Unlike forms, however, the presentation of a layout is performed on the desktop of Process Experience. Furthermore, there are many more panels which can be added to a layout that could not be added to a form. Click the Configure button from the entity editor to configure the layout presentation.

The layout designer is divided into two sections: the palette of panels on the left, and the desktop in the editor (with attributes on the far right). You can drag panels from the palette and drop them onto the desktop to create your layout. There are many types of panels which you can add:

Figure 5-12:
Palette of panels for a layout



Each panel which you add to the layout has some common and some unique attributes. Some of the common attributes which panels share include:

- Name: the title of the panel which will be displayed to the Process Experience users.
- Description: an arbitrary description to describe the purpose and function of the panel in the layout.
- Chrome: basic instructions on how to display the decorative dividers between panels, such as margins, borders, and title bars. Chrome options include:
 - Full: include the header and any borders on a plain background. If you use full chrome, you can also include a custom icon for the panel.
 - Container only: display only the container.
 - Title only: display only a title in the panel.
 - None: no header or borders are displayed, and the background is transparent.

Furthermore, depending upon the type of panel you add to the layout, there may be other properties which you may need to set. For example:

- In the layout panel, you must select the layout you wish to include.
- In the results panel, you must select whether you want to run or link to a list. With the run option, the results of a selected list will be displayed. With the link option, a list is selected and then its results are displayed.
- In the form panel, you must select the form to display.

There are several different types of panels which you may add to a layout. These panels are:

Actions panel	The actions panel relates to the actions the end user can perform on an item, contained on an action bar. These may be create, delete, modify, or any other action. The actions are added to the panel as buttons and menu items from an action bar building block. When the end user selects the button/menu, the action is executed. The actions panel is only available for an item layout.
----------------------	--



Some actions may require additional parameters. If this is the case, the user will be prompted to enter the parameters, so that they may be passed to the action before it is invoked.

Activity flow summary panel	Use the Activity flow summary panel to select the activity flow tasks required to process an item such as a claim, order, or service request in your application so that an application user can view the activity flow properties for different action performed on the activity flow tasks.
------------------------------------	---

It is available only when the entity contains an activity flow building block.

You can select any of the following properties to show as columns in the activity flow list in the task panel in your application:

- Acted by: user who most recently worked on the activity flow.
- Acted on: date when the activity flow was most recently worked on.
- Category: category of the activity flow.
- Completed on: date when the activity flow was completed.
- Description: description of the activity flow.
- Display name: name that is displayed to users of the activity flow.
- Initiated on: date when the activity flow was initiated.
- Progress: progress of the activity flow.
- Started by: user who started the activity flow.
- Status: status of the activity flow.

The Display Name, Initiated on, Progress, and Status properties are selected by default but you can select other properties as needed. The Display Name property is required and is disabled so that you cannot clear it.

The Task list presents tasks from the activity flow model. When an application user clicks an initiated activity flow, the activity flow details are displayed in the Activity flow summary layout configured in the activity flow model.

Breadcrumbs panel	The breadcrumbs panel provides a trail back to the page from where the end user started. Breadcrumb functionality is provided as a panel so that you can choose whether to use it for easy navigation or choose an alternative direction for the end user.
Business workspace panel	Use the Business Workspace panel to display the folder browser widget that the Content Server exposes. Using the folder browser widget users of your application can perform all the document management operations that are supported by Content Server.
Contents panel	Use the contents panel to display the content list in your application. Content can be files which are attached to an item. That content can be stored in Content Server. You need to add the Content building block to an entity.
	It is recommended that you also include a form panel so that users can see the item, its attachments, and a preview of a selected item.
Create panel	The create panel enables users to quickly create items.
Discussions panel	Discussions may be added as a building block to entities so that teams may collaborate effectively while working with the item. Discussions contain topics on a message board. Users will read and create messages for the message board from the discussions panel. If you include a discussion building block with an entity, at least one of your layouts must include the discussions panel in order for users to be able to use discussions with the item.

Emails panel Use the emails panel to provide email functionality in your applications. This enables users of your application to send, receive, reply to, and forward emails (and attachments) from an item.

This panel is not available for a home page layout.

External UI panel The external UI panel displays data from an external repository. This could include the Process Platform inbox, for working with process tasks, or another portal of your design.

Form panel A form panel displays the properties of an item, usually for user input. First, you need to create forms as building blocks for entities. In the form panel's details, the name option must be set to the name of the form to be displayed.

iHub panel The iHub panel displays the list of reports on the connected iHub instance. You may select a report to be displayed on the layout. In order to connect to the correct iHub instance, the iHub URL must be configured in Process Platform through the iHub Connection Manager.

Layout panel A layout panel includes other layout building blocks. Only layouts that were added to the project with a type of layout panel can be added to a layout panel.

Lifecycle progress panel Similar to an activity flow progress panel, the lifecycle progress panel provides a high level view of an item's progress so a user can view the completed, current, and future states of an item. If users deviate from a primary path, they can also view the alternate path from the deviation point.

To use this feature, you must configure one transition for each lifecycle state as a primary transition.

Lists panel The lists panel displays the set of lists that the logged on user has permission to use. This panel assumes that the layout also includes a results panel that displays the results of the selected list. It also assumes that the results panel has the link to list panel option enabled. On the lists panel, the user can reorganize the set of lists, create new sections, drag lists between sections, hide, and rename lists.

Preview panel The preview panel is used to preview an item inside a page. The preview panel responds to the selection of an item in another panel, presenting that item's entity preview layout. For example, a preview panel in an application layout that also contains a results panel displays any item selected in the results panel. The preview panel can display many items at the same time, organizing them using tabs.

Results panel The results panel displays the results of a list, which is typically a list of items. Results panels provide the main functionality of the standard user home page. When you add the results panel to a layout you can choose to have the panel display the results of a fixed, specified list or to respond to the selection of a lists panel elsewhere in the layout. Results panels are often used in item layouts embedded in tab containers tied to predefined lists.



The following options are available to a user in the Results panel:

- The filter panel can be "pulled out" from the left edge of the panel to specify filters to reduce the number of items presented in the results.
 - The current set of filters is presented above the results.
 - The menu provides access to functions to customize the columns presented in the results.
 - The actions bar provides the ability to invoke actions on items selected in the results.
 - Pagination controls provide the ability to page through large result sets.
-

The results panel provides extensive personalization features. When a user returns to a layout that includes a results panel that is not tied to a specific list, it automatically displays the results for the last list used. The results panel remembers how the user last configured each list's columns and the filters they used. The results panel also attempts to remember which items the user most recently selected.

If the list is defined with a limit to the number of items to be displayed and there are more items available than can be displayed, the footer in the results panel provides the ability to access multiple "pages" of result items. The following options are provided in the footer:

- Results per page - Enables the user to choose the number of results they wish to display in each page (up to the limit specified by the underlying object).
- Item range - Indicates the index of the items displayed in the current result page. For example, 1 - 20 indicates that the first 20 available items are being displayed.
- Total count - Indicates the total available number of items. This is only included if the view is configured to display this total.
- First - Displays the first page of result items.
- Previous - Displays the previous page of result items.

- Page - Indicates which page of result items is being displayed. This can be changed to skip to any page. Entering a number less than 1 displays page 1. If the total count is not displayed, a user can enter a number that is past the last available page, in which case a blank result page is displayed.
- Next - Displays the next page of result items.
- Last - Displays the last available page of result items.

Tasks panel Use the tasks panel to select that tasks that must be performed to process an item such as a claim, order, or service request in your application. It is available only when the lifecycle building block is added to the entity.

You can select any of the following properties to be shown as columns in the task list in your application:

- Status: current state of the task
- Subject: task subject
- Assignee: user or role responsible for the task
- Priority: set between 0 (low) and 5 (high)
- Due date
- Delivery date: date when task was created
- Start date: when task was started

Web Content panel The web content panel is used to embed almost any web page in a layout. The design options for this panel specify the URL of the web page to be displayed. You can specify an exact URL to be shown when the panel opens. For example, if a user's task is to look for newspaper articles that contain certain information, you might configure the URL as:

`http://www.nytimes.com/`

to go directly to that page.

Alternatively, you can specify the URL using variable substitution syntax that enables you to include properties in the URL. These can be any of the system configuration properties. This is typically used to specify the host name in the URL so it can be changed without having to change the project's definition.

In an item layout, you can use any of the entity's properties to embed web pages from existing applications that take parameters in their URLs. For example, assume that an existing customer management application takes a URL with a CustomerId parameter to return a web page with customer information. You can include a web link panel in the layout for a loan application that specifies a URL of:

`http://MyCustomers.com/
CustomerDetails?Id={item.Properties.CustomerID}`

using the CustomerId property from the item entity to provide parameters for the URL.

When the web content panel appears on an item layout, all of the properties of the current item are available to be used to generate a customized URL that is specific to the entity being viewed. You do this by using variable substitution that includes values from the entity type in the URL. For example:

```
https://maps.google.com/  
maps?q={item.properties.ZipCode}&output=embed
```

This dynamically constructed URL displays a map of the zip code that appears in the ZipCode property of the object type's property group (named Properties). All such variable name references are case insensitive, but other parts of the URL and the values substituted for the named variables are as they were entered. The way in which the values are interpreted by the destination website is up to that site.

While the variables used in the URL would typically be properties in an entity (as in the previous example), any property in an entity can be used in the construction of the URL.

Examples of the types of properties that can be used are:

- Item property: {item.<Element name>.<Property name>}
- Item ID: {item.id}
- Current user's name and user ID: You can access the user's user id by specifying {user.Properties.UserId} or just {user.UserId} and you can access the user's full display Name by specifying {user.Properties.FullName} or just {user.FullName}. Variables referencing user properties are not limited to use in panels that appear on object layouts. They can be equally useful on home page layouts.
- Solution variables: {config.variable} where variable is expected to be defined as a solution variable within the containing solution using the Process Experience Administration Tool.
- System properties: system level properties that are available for use in applications. Typically, the value of these properties depends on the environment where the application is installed. The following system properties are available:
 - {system.baseURL} is the beginning URL portion for references to other OpenText applications running on the same server in the same organization.
 - {system.organization} is a reference to the current user's organization.

If a variable cannot be resolved (if it was mistyped, is unavailable, or does not exist), the value is not substituted and the variable reference is displayed as entered (enclosed in {}).

If the property value is empty, the parameter is substituted with null. Assume that a user creates items using the appraisal object type. Each time the value of the city property changes, the web content panel displays a map of that city. The following example shows how this is specified:

```
https://maps.google.com/  
maps?q={item.properties.CityP}&output=embed
```

If the destination website does not provide meaningful feedback on malformed URLs, there are URL echo sites that can help to determine the resulting URL was by echoing it back to be displayed in the panel.

The capability to generate a customized URL that is specific to the object depends on the external system the user is connected to.

Work panel The work panel displays a summary of the lists that the user has permission to use.

XForm panel The XForm panel displays a list from which you can select a Process Platform user interface (written in XForm) to open in the panel. The Process Platform XForm must be developed separately, either in Process Platform or in another development environment.



You will build a Process Platform user interface with XForms in course 4-4913: Process Modeling for Process Platform.



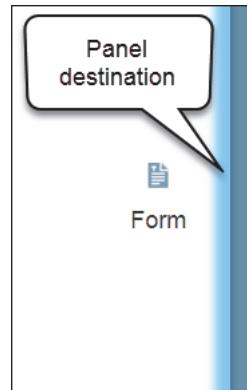
Build the desktop for an item layout

1. Open the Business entity from the Warehouse Application, if it is not open already.
2. Select the *workWithBusiness* layout.
3. On the right, click **Configure**.
4. Drag the Results panel from the set of panels and drop it onto the Desktop editor.
5. Under the General properties section, set the name to **Clients** and the Chrome to **Title only**.
6. On the right side, under Panel Properties at the bottom of the attribute list, select **Run list**.
7. From the drop-down of lists, select **Client List**.

8. Drag a Form panel from the panels palette and drop it to the right of the results panel on the Desktop editor.

As you drag the panel to position it, you should notice a blue bar which provides hints as to where the panel will be placed when it is dropped.

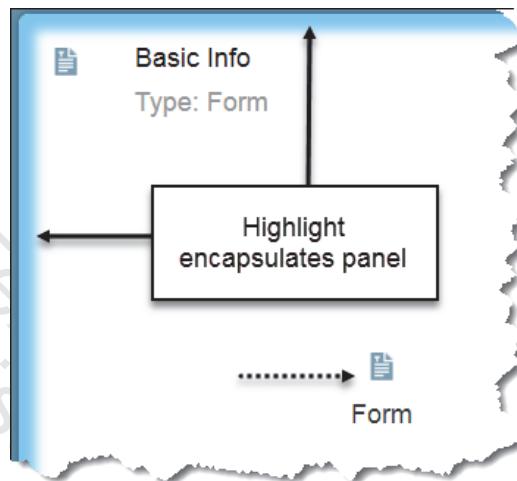
Figure 5-13:
Destination of panel
when dragged to desktop



9. Under the General section on the right, set the name to **Client Info** and the Chrome to **Full**.
10. At the bottom of the Properties section on the right, and with the form panel selected, select the Panel Properties.
11. Under the form drop-down, select **Create Business Customer**.
12. Drag an Actions panel and drop it above the Results and Form panels.
13. Set the Panel properties of the actions panel to the **Default** action bar.
14. Grab the handle separating the actions panel from the other two and resize it so the actions panel takes less desktop real estate.
15. Resize the results and form panels so the form panel has more real estate than the results panel (about 15%).
16. Save and close the layout.
17. Close the Business entity.
18. Validate the Warehouse application and publish it to the organization.
19. Open Process Experience in another window.
20. Refresh the browser.
21. From the Process Experience home page, select the Customer > Client List from the Lists panel.
22. Open your new client from the Results panel to confirm that the layout is displayed properly.

Building a tabbed panel Creating sub-forms which encapsulate small pieces of functionality is an efficient method to accomplish UI development. Sub-forms can be easily edited as business needs change, and those changes are propagated to wherever the sub-forms are used. Instead of creating a form which contains several sub-forms, you may choose to add several sub-forms to a single layout. By stacking form panels on top of one another, the Process Experience UI engine arranges stacked panels into tabs. The entire panel will be highlighted to indicate that the panels will be layered into tabs.

Figure 5-14:
Creating tabbed panel in
layout editor



When creating a tabbed panel, the name of the panel becomes the label of the tab. It is therefore a good idea to set the Name property of each panel in the layer.



It may be preferable to create the appearance of tabbed panels by layering forms in a layout instead of using a tabbed container in a form. Stacking forms in a layout to create tabbed panels provides you with greater design flexibility and reusability.



Stack sub-forms to create a tabbed panel in a layout

1. Open the Person entity in the Warehouse Application of your collaborative workspace.
2. Click **Add > Form**.
3. Set the Display Name to **Create Person**.
4. Set the Name to **Create**.
5. Click **Add**.
6. Click **Configure**.
7. Drag the following fields onto the form: *Id*, *Client Code*, *First Name* and *Last Name*.

8. Set *Id* to be *Text Only* and both *Client Code* and *Last Name* to be required.
9. Save and close the form.
10. Click **Add > Layout**.
11. Set the Display Name to **Edit Person** and Name to **workWithPerson**.
12. Set **Full Layout** to true.
13. Click **Add**.
14. Click **Configure**.
15. Drag a Results panel to the desktop editor.
16. Set the name of the Results pane to **Clients**.
17. Set the Chrome to **Title only**.
18. Set the panel properties to **Run list** and the list to **Client List**.
19. Drag a Form panel and place it to the right of the results panel.
20. Set the name of the form panel to **Personal Info** and Chrome to **Full**.
21. In the panel properties section, set the Form to **Create Person**.
22. Drag another form panel and layer it on top of the personal info form.

You can tell if the forms are stacked if the panel is completely outlined in blue.

23. Set the name of the new form to **Address Info**.

Layered panels are always full chrome.

24. Set the form in the panel properties to **Address subform**.

This form was inherited from the parent entity: Customer.

25. Save and close the layout.
26. Save and close the entity.
27. Validate and publish the application.
28. Switch to Process Experience and refresh the browser to capture the published changes.
29. Add a new Person item.

Figure 5-15:
Create a Person item

Create Person	
Id	* Client Code
16,386	GSMIL
First name	* Last name
Guy	Smiley

30. Select the Client List on the home page.

31. Locate the new person item you created and open it in the editor.

The two forms which you stacked in the layout create tabbed panes.

Figure 5-16:
Customer layout as
tabbed panels with sub-
forms



A full layout can be assigned to a list so that, when the list is selected, the corresponding layout will be used for items in the list. If you build a list and do not add a layout to it, then the first full layout you build will automatically be associated to the matching entity's list. In this way, you can use a different layout for each list in Process Experience.

Layouts with flow objects

In an upcoming chapter, you will learn about the Lifecycle and Activity Flow building blocks. Both of these building blocks are used to capture an item during different stages (or states, or phases). For example, when a new problem issue is received in IT Support, basic information is captured, such as the complainant and the nature of the issue. Once the basic information is captured, a support specialist may contact the complainant and gather more information, including the make and model of the device, operating system, and so forth. As the issue progresses, support specialists may confer, attempt various solutions, schedule support visits, and so forth, until a solution is discovered and the issue closed. This example describes a support ticket in various stages of its lifecycle: Open, Triage, In Progress, and Resolution.

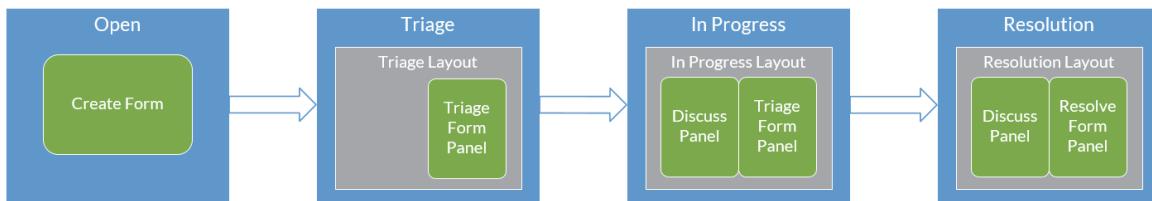


Figure 5-17: Sample IT support lifecycle with layouts

At each stage of a flow, different sets of information may be required which are specific to the actor. At the onset, when the support ticket is created, only enough information is required to create the ticket. In this case, a Create form can be used. During the other stages, however, different types of information may be required. Triage, for example, requires more information than Create. In Progress, on the other hand, may require collaboration between support specialists, which suggests employing the Discussions building block with its corresponding layout panel.

It is a common strategy to use a layout for each stage of such a flow, where the layout reflects the data which is required in order to effectively complete the stage and transition to the next stage of the lifecycle.



You will build a Lifecycle and Activity Flow in an upcoming chapter.

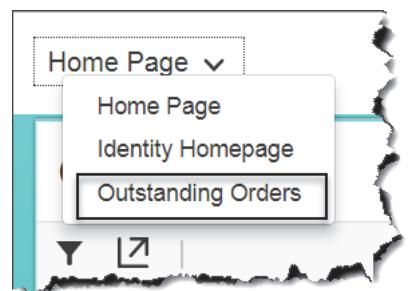
Home page layout

A home page layout is a little different from an item layout. A home page layout may include other applications and utilities other than those related to a specific entity. For this reason, a home page layout is a different document which needs to be created in the workspace, not in the entity.

Whereas item layouts are built with the layout building block in the entity editor, home page layouts are built with the home page layout document in the workspace.

Each home page layout document has the same basic information: a display name, a name, and a description. The home page layout also has the same Desktop editor. Not all panels are available for home page layouts as for item layouts. However, when the solution is published to Process Experience, a home page layout will be accessible through the Home Page button on the Process Experience desktop.

Figure 5-18:
**Home Page button on
Process Experience
desktop with new home
page layout**





Build a home page layout for outstanding orders

1. On the Process Platform user start page, select the Northwind Workspace.
2. Right-click the Themes folder and select **Rename**.
3. Set the name of the folder to **Themes and Layouts**.
4. Select the folder, right-click it and select **New > Other**.
5. Select **Home Page Layout** from the list of available documents.
To filter through the documents easier, start typing the name of the document in the filter field above.
6. In the new home page layout document, set the Display Name to **Outstanding Orders**.
7. Set the Name to **hpOutstandingOrders**.
8. Set the Layout Type to **Home Page**.
9. Save the document.
10. Click **Configure**.

The layout editor appears.

11. Add the following panels to the layout desktop:
 - a. Lists: set Chrome to **Full**
 - b. Results: set name to **Outstanding Orders** and Chrome to **Full**
 - c. Create: set Chrome to **Full**

You may arrange the panels as you choose. The following screen capture is provided as an example of a possible arrangement of panels:

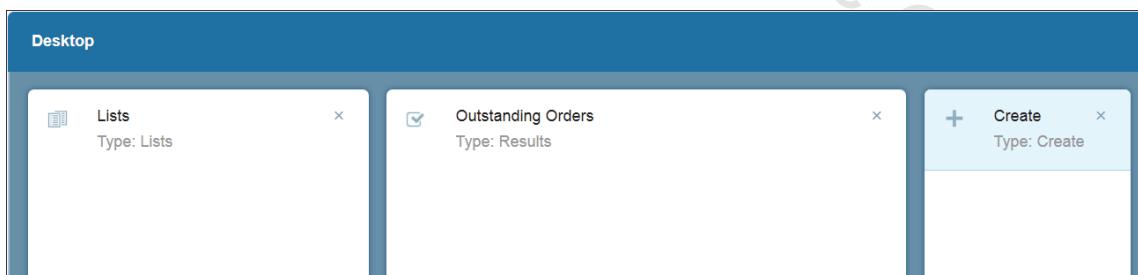


Figure 5-19: Sample desktop for Outstanding Orders home page layout

12. In the Outstanding Orders results panel, under **Panel Properties**, select **Run list**.
13. Set the list to **Outstanding Orders**.
14. Resize the panels to fit comfortably in the desktop.
15. Save and close the home page layout.
16. Save and close the home page layout document.
17. Validate your project and publish it to the organization.

18. Open Process Experience (or refresh the Process Experience page if it is already open).
19. Click Home Page in the top left and confirm that the Outstanding Orders homepage layout is available.
20. Click Outstanding Orders and confirm the page layout is displayed correctly.

Discussion building block

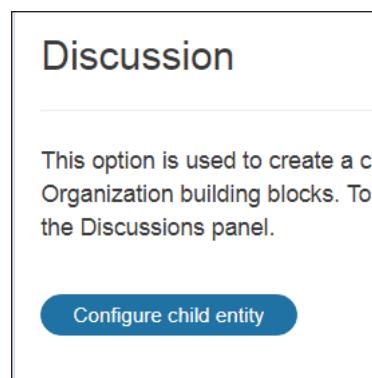
As previously mentioned, the Discussion building block is used for team members to share messages and collaborate on items. Messages are passed between team members through a message board. The Discussion building block adds the message board to the entity instance, and thereby allows Process Experience users to team up. The Discussion building block can only be added once to the entity: once it has been added, its presence is removed from the list of building blocks. After you add the Discussion building block to an entity, you must create a layout that includes the Discussion panel in order to take advantage of the features of Discussion.



Discussion building blocks cannot be added to child entities.

When a Discussion building block is added to an entity, it is added as a special kind of child entity object. The Discussion message board contains a pre-defined identity, with pre-defined properties: topics and replies. Unlike other building blocks, Discussion does not have a display name or editable name attribute. Because it is structurally similar to a child entity, the configuration button for working with Discussions is labeled **Configure child entity**.

Figure 5-20:
Details of Discussion building block



Configuring discussions Besides the identity building block, discussions include two properties: DiscussionNote and DisplayOrganization. The DiscussionNote property is the body of the message itself, and allows comments and replies to be added to the message board. The structure and appearance of these messages on the board (grouped in topics and replies) is handled by the DisplayOrganization property. Neither of these properties may be modified.



Discussion is treated as a child entity, and therefore, you are able to add other building blocks to it. However, this practice is not recommended.



Do not delete any of the properties of a discussion.

The Discussion building block contains an identity and two pre-determined properties: DiscussionNote and DisplayOrganization. Since these are treated as properties of a child entity, the delete button is enabled. Do not delete any of these features: if so, the Discussion building block will not function properly.



Add a discussion with layout to the business entity

Occasionally, your customer service team may want to include and maintain notes about business customers in the company portfolio. A Discussion building block will capture this aspect of the customer service application.

1. Open the Business entity from the Warehouse Application.
2. Click **Add**.
3. Select **Discussion**.
4. Click **Add**.



In order to take advantage of the features of the Discussion building block, you must build it into a layout that uses the Discussion panel.

5. Select the **workWithBusiness** layout and click **Configure**.
6. Drag a Discussion panel to the far right of the Desktop presentation.

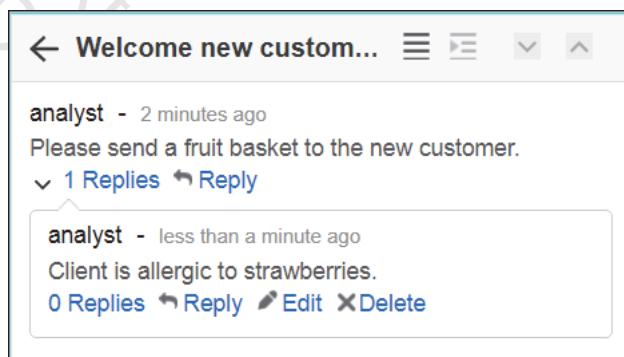


If you add a Discussion panel without adding a Discussion building block, you will generate a REST error in Process Experience.

7. Set the width of the discussion panel to approximately 30%.
8. Save and close the layout.
9. Save and close the entity.
10. Validate and publish your Warehouse Application.
11. On the Process Experience page, refresh the browser to accept the changes.
12. From the Process Experience home page, select the Customer > Client List.
13. Open a business customer in the new layout. You should have at least one business customer. If you don't, create one first.
14. In the discussion panel on the far right, click '+' to begin a new conversation.
15. Add a title and message to your discussion entry.

Users have the ability to create discussion topics, edit or delete their own comments, or reply to existing topics.

Figure 5-21:
Discussion and reply in discussion panel



Summary

Having completed this chapter, you should be able to:

- Create a form for an entity
- Explain layout panels and their composition
- Create a layout for an entity
- Create a home page layout
- Create a custom home page for Process Experience
- Add a discussion building block to an entity

Exercises

1. Build sub-forms for supporting new forms and layouts:
 - subformContact: contains phone and fax information for Customer entity.
 - subformProduct: contains basic information (i.e., ID, name, category, price) for Product entity. Consider making changing the label of ID, making it read-only, and changing the presentation of the category property.
 - subformStock: contains only stock information (i.e., on hand, on order, reorder level, shipping units, discontinued) for Product entity.
 - subformPerishable: contains basic information (i.e., keep refrigerated, temperature, best before date) for PerishableProduct entity.
 - subformOrder: contains basic information (i.e., ID, order date, requested date, employee) for Order entity.
 - subformShipping: contains shipping information (i.e., ship via, shipped date, freight) for Order entity.
 - subformClient: contains read-only client information for Order entity (e.g., Bill To client relationship, client's address, city, region, country, phone properties, all set to read-only).
 - subformBusiness: contains basic information for Business entity (e.g., ID, client code, company, logo, contact).



Add your contact subform to the Person and Business entity layouts as a tabbed panel with the name: Contact Info.

2. Build Create forms to support the following entities. You may use the following screen captures as examples. Consider using the sub-forms you created in the previous exercise.

Product > Create

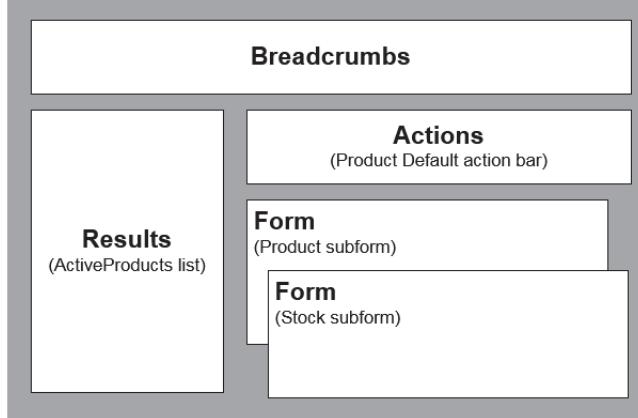
Create Product

Id	Name	
2	Speziales Quark	
Category	Unit cost	
Dairy Products	\$1.99	
Stock Info		
In Stock	On Order	Reorder Level
214	0	20
Shipping Units		
24/case		
<input type="checkbox"/> Discontinued		

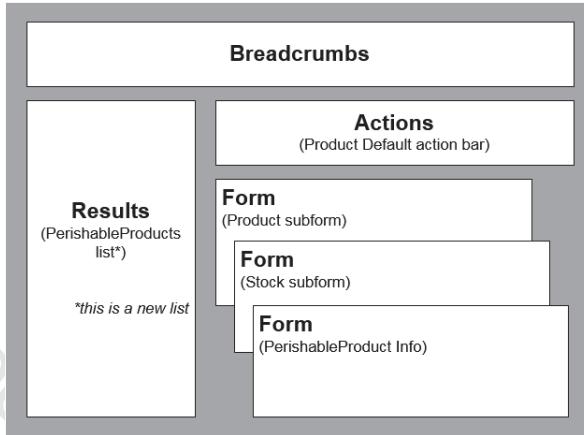
PerishableProduct > Create

The screenshot shows a form titled "Create PerishableProduct". At the top, there are three tabs: "Product Info", "Stock Info", and "Perishability", with "Perishability" being the active tab. Below the tabs, there are two main sections. The first section contains a checkbox labeled "Keep Refrigerated" and a text input field labeled "Temperature" with the value "72.0". The second section contains a label "Best Before" and a date input field with the placeholder "mm/dd/yy".

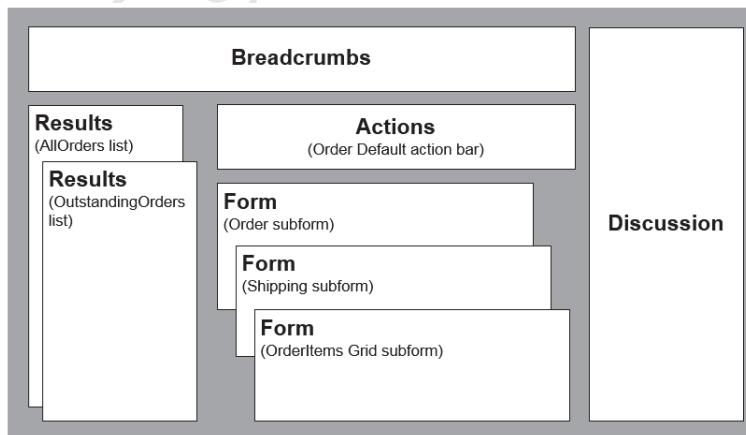
-
- When you build a form for the PerishableProduct entity, which is sub-typed from Product, use the same form name (Create) but make sure you click **Replace Original** in the left margin of the entity editor. There cannot be two forms named Create in the subclass.
-
3. Create full layouts to support the following entities. You may use the following layout plans as your guide. Consider using sub-forms in panel stacks to create tabbed panels.

Product layout (workWithProduct)

PerishableProduct layout (workWithPerishable)



Order layout (workWithOrder)



Consider redesigning the Order/Create form to use your new sub-forms.

You will not be able to test the order layout until you add the Discussion building block, below.

4. Add a Discussion building block to the Order entity. Users should be able to collaborate during order fulfillment.
5. Validate and publish your work to the organization in order to test it in Process Experience.
6. When you are finished, commit your work to the repository under the name: "Chapter 5 - Forms and Layouts".



Tips, tricks, and traps

1. Repeating group container

When adding a ‘to many’ relationship to a form, you may choose to add the contents as a grid container or a repeating group container. You used the grid container with the order items in this chapter. The grid container presents each related item in a table format of rows and columns. The repeating group container presents each related item as individual, repeating sub-forms. You must design the form which will be repeated. The items in a repeating group container occupy more visual real estate, because of the nature of the forms, but are more feature-rich than a table grid.

2. Creating new items from a layout

When designing a layout, there is a checkbox labeled ‘Use to create new items’. If this checkbox is enabled, you may use the layout as a substitute for a ‘Create’ form in order to create new items. When adding a new item, Process Experience will present the layout instead of a Create dialog.

3. Filtering results in a form field

You can filter the results presented in the drop list of a form. When designing your form and adding a drop list, click the ‘filter’ button adjacent to the browse list feature of a drop list.

The screenshot shows a configuration window for a form field. At the top, it says 'Type' with a dropdown menu set to 'Drop list'. Below that is a section labeled 'Browse list' containing a text input field with the value 'Client List' and a small circular icon with a magnifying glass symbol to its right, indicating a filter function.

- To limit the results presented by the drop list, click the filter button.
- Enter an expression (written in expression language) to dynamically limit the results.



You will learn the basics of expression language in the following chapter.

4. Drop list series

A ‘to one’ type of relationship may be added to a form in a number of ways: as a drop list, as a text box, a list box, a simple browse button, or as a drop list series. A drop list series allows the user to ‘drill down’ choices from most generic to most specific. For example, you can use a drop list series to:

- select a country,
- select provinces or states for that country, and
- select a city from that province/state.

Drop list series are presented as a collection of drop lists in a framed container. You must select a list from which the properties will be drawn, and you must add properties from that list for each drop list of the series.

The screenshot shows a user interface for a 'Bill To' section and a configuration panel for a 'Drop list series'. The 'Bill To' section contains three dropdown menus labeled 'Country', 'City', and 'Client Code'. The configuration panel on the right shows a 'Browse list' set to 'Client List', and three properties: 'Property 1' (Country), 'Property 2' (City), and 'Property 3' (Client Code). An 'Add' button is also present.

6. Rules

Objectives

On completion of this chapter, participants should be able to:

- Explain the Rule building block
- Describe rule actions
- Explain the expression language for building rules
- Build a rule for an entity
- Add audit history to an entity

Overview

The Rule building block is provided as a means to add simple business logic to an entity. This business logic could be used on a form, to display or hide fields, or it could be used to cross-reference and double-check data entered. Rules provide a simple, powerful means to introduce logic to an entity. Rules are relatively easy to use, but they can be even more powerful and you can exert more control over the solution with expression language.

Another important aspect of working with entities is maintaining an audit history of your items. The History building block allows you to monitor changes to an item when it is created, and each time it is modified.

What you will build in this chapter

- Rule building block: rules allow you to add programmable business logic to an entity in order to set values, hide fields, trigger buttons in action bars and more.
- History building block: history can be added to an entity in order to maintain an audit trail.

Timing

Lecture: 45-60 minutes

Exercises: 75-90 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform 16.3 Entity Modeling Guide
- Regular expressions (<http://www.regular-expressions.info/>)

Rule building block

The Rule building block is composed of three parts: a trigger, a condition, and an action (“when, why, and what”). Once the trigger is set off, a condition (or set of conditions) is evaluated. If the condition is met, then the related action will be executed. The trigger could refer to a button which is clicked (e.g., on an action bar) or a property which is changed. For example, if the total cost is greater than \$300 when an order is created, a 10% discount will be applied on the total cost. Other examples may be to display a Price Override button on a form if the Process Experience user is listed as a manager when a user asks for an override, or show a warning dialog if customer’s shipping address is different from the address on the contact form.



Currently, you can only execute one action in a rule.

Rules are built in either basic or advanced mode. In basic mode, a simple editor is provided to build conditions and actions. You were introduced to the basic mode earlier in this course, when you built a filter for a List building block. Advanced mode is much more complex and is based upon a rule expression language. Advanced mode requires a good degree of comfort with principles of programming and development. If necessary, you can build rules using basic mode, then switch to advanced mode to manipulate the rules using the rule expression language. Some rules may be too complex to express in basic mode.

When you design a rule in the entity designer, you must first provide basic information for the rule, such as the display name, a unique name, and an optional description. After you have saved this straightforward information, you will use the rule editor to build the rule (i.e., the trigger, conditions, and actions) using either basic or advanced mode.



Add a Rule building block to an entity

1. Open the Warehouse Application in the Northwind Workspace, if it is not open already.
2. Expand the Entities folder and double-click the Order entity to open it in the editor.
3. Click **Add > Rule**.
4. Set the Display Name to **Speedy base price property set**.
5. Set the Name to **propSetSpeedyBasePrice**.
6. Set the Description to **Set Speedy base price for freight cost**.
7. Click **Add**.
8. Save the Order entity.

Editing the logic of a rule

After you have added basic information for a rule, you must provide the rule logic: that is, the trigger, condition(s), and action. Process Platform is equipped with a built-in rule engine, which will verify your rules as they are built. If the rules are incomplete or inaccurate, the rule engine will invalidate them in the entity editor.

Figure 6-1:
Rule invalidated by rule engine

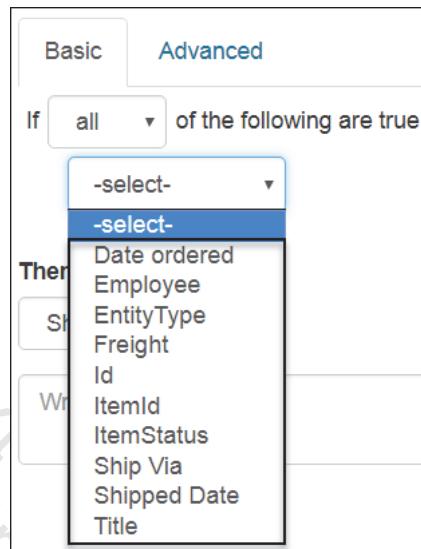


If a rule is invalidated by the engine, you can still validate the application project and workspace, but you cannot publish it to the organization.

Every time you add a Rule building block for an entity, the rule will be in an invalid state until you add a trigger, condition(s), and action.

The rule editor will automatically assemble all the relevant building blocks for the entity, so that you may build conditions and actions using basic mode. The conditions around which you can build your rule include the Identity, Title, and Property building blocks, but not the building blocks of any related entity (i.e., Relationships).

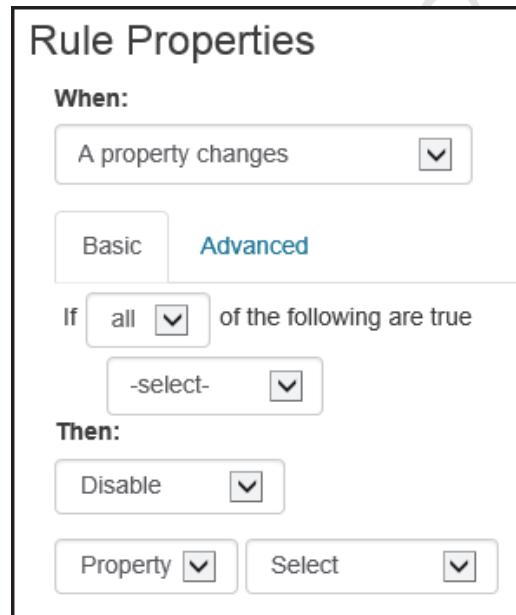
Figure 6-2:
Identity and property
building blocks in rule
editor



Basic mode

The basic mode of the rule editor presents the rule in a format which is intended to be easy to read. If you can express the rule in a simple “when-if-then” or just an “if-then” statement, then you should be able to build the rule using basic mode. The basic mode editor presents itself in this “when-if-then” format:

Figure 6-3:
Basic mode in rule editor



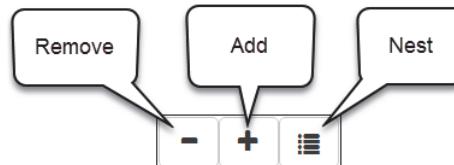
Triggers (“When”) There are six different possible settings for the trigger (i.e., the “when”):

- A property changes: the rule is triggered when one of the values for a property, normally in a Form, changes and the focus shifts from the field to another field.
- A user triggers an action button: the rule is triggered when a user clicks an action button, such as those associated with Action Bars. Use this trigger when you want to build a custom rule as a button for an Action Bar.
- An instance is created: the rule is triggered when an entity instance is created. For example, you may wish to trigger a rule to determine the default shipping method based on country when an order is created.
- An instance is deleted: the rule is triggered when the item is deleted. For example, you may wish to trigger a rule to determine the total cost of an order after a line item is deleted.
- A relationship changes: the rule is triggered when the specified relationship is changed. This occurs when a relationship is either added or removed, but does not support child relationships.
- An instance is initialized: the rule is triggered when a new instance of the entity is initialized. This is a useful rule to use when you want to populate user-defined default values before a Create form is displayed.

Conditions (“If”) Conditions are “if” statements related to a property of the entity. You can evaluate as many properties of the entity as are necessary and can choose whether to execute the corresponding action(s) if any or all of the conditions are true. If you are checking any of the conditions, then only one of the conditions needs to evaluate to true. If you are checking all of the conditions, then all of them must evaluate to true: if any one of them is false, then the corresponding actions will not fire.

Conditions are built similarly to filters in a List building block. You can add conditions by using the plus (+) button, remove a condition using the minus (-) button, or nest conditions using the nest button.

Figure 6-4:
Remove, add, or nest
conditions



Each condition must be developed with a property, an operator, and an operand. The operand may be optional for some conditions. As you discovered when building filters for lists, the operator depends upon the data type of the property; these operators could be checking if a property is empty or not, if a date field occurs before, after, or within the range of target dates, if number fields are greater than, less than, or equal to operand values, or if string fields contain smaller substrings. Operators such as “empty” and “not empty” do not require operands.

Nested conditions When evaluating conditions, you can choose to evaluate all or any of the conditions. If you're evaluating all conditions, then every condition in the statement must be true for the action to fire. If you're evaluating any condition, then only one must be true before the action is fired. In some circumstances, you may want to group sets of conditions together and evaluate them in more complex arrangements. For example, consider postal codes in the United Kingdom. Valid postal codes range from 5-7 alphabetic and numeric characters, in any of the following formats:

A9 9AA

A9A 9AA

A99 9AA

AA9 9AA

AA99 9AA

AA9A 9AA

You may choose to build a conditional statement comparing countries to postal code values in order to determine if the postal code is in a valid format:

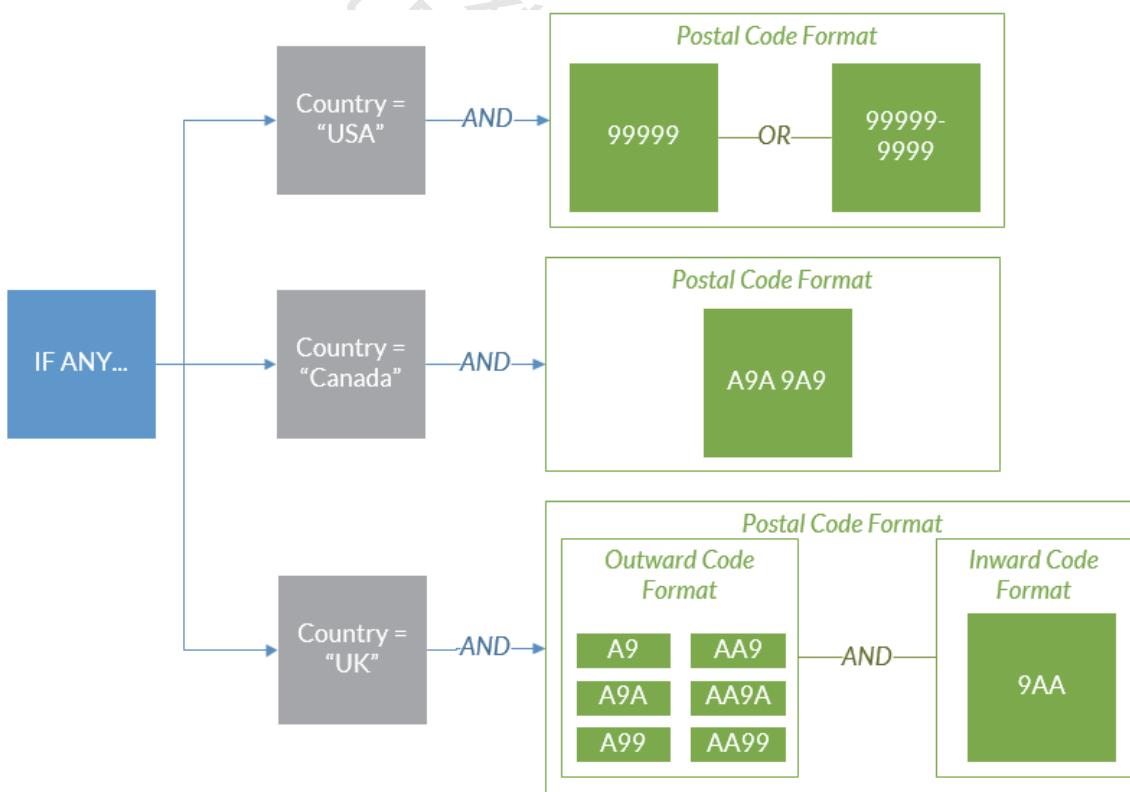


Figure 6-5: Graphic representation of valid postal code format rule

Actions (“Then”) If the conditions are satisfied according to their expression, then you can fire an action. Currently, you can only fire one action from a rule. As well, there are only a certain set of actions which you can perform depending upon the trigger.



There is no “else” option with a rule. That is, you cannot build a single rule which will set a field to either one value or another, depending upon the condition. You must build two rules, one for each circumstance.

The set of actions which you can execute are:

- **Show warning.** A warning reports that the entity instance requires attention. A warning does not prevent the user from saving changes to the entity instance, they are only intended for assistance and direction. The warning is displayed in Process Experience. You can choose the warning message to display to the user. Warnings are evaluated when an entity instance is opened or when any property referenced in the condition has been modified.
- **Show error.** An error prevents a Process Experience from making changes to an entity instance because the change would leave the item in an unacceptable state. When an error is triggered, it is presented to the user in Process Experience, but the error event is also captured at the server.



Errors are captured on the server, too, because the error event could cause other processes or activities to activate. For example, if the shipping cost is set to \$0 or less, you may want to throw an error which will trigger an e-mail to be sent to the manager.

In Process Experience, any rule which throws an error is reported immediately when the change is made. A new item cannot be saved while it is in an error state: the error must be corrected first.

It is possible for an existing item to be saved to the server while it is still in an error state. This can occur when a rule is added (or an existing rule is modified) after an item has been created. When a user opens such an item, the error is reported as a warning, not an error. As long as none of the properties involved in the error rule's condition are changed, the user is permitted to update the item and save changes. Any change to any of the condition's properties triggers the error rule and blocks saving changes.

You may set the error message which is to be displayed to the user in Process Experience.



Some fields have intrinsic errors, such as date fields which prevent typing text, or violating minimum/maximum value constraints. These intrinsic errors are signaled before user-defined error rules.

- **Show information.** Like warnings and errors, this option displays a message to the user in Process Experience. Unlike warnings and errors, the message which you choose to display is shown as a dialog on the screen. The user cannot proceed until the OK button is clicked. This option is only available when you choose the 'property change' trigger.
- **Set.** The set action sets a value for a targeted property. When using set, you must also specify the target and source. The target expresses the property you wish to change and the source represents an expression. This may be a fixed value, or it may be a mathematical equation which uses other fields (e.g., discount = total cost X 0.9).

The target field must be a fully qualified property. To be a fully qualified property, you must express it with the following syntax:

```
item.Properties.PropertyName
```

For example:

```
item.Properties.Discount
```

The source is an expression that may also contain fields, expressed as fully qualified properties or constant values.



If you have several rules which may change a single property, you must be very careful of the values you set; a recommended practice is to ensure that only one of the rules evaluates to true.



Property names are case sensitive. If the property name is entered incorrectly, an error is displayed in the rule editor.



You can often just type in the name of a simple property in the Target field and the rule engine will automatically complete the expression language if it locates a matching property in the entity.

When you add a property to the target of a set command, that property is automatically disabled in the appropriate forms. This is to allow the rule to set the value for the property, instead of allowing the user to manually change it. If a condition has been set on the rule, the target property is only disabled if the condition evaluates to true. If the condition is false, the property is not disabled and the user can manually change the value.

- **Hide.** The hide command will hide any named property on a form, or groups of properties which have been assigned a named category.
- **Disable.** The Disable command will disable any named property on a Form, or groups of properties which have been assigned a named category.



When you design forms, each field has an additional attribute named 'Category'. You may add a value to the field's category - even multiple fields with the same category - to have a rule hide/disable fields bearing that category. Form categories are arbitrary text and are case-sensitive.

- **Start process.** This action can be used to start a business process model which was created in Process Platform. Use the find button to locate an existing business process model, or click new in the results dialog to build a new ad hoc business process model. If you are selecting an existing business process model document, it must be visible from your collaborative workspace.

Figure 6-6:

Parameters for starting a process as a rule action

There are additional options available when you start a process. When the process is successfully started, a unique process ID is returned. This process ID (usually a long, hexadecimal number) can be stored in an entity's property. That property should be a text type. Use the Save process ID drop-down to select which property will store the process instance ID. This option can be used to disable an action button in an action rule that enables the user to start a process.



When designing your application, you should create an expression on the rule that will evaluate to true only when the text property is empty. This will disable the action button when the process ID has been filled in the property and prevent the Start process button from being clicked twice.

You can also add circumstances in which the business process will start, such as:

- When a property in the condition is changed, or
- When the condition changes to true.

As well as:

- On lost focus of form, or
- On lost focus of property.

- **Apply styles.** This action allows you to change the appearance of a field on the form by adding or changing its decoration. You can choose a property or form category and decorate the text (bold, italic, underline, color change), the field label (bold, italic, underline, color change), background color, or border color.



Add logic to the rule

1. Open the Order entity if it is not open already.
2. Select the propSetSpeedyBasePrice rule from the list.

The rule will currently be in an invalid state. This is normal and acceptable because you have not added any logic to the rule yet.

3. Click **Configure**.
4. Set the trigger “**When**” to **A property changes**.

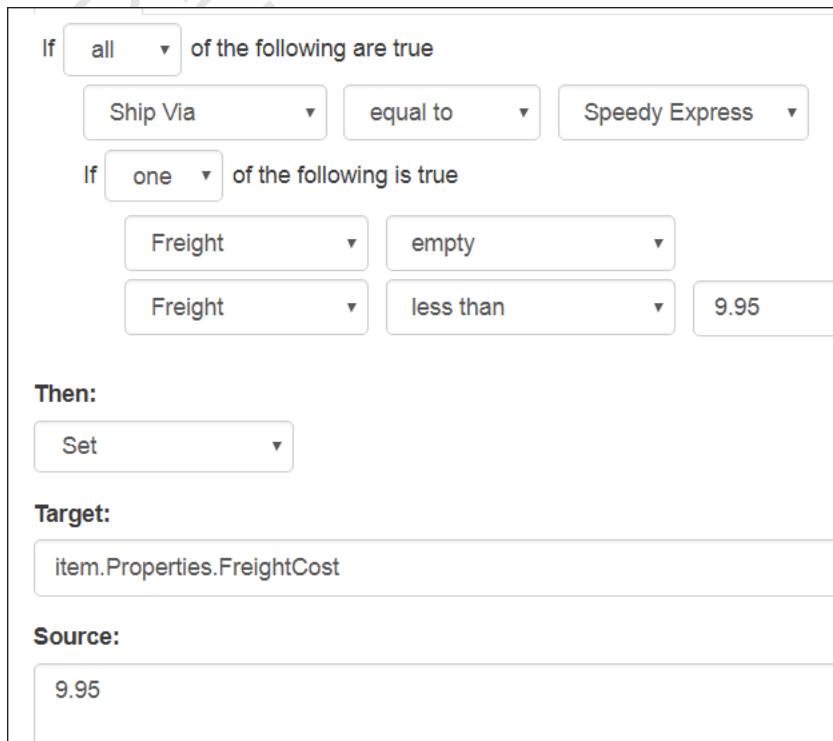
The rule will be evaluated whenever a property changes.

5. Under the condition section (“**If**”), set the inclusivity drop-down to **all**.
6. From the set of properties, select **Ship Via**.
7. Change the operator to **equal to**.

Ship Via is an enumerated field, so the operands are derived from a set list.

8. Select **Speedy Express** as the operand.
9. Add a nested condition.
10. Change the inclusivity condition to **one**.
11. Select **Freight** and the operator **empty**.
12. Add a condition in the nest.
13. Select **Freight** and the operator **less than**.
14. Set the value to **9.95**.
15. Under the action section ("Then"), set the action to **Set**.
16. Set the Target to **FreightCost**.

- If you have the name of the property correct (not the display name), then the rule editor will complete the full expression language automatically.
17. Set the Source value to **9.95**.
18. Click **Save**.
19. Close the rule logic dialog.



The screenshot shows the rule logic configuration for a minimum freight cost. The logic is structured as follows:

- Condition 1:** If all of the following are true
 - Ship Via equal to Speedy Express
- Condition 2:** If one of the following is true
 - Freight empty
 - Freight less than 9.95
- Action:** Then: Set
- Target:** item.Properties.FreightCost
- Source:** 9.95

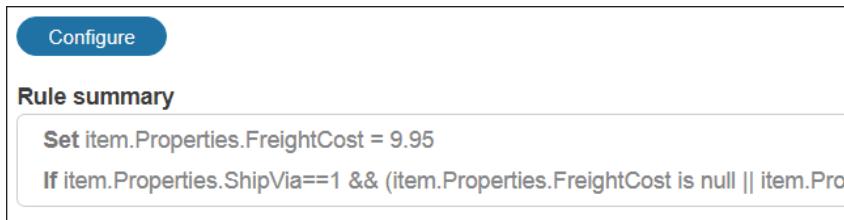
Figure 6-7:
Rule logic for Speedy Express minimum freight cost



After you save and close the logic, you are returned to the entity editor. The entity editor may still show that the rule is in an invalid state. Click on the rule or click refresh to update the entity with your changes.

20. Examine the Rule summary in the entity editor.

Figure 6-8:
Rule summary for
Speedy Express freight
cost



What the rule logic states:

Whenever a property changes, if ShipVia is Speedy Express and the Freight cost is either empty or less than 9.95, then set the Freight cost to 9.95.

Advanced mode

In advanced mode, you must use the rule expression language to define the rule logic. The rule expression language is easy to use, but does require a firm understanding of programming and development principles.



Building complex rule logic with the rule expression language in advanced mode is a development initiative which should be undertaken by development teams.

The fully qualified name of a property (i.e., `item.Properties.PropertyName`) is an example of rule expression language. In advanced mode, the fundamentals of setting rule logic is the same - the rule will have a trigger, a condition, and an action - but the rule condition will be written in expression language. In advanced mode, you are only provided with an expression language editor for building the condition.

Expression language for rules

Expression language can be used in advanced mode to write conditions for your rules. Expression language can be used on logical, mathematical, or even text values. Regardless of how you build the condition, it must nevertheless evaluate to a Boolean (i.e., true or false) result.

Your expression in advanced mode may contain operators and functions. Operators are typical logical and arithmetic operators commonly found in computer programming:

- Logical operators: `||`, `&&`, `true`, `false`, `==`, `!=`, `<`, `>`, `<=`, `>=`
- Mathematic operators: `+`, `-`, `*`, `/`, `% (mod)`

Furthermore, there are a large number of functions which you may use to process text values, including (but not limited to):

- **substring(sVal, iStart, iEnd)**: extract a value from a string from start point to end point.
- **length(sVal)**: number of characters in a string
- **sVal in list(val1, val2, val3, etc.)**: determine if a value can be found in a list (array)



You may also use regular expressions for a rule condition.

Keywords in expression language You may use entity instance data in your expressions as well. It's important to remember that when building complex business rules using expression language, you are using an instance of the entity, not the entity itself. As such, you cannot dynamically change an entity by adding, removing, or changing any of its building blocks, but you can work with certain run time values.

You can only work with the instance of one entity at a time: the entity in which you are building the rule logic. The instance of that current entity is referred to in the expression language as the *item*. Use the keyword *item* to capture the entity instance in its current context.

The *user* keyword refers to the current user logged in to the system. From the user, you can determine personal and role information.



There are other expression language keywords, such as *system*, which represents system values. The *system* keyword, however, cannot be used in rules.

When using a keyword such as *item* or *user*, you can access specific values or methods by adding the dot operator. For example, to access the whole set of an entity instance's properties, use:

```
item.Properties
```

for a specific property, such as First Name, use:

```
item.Properties.FirstName
```

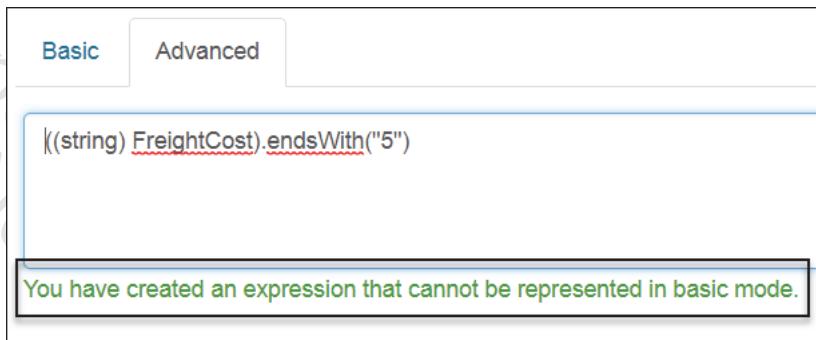


There are also many functions and attributes which you can use in expression language. Please refer to the documentation for a more complete guide to working with expression language in rules. This course is intended only to introduce you to entity rules and the expression language.

Using expression language to augment rules in basic mode It is a great advantage that the rule editor's basic mode builds rule logic in expression language, but leaves the underlying expression hidden to the user. However, the developer can switch to advanced mode in order to view the underlying expression add more logic to an existing basic rule.

When a rule is built in basic mode, select the advanced tab to view the underlying expression language representation of the rule. From this tab, you can change the basic rule by writing more complex rule logic. If the rule logic cannot be represented in basic, the rule editor will display a warning and prevent you from switching back to basic.

Figure 6-9:
Warning message when rule logic too complex for basic mode



Write a basic rule and edit it in advanced mode using expression language

1. Open the Customer entity from the Warehouse Application.
2. Click **Add > Rule**.
3. Set the Display Name to **Compare Fax and Phone warning**.
4. Set the Name to **propWarnFaxPhoneCompare**.
5. Set the Description to **Warn user if fax and phone number are identical**.
6. Click Add.
7. Click Configure.
8. Set the "When" trigger to **A property changes**.
9. Set the inclusivity operator for the "If" condition to **all**.
10. Select **Phone** as the first property.
11. Select **not empty** as the operator.
12. Click the plus (+) button to add a condition.
13. Select **Fax** as the property.
14. Select **not empty** as the operator.
15. Click the plus (+) button to add another condition.
16. Select **Phone** as the property.
17. Select **equal to** as the operator.

18. Set the operand to **Fax.**



This is not the property, **Fax**, rather it is a simple string.

19. Change the action (“Then”) to **Show warning.**

**20. Set the message to “Phone number and fax number are identical.
Please confirm with customer if this is valid.”**

21. Save the rule.

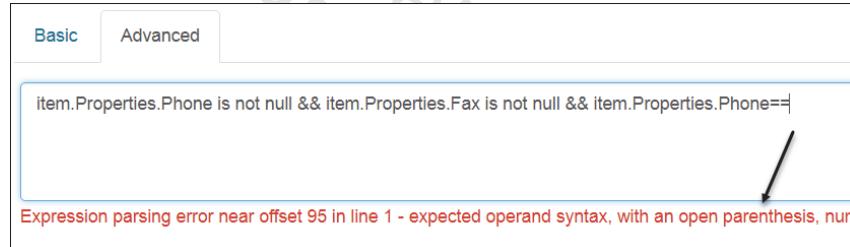
You must compare the phone and fax properties, but currently, you are only comparing the value of the phone property to text. You can only change this in advanced mode.

22. Click the Advanced tab.

23. Delete the word “Fax” after the equals signs, including the quotes, and replace it with **item.Properties.Fax.**

As you make changes to the rule logic, the syntax checker is simultaneously parsing your expression and may issue warnings. It is safe to ignore these warnings until you’ve completed your expression.

Figure 6-10:
**Live parser issuing
warning while writing
expression**



24. Save the modified rule.



You will notice, if you switch back to basic mode, that you are able to add expression language. You must however qualify it with an expression language tag:

`${item.Properties.Fax}`

25. Close the rule.

26. Examine the rule summary in the Customer entity.

27. Close the Customer entity.

28. Validate the Warehouse Application.

Expression language and title building block Earlier in this course, you created a Title building block. The title may contain either system-generated or user-generated text. If you are using a system-generated message for the title, you may choose to use expression language in the message. You cannot use expression language to perform any logical validations, but you can use keywords such as *item*, *user*, or *system* to set values. Whenever you use expression language in a title block, you must use the proper syntax (i.e., **{expression}**).



Use expression language in a title building block

1. Open the Order entity.
 2. In the entity editor, on the left side, select the Title building block.
 3. In the Title Specification field, set the value to:
`Order {item.Identity.Id}: Placed {item.Properties.OrderDate}`
 4. Save the Order entity.
 5. Open the sub-form of the Order entity which uses the basic order info, such as *Id*, *OrderDate*, and *Employee*.
 6. Click **Configure**.
 7. In the presentation header, click **Insert** section before.
- This option adds a divided section above the presentation of the form.
8. Drag the title to the upper section.
 9. Resize the upper section to fit the title.
 10. Clear the **Show label** option.
 11. Set the Type to **Text only**.
 12. Clear the **Default text style** option.
 13. Change the text style to **Size 20px** and **Color blue (#0000ff)**.
 14. Save and close the form.
 15. Save the Order entity.
 16. Validate and publish your changes to Process Experience.
 17. Refresh the Process Experience home page and confirm that the title is added to the form.

Figure 6-11:
Title using expression language on a form

The screenshot shows a form with a header bar containing three tabs: "Order Info", "Shipping Info", and "Items". Below the tabs, the text "Order 2: Placed 2017-12-11 12:45:00.0" is displayed. At the bottom of the form, there are two fields: "Order Number" and "Client Code".

Using a rule as a dynamic action on an action bar

One of the triggers of a rule may be an action button. As you learned when working with action bar building blocks, you can either use the static buttons available with building blocks such as file and history, or you can create your own buttons from rules.

To build a dynamic action button from a rule, create the rule from the Rule building block, configure logic for the rule in either basic or advanced mode, and set the trigger to “A user triggers an action button”. The display name of the rule is used as the label on the action button.

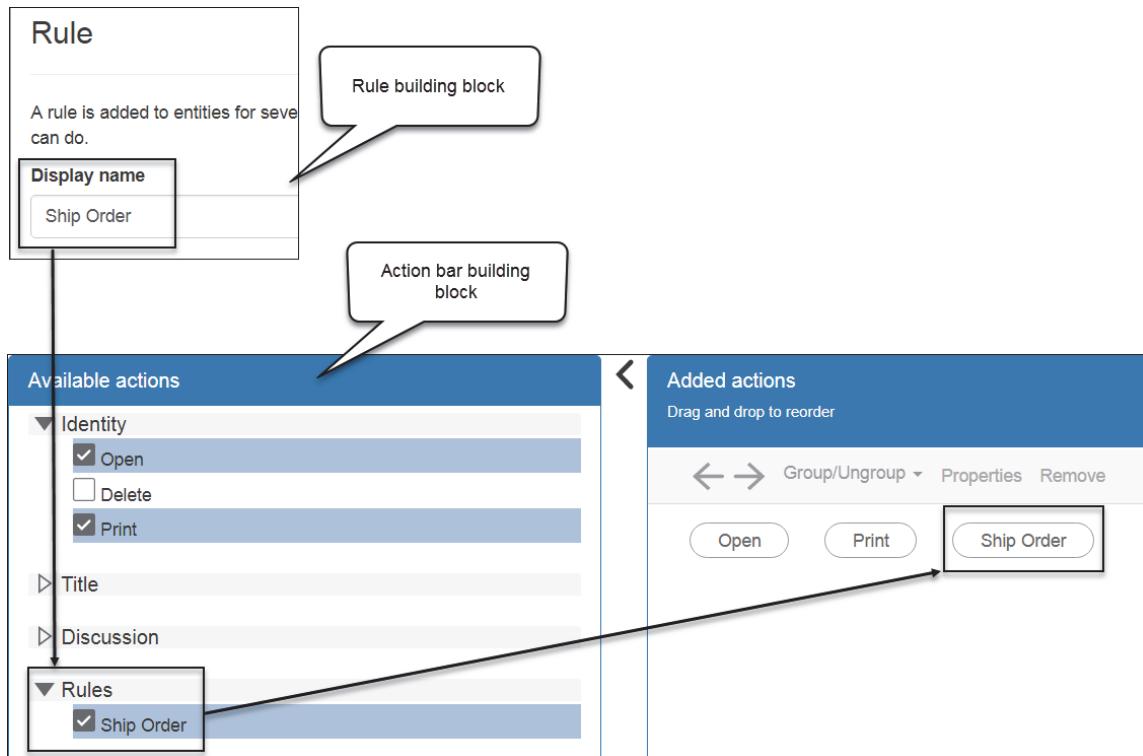


Figure 6-12: Rule used as action button in action bar



Add a button rule to an action bar

1. Open the Order entity from the Warehouse Application.
 2. Click **Add > Rule**.
 3. Set the display name to **Ship Order**.
 4. Set the name to **pbShipOrder** and the description to **Button rule to set the shipped date**.
 5. Click **Add**.
 6. Click **Configure**.
- The display name will be used as the button label.

7. Set the 'When' to **A user triggers an action button**.
8. Set inclusivity to **all**.
9. Add the following conditions to the 'If' clause. Click the add condition (+) button to add a new condition:
 - a. Employee not empty.
 - b. Freight not empty.
 - c. Freight greater than/equal to 0.
 - d. Ship Via not empty.
 - e. Shipped Date empty.

The logic which forms this rule is to make sure all relevant fields are complete before allowing an order to be shipped.

10. Set 'Then' to **Set**.
11. Set Target to **item.Properties.ShippedDate**.
12. Set Source to **now**.

now is a function which uses today's date and time.

13. Set After actions are performed to **Close current item**.
14. Save and close the rule.
15. Select the Order entity's default action bar and click **Configure**.
16. Under the Rules header on the left, select **Ship Order**.

The Ship Order button rule is added to the end of the action bar.



Even though a button is added to the action bar, it will not be usable, or even visible, until the condition is met.

17. Save and close the action bar.
18. Save the Order entity.
19. Validate and publish the Warehouse Application.
20. Check an order you have active in the Process Experience workspace.

If the conditions are met (shipped date is not set, employee, freight, ship via are set, and freight is more than 0), then the Ship Order button appears. Otherwise, the button will not be visible.

21. Click **Ship Order**.

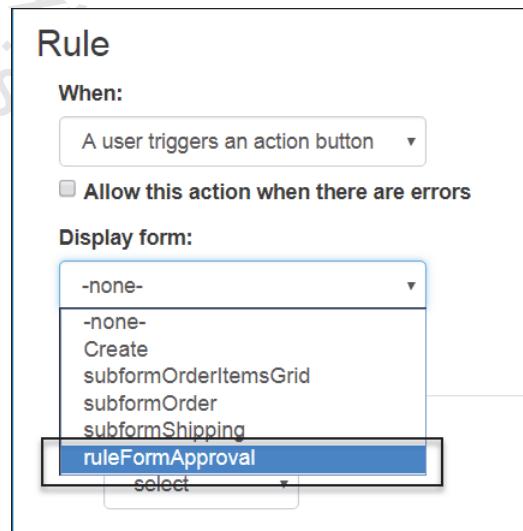
Three things should occur:

1. You should be returned to the default layout (i.e., the layout you were using before you opened the item).
2. The Shipped Date should be set to today's date and time.
3. The order should NOT appear in the Outstanding Orders list.

Display form When building a rule that uses the “A user triggers an action button” trigger, there is a field which is added named *display form*. In some circumstances, you may choose to display a particular form, soliciting input from the user, before the action is performed. An example may be when sending a form to a manager for approval. When the manager receives the form, you may choose to display a special form with Approve and Reject buttons, and provide a field so the manager may insert a comment. If you want to insert a form when clicking an action button, set the display form field to the form which you want to display. The form must be built beforehand, and it must be included in the current entity. The display form is presented to the user first, before the conditions are evaluated or the action performed.

Figure 6-13:

Display an approval form first when action button triggered



Use the display form to set values in the item which you can then use in the conditions of the rule.

In the following example, you will receive a sample delivery from your suppliers. This delivery will transfer stock from the On Order field to In Stock.



Add a button rule with a display form

1. Open the Product entity from the Warehouse Application.
2. Click Add > Property.
3. Set the Display name to **Shipment Received** and the Name to **ShipmentReceived**.
4. Set the type to **Boolean**.
5. Set the true value to **Yes**, the false value to **No**, and the none value to **N/A**.
6. Set the default value to **Yes**.
7. Click **Add and continue**.

This option will allow you to add a second property.

8. Set the Display name to **Comments**.
9. Set the Name to **ReceiveComments**.
10. Set the Property Type to **Long Text**.
11. Click **Add**.

The next step is to add a form which will pop up for the user when the shipment is received.

12. Click **Add > Form**.
13. Set the Display Name to **Receive subform**.
14. Set the Name to **ruleFormReceive**.
15. Click **Add**.
16. Click **Configure**.
17. Drag the On Order field to the form.
18. Drag the Shipment Received field to the form.
19. Resize the radio button list so labels are properly displayed.
20. Drag the Comments field to the form.
21. Under the Categories section of the Comments field, add a value - **IncorrectShipment** - and click **Add**.
22. Save and close the form.

If the shipment received matches the units on order, then there are no problems and you can process the order. Otherwise, if the shipment does not match the order, then you'll need to provide an explanation in the comments section.

By adding a category to the comments section, you can use a rule to either hide or show the comments section depending upon the value of the shipment received.

23. Click **Add > Rule**.
24. Set the Display Name to **Hide Shipment Comments**.

25. Set the Name to ***propHideShipmentComments***.
26. Set the Description to ***Hide the comments field if the shipment matches units on order.***
27. Click Add.
28. Click Configure.
29. Set the trigger to ***A property changes***.
30. In the conditions section, set the inclusivity marker to ***all***.
31. Set the rule condition to ***Shipment Received equal to Yes***.
32. Set Then to ***Hide***.
33. Set the first drop-down to ***Category*** and the second to ***IncorrectShipment***.
34. Save the rule and close the rule editor.
35. Click Add > Rule.
36. Set the Display name to ***Receive Shipment***.
37. Set the Name to ***pbReceiveShipment***.
38. Set the Description to ***Set InStock values from OnOrder***.
39. Click Add.
40. Click Configure.
41. Set 'When' to ***A user triggers an action button***.
42. Set the Display form to ***ruleFormReceive***.
43. Set inclusivity to ***all***.
44. Add the following conditions to the rule:
 - a. Discontinued equal to No.
 - b. On Order greater than 0.
 - c. On Order not empty.

The shipment button is only active if the product has not been discontinued and there is a shipment expected (On Order > 0).

45. Set Then to ***Set***.
46. Set the Target to:
`item.Properties.UnitsInStock`
47. Set the Source to:
`item.Properties.UnitsOnOrder + item.Properties.UnitsInStock`
48. Save and close the rule editor.

The next step is to add the button to the action bar.

49. In the entity editor, select the default action bar.
50. Click ***Configure***.
51. Select the Receive Shipment rule under the set of available actions.
52. Save and close the action bar editor.

53. Save the Product entity.
54. Validate and publish the Warehouse Application.
55. When you are testing the display form in Process Experience, you must first refresh the browser. Select a product, make sure its On Order value is greater than 0, and click Receive Shipment.
 - a. Does the form appear?
 - b. Is the comment field hidden appropriately?
 - c. Does the units in stock count adjust when the shipment is received?
 - d. What happens to the On Order value? Discuss.

History building block

You can maintain integrity of items in the run time solution. Some items must be heavily regulated and audited. The History building block is used to establish an audit history on items. The audit history is recorded in a special document called a *history log*. You can create and maintain as many different history log documents as you want in Process Platform.

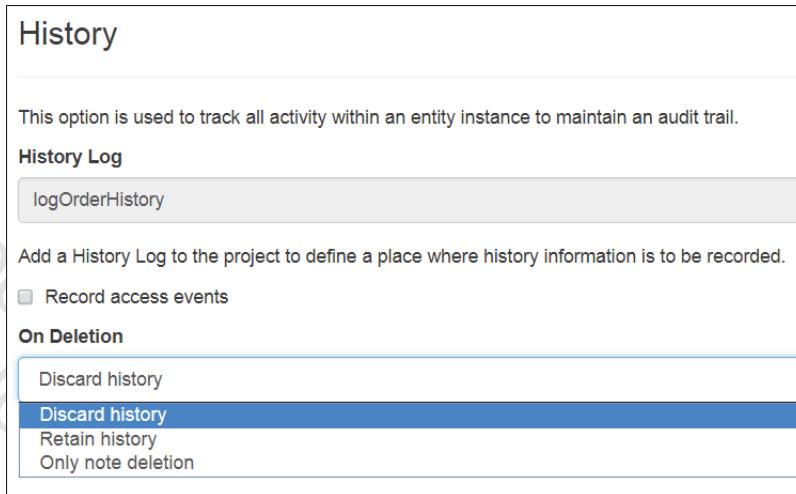
Only one History building block may be added to an entity. Therefore, the building block does not have a display name, and the name property is set automatically. History will track all activity within the item in order to maintain an audit trail. Events such as item creation and modification will be recorded to the history. You also have the option to add a record every time a user accesses the item.

If the item is deleted, you have different options for preserving the audit history. You may choose to discard the entire item history, to retain it, or you may choose to discard the entire history and record a single note indicating that the item was deleted. This option is referred to as “Only note deletion”.

Adding history

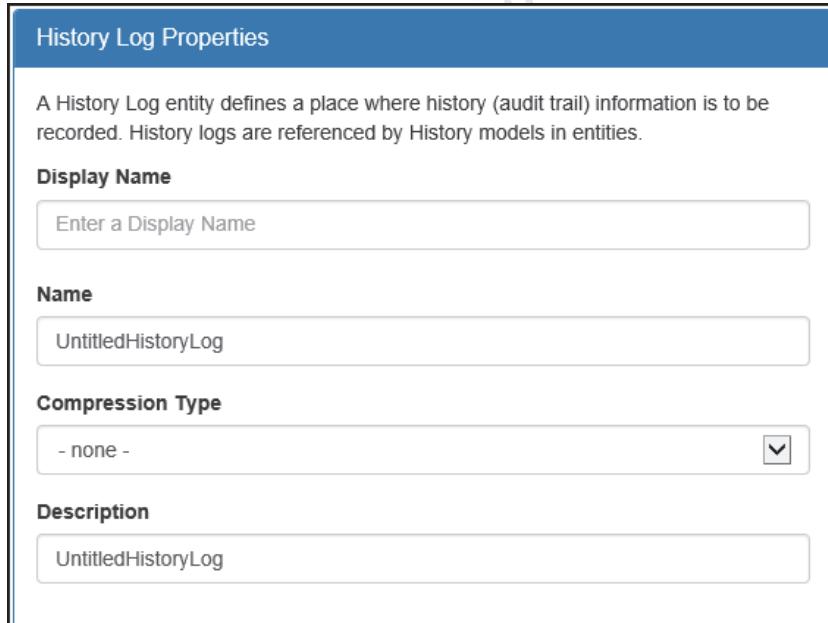
As mentioned, when the History building block is added, you cannot name it: the name and display name are both pre-defined in the block. You need only specify the log to which the audit history will be stored, whether to include access events, and what duties to perform on deletion.

Figure 6-14:
Basic details of History building block



History log documents History is recorded into a history log document. This is a document which is created and maintained in Process Platform. The history log document will contain a display name, name, and description, as well as information about how to compress the log. Default options for compression include Deflate and GZip.

Figure 6-15:
History log document properties





Add audit history to the Order entity

1. Open the Order entity from the Warehouse Application.
2. Click **Add > History**.
3. Click the Search button next to History Log.

There are no history log documents currently available, but you can create one.
4. Click **New > History Log**.
5. Set the Display Name to **Order History**.
6. Set the Name to **logOrderHistory**.
7. Set the Compression Type to **GZip**.
8. Set the Description to **Audit log for company orders**.
9. Click **Save**.
10. In the save dialog, keep the name and description. Next to the Location field, click **Search**.
11. Right-click Warehouse Application and select New Folder.
12. Set the folder name to **Security**.
13. Select Security as the folder and click OK.
14. Confirm that the History Log will be stored in the Security folder of the Warehouse Application and click OK.
15. Close the document.
16. Select the logOrderHistory log from the selection dialog and click OK.
17. Set **Record access events** to true.
18. Set On Deletion to **Discard history**.
19. Click Add.
20. Save the Order entity.
21. Select the default action bar.
22. Click **Configure**.
23. Select the History option to add the History button to the default action bar.
24. Save and close the action bar editor.
25. Save and close the Order entity.

26. Publish the Warehouse Application.



To test the History block, open and modify an existing order in Process Experience. The audit history will only begin recording after it has been published to the organization. All the events which occurred before publication were not captured in the log.

Figure 6-16:
**Sample of history log on
an existing order**

History	
Modified On ▾	Action
3/8/2018 12:24 PM	Item 'Order 1: Placed 2018-03'
3/7/2018 3:28 PM	Property 'RequestedDate' was
3/7/2018 3:28 PM	Property 'RequestedDate' was
3/7/2018 3:27 PM	Property 'RequestedDate' was
3/7/2018 3:27 PM	Item 'Order 1: Placed 2018-03'

Summary

Having completed this chapter, you should be able to:

- Explain the Rule building block
- Describe rule actions
- Explain the expression language for building rules
- Build a Rule for an entity
- Add audit history to an entity

Exercises

1. Use expression language to set the title of the Product entity to the item's name. (Hint: you may need to modify the Product Form in order to view the results in Process Experience.)
2. Use expression language to set the title of the Person entity to the last name and first name as in the following screen capture:

Personal Info		Address Info	Contact Info
Smiley, Guy			
Client ID	16,385	* Client Code	GSMIL
First name	Guy	* Last name	Smiley

(Hint: you may need to modify the Person entity to include the title, and modify the form in order to view the results in Process Experience.)

3. Write a rule for the Order entity which is triggered when a property is changed. The rule will throw an error if the requested date is less than the order date.
4. Write a rule (or rules) for the Order entity. The rule will set the requested date to three weeks from the order date. (Hint: make sure the order date has a default value of today first, then use simple arithmetic and the duration keyword to add three weeks to that date). The duration keyword uses the following syntax with 1-6 integers:

```
duration(years [,months [,days [,hours [,min [,seconds]]]]])
```

5. Write a rule named Discontinue that will set the Discontinued property for a product to be true. Use the rule as an action on the action bar.
6. Commit your work to the repository with the description: "Chapter 6: Rules and History".

7. Identity package and security

Objectives

On completion of this chapter, participants should be able to:

- Add business-relevant data to the identity package in Process Experience
- Extend an existing solution to include the identity package
- Build hierarchical and “flat” organizational models using the identity package
- Create a team worklist using the identity package
- Add permission-based security to an entity
- Add tracking to an entity
- Write and apply a condition to a security permission

Overview

Identity package is a special package installed with Process Suite which allows you to manage organizational models in your organization directly from the Process Experience runtime. A Process Experience end user with the appropriate authority can create organizational structures, organizational models, and units, as well as more advanced business functions such as managing a list of business contacts, countries and so on.

Once you have built organizational models for your business, you can then use the identity package and OpenText Directory Services (OTDS) to assign users to organizations, thus establishing a hierarchy of users and groups in your environment.

You can also use the Security building block attached to an entity in order to restrict access to elements of that entity, either at the user, role, or organizational model level.

You will also learn about the Tracking building block. Tracking allows users to follow changes on an item, but uses the identity package.

What you will build in this chapter

- Populating the identity package constructs with sample data.
- Security building block: this building block, when added to your entities, can restrict the use and function of other building blocks, such as properties and rules.
- Tracking building block: add a type of audit trail (like History) to your entities, except tracking does not use a log file and has more functionality.

Timing

Lecture: 75-90 minutes

Exercises: 60-75 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform 16.3 Entity Modeling Guide
- Process Experience 16.3 User's Guide

Identity package

Identity package is unlike many of the other constructs you've worked with thus far. Identity package is not a building block, rather it is a complete and deployed solution in Process Suite. You may have already noticed the identity package solution in the Process Experience administration console, and seen the identity package artifacts in the Process Experience runtime.

Identity package is a complete, pre-configured solution which was developed in part with entities. It is important to other solutions because it allows builders to add important structures such as organizational units and models, and to build those models into organizational hierarchies for users and groups. Identity package relies upon OTDS so that it may incorporate users into its solution.



In previous versions of Process Suite, organizational models and units were built directly in Process Platform as organizational units and models, respectively. Unfortunately, there were restrictions with these models in Process Experience, and changing a model required coding and repackaging a solution file. Furthermore, these organizational models could not be distributed in a multi-tenant solution. Since the introduction of the identity package, this is no longer the case.

Building blocks dependent on identity package Some of the newer building blocks in entity modeling rely upon elements of the identity package. For example, the Assignee building block, which you will learn about later in this course, accesses information from the identity package in order to assign a user or role to an item. The building blocks which depend upon the identity package are:

- Assignee
- Deadline
- Email

When you build your solution, you can also leverage all the entities, properties and lists which are included in the pre-configured identity package solution.



Since the identity package contains entities, you can build relationships to any entity in the identity package to incorporate it into your solution. This will allow you to add building blocks such as forms, lists, and rules which use properties from the identity package entities. Furthermore, you can also create an entity which is a subclass of an identity package entity in order to create your own, customized version of the identity package.

Entities in the identity package There are two types of entities used in the identity package: entities which represent OTDS objects and entities which are built exclusively to support the identity package.

Entities from OTDS The following entities are synchronized from OTDS:

- User
- Group
- Role

Master data entities The following entities - called *master data entities* - have been built exclusively for the identity package and are not synchronized with OTDS:

Address	Assignment	Country
County	EmailAddress	EmailAddressType
EmergencyContact	EmergencyContactRelationship	
Enterprise	Genders	Identity
Language	NationalId	OrganizationalUnit
Person	PhoneNumber	PhoneNumberType
Position	SocialAccount	SocialMedia
State	Title	VisaType
Worklist		

Each one of these entities may contain lists, properties, relationships, forms, layouts, and other building blocks.



A complete account of all the building blocks for each entity in the identity package can be found in the Entity Modeling Guide.



Add data to the identity package

1. Log in to Process Experience as the `analyst@training.local` user (password: opentext) if you are not already logged in.
2. Click **Home Page > Identity Homepage**.
3. Examine each of the lists provided under the **Identity** category.
4. Select **All countries**.
5. In the results panel, click add (+).
6. Set the name to Spain and the 2-digit ISO code to “ES”.
7. Click **Create**.
8. Open the country from the results list.

The country layout includes a list of all states/provinces.

9. Click add (+) to insert a new row into the state/province table.
10. Set the name to Valencia.
11. Return to the identity home page.
12. You may add languages, contacts, phone number types, social media entries, and visa types, if you wish.

You will work with users, contacts, enterprises, organizational units, and worklists later.

Some additional notes about these master data entities:

- The Person entity is not a user of the system; that is to say, a Person is not a User. A Person could be an individual who creates a profile on your website, someone who registers for services, a policy holder, someone who submits an insurance claim, or a contact person. In Process Experience, the Create form is called: “Contact”.
- A User is synchronized from OTDS and does use the system, but also has a relationship to the Person entity so that you may include properties of the Person.
- An Organizational Unit (OU) is a group of Users who form a functional team. Each user may have a qualifying designation, which is called a Position. Persons are assigned Positions, then added to the OU. This allows you to create an ad hoc group of users collaborating together as a single team with different roles, but each performing a specific function.
- Enterprise is a type of organizational unit. A single tenant in your Process Suite environment may contain one or more enterprises. Enterprises can sometimes be called companies or organizations, and can be subdivided into other organizations, called divisions. Enterprises are the root for organizational units.

Customizing identity package	<p>The contents of the identity package are contained in three CAP files. These contents cannot be edited or changed in any way. If you want to make changes to the identity package, you must extend an existing entity (by subtyping it) or building a new entity which has a relationship (or relationships) to an entity in the identity package.</p>
-------------------------------------	---

In this first example, you will leverage the identity package by connecting it into your existing solution. Your business entity contains a property, ContactName. You will replace that property with a relationship to the identity package.



Connect an entity into the identity package

1. Open the Business entity in the Warehouse Application.
2. Locate the property named ContactName and click **Delete**.
3. When prompted with a confirmation dialog, click **Yes**.
4. Click **Add > Relationship**.
5. Set the display name to Contact Name and the name to oneContact.
6. Set the type to “To peer” and the multiplicity to “To one”.
7. Click the **Browse** button for the target entity.

8. Select Person (Platform) from the filtered results and click OK.



Make sure you select the Person entity from the identity package (i.e., **Person (Platform)**) and NOT the Person entity from your solution (i.e., Warehouse Application/Entities/Person).

9. Click **Add**. Your solution is connected to the identity package.

The next step is to modify the form to include the new connection.

10. Select the Create form from your Business entity.

11. Click **Configure**.

The contact name field has been removed from the form because the field was deleted.

12. Drag the [0..1] Contact Name relationship from the Components list to the form presentation.

13. Change the Type to **Text box**.

14. Set the Browse list to **PersonList**.

This list was included with the identity package.

15. Set the Property to **DisplayName**.

16. Change the label to Contact Name.

17. Under the Actions subsection on the right, make sure that Browse, Create, and Clear are all selected.

18. Save and close the form.

19. Click **Add > Form**.

20. Set the display name to Contact ID Package subform and the name to subformContactIDPkg.

You created a Contact subform at the end of a previous chapter.

21. Click **Add**.

22. Click **Configure**.

23. Drag the [0..1] Contact Name relationship and drop it on the presentation.

24. Change the type of the presentation to text box.

25. Set the Browse list to **PersonList**.

26. Set the Property to **DisplayName**.

27. Set the label to Contact Name.

28. Make sure that the Clear, Create, and Browse actions are available.

29. Drag the following fields from the [0..1] Contact Name relationship and drop them in a sensible arrangement on the presentation:
 - FirstName
 - LastName
 - User_ID
 - Email
 - Phone
 - Mobile
30. Set each of these properties from the identity package to Read only.
31. Save and close the form.
32. Select the workWithBusiness layout.
33. Click **Configure**.
34. Select the Contact Info subform from the stack of forms if it is present. If it is not present, add a new form panel to the stack of forms.
35. Set the Name of the form to Contact.
36. Set the Form to **Contact ID Package subform**.
37. Save and close the layout.
38. Validate your work and publish it to the Process Experience runtime.
39. Return to Process Experience and refresh the browser to collect the published changes.
40. Select the Client List from the home page.
41. Open one of your business customers and select the Contact tab.
42. Click the add (+) button to create a contact for the customer.



The table for addresses is an example of a repeating group container. The State/Province field is an example of a filtered field, as presented in the Tips and Tricks section of the previous chapter.

43. Add some information and save the contact.

Client Info	Contact Info	Address Info
Contact Name	Jeff Wall	<input type="button" value="Search"/> <input type="button" value="+"/> <input type="button" value="X"/>
First name	Jeff	Last name
User ID		Email
Phone	905-937-4597	Mobile

Figure 7-1:

Business customer layout after connecting Business entity to Person entity in identity package

Synchronization, consolidation, and push service With OTDS, users and groups (if applicable) are created in source directories, such as Active Directory, LDAP, or other sources. When OTDS is installed and configured, it synchronizes changes to these source directories and updates the active list of users and groups in OTDS. Changes can also be forcibly captured from directory sources by consolidating them.

Another part of the OTDS configuration is to build services to push users and groups to OpenText products. This is called a *push service*. Once OTDS has received an updated list of users and/or groups, the push service sends those user IDs to OpenText products such as Process Suite, Content Server, Media Manager, and more. This push synchronization is typically performed one-way: from OTDS to OpenText product, and not the reverse.

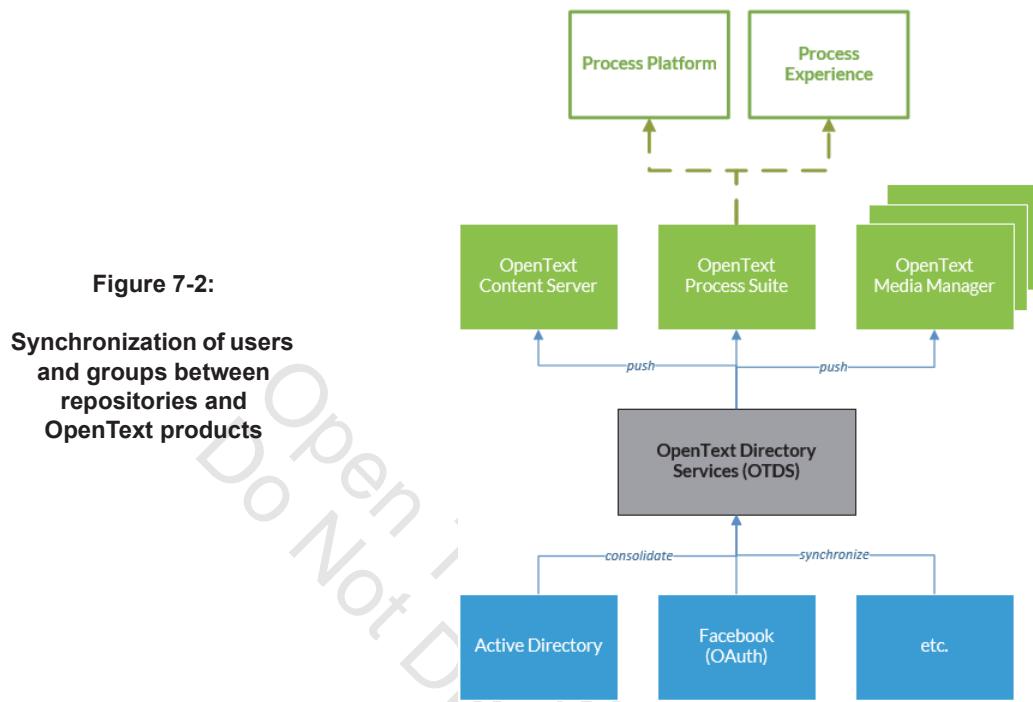


Figure 7-2:
Synchronization of users and groups between repositories and OpenText products

 The push connector is called the OTDS Push Connector. If you want to learn more about OTDS, consider course 3-0300 OTDS Installation and Configuration.

The nature of this synchronization dictates that users and groups must be built first in source directories, such as Microsoft Active Directory. OTDS synchronization will pick up changes and push them to OpenText products so that users can log in to those products using their Microsoft Active Directory user ID and password. From that point, OpenText Directory Services will enforce single-sign on (SSO) for all connected OpenText products.

With the identity package, users, groups, and roles from source directories are modeled as entity instances in Process Suite. Identity package includes other entities to create a more robust profile of each user or group captured.

 Because your system uses Process Suite with OTDS, users are not added in Process Platform directly. Rather, users are added to the Active Directory, synchronized to OTDS, and consolidated with Process Suite.

**Add a user**

1. Start the Windows Server Manager.
2. Click Tools > Active Directory Users and Computers.
3. Click **Create a new user**.
4. Set the First name, Last name, and the Full name.
*For the purpose of this exercise, we will use **Ramiro Alba**.*
5. Set the User logon names to a shorter representation of the name which is between 4-8 characters.
*In this example, we will use **ralba**.*
6. Click **Next**.
7. Set the password and confirmation to **opentext**.
8. Clear the **User must change password at next logon** option.
9. Set the **User cannot change password** and **Password never expires** options.
10. Click **Next**.
11. Review the settings then click **Finish**.

The user should appear in the active directory list.

12. Open a browser window.
13. Launch the OTDS administration console.
<http://localhost:8080/otds-admin>

This is available as a shortcut in both Google Chrome and Internet Explorer.

14. Log in with user name **otadmin@otds.admin** and password **opentext**.
15. In the OTDS administration console, select Partitions.
16. Click ClassDomain > Actions (at the end of the row) > View Members.
17. Wait several moments. The user should be added automatically to the partition.

Guest	Guest@training.local
krbtgt	krbtgt@training.local
otadmin	otadmin@otds.admin
OTSysUser	OTSysUser@training.local
Ramiro Alba	ralba@training.local

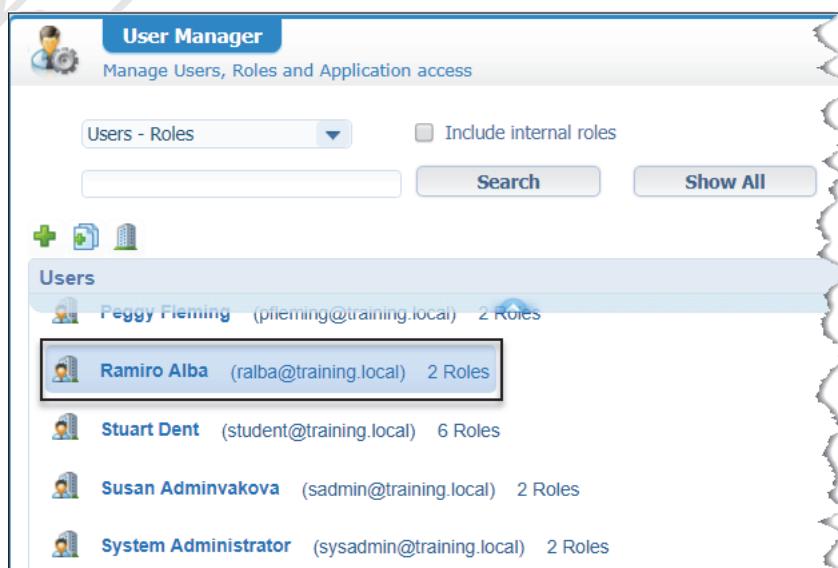
Figure 7-3:
User automatically
synched to OTDS



Working with users, consolidation, and the push connector is discussed in course 3-4913: Process Platform Administration.

18. Assign the user some roles specific to Process Platform. Click the Actions link for user ralba and select Edit Membership.
19. Click Add To Group.
20. Search the list for the Cordys@Work#Developer role. Select it and click Add Selected.
21. Add the Cordys@Work#Analyst role as well.
22. Close the group window.
23. Wait several moments. Log in to Process Platform as user analyst@training.local (password: opentext, if required).
24. Switch to the System organization.
25. Start the User Manager application from the application palette.
26. With Users - Roles selected in the drop-down, click Show All.
27. Locate your new user, Ramiro Alba, from the list of all users.

Figure 7-4:
User automatically synched to Process Platform



Your new user still has to be enabled to work in the Warehouse organization. You will learn how to process users in course 4-4913: Process Modeling for Process Platform.

28. Switch back to the warehouse organization afterwards.

Working with organizational models

Users are added to groups in OTDS. These users and their groups are then synchronized to Process Suite so that they can be accessed through the identity package in Process Experience. Groups, however, do not suggest or imply any hierarchical order to the way in which employees are related. An organizational model, which consists of organizational units, will capture these hierarchical structures.

Organizational models are built with the identity package in Process Experience for a number of reasons:

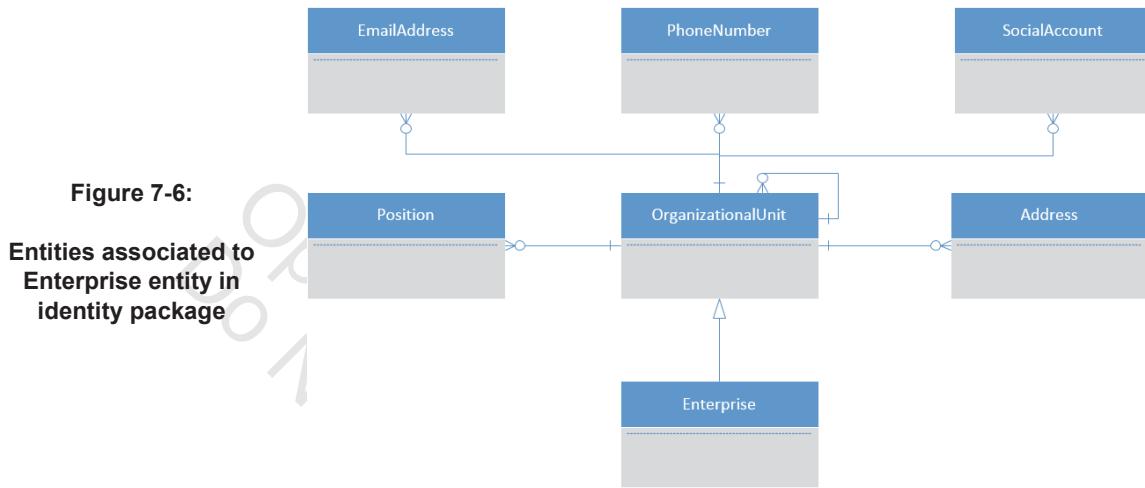
- Organizational models typically capture a run-time picture of a business, and Process Experience is used in a business run-time.
- Organizational models can be edited quickly and easily in Process Experience without the need for development.
- Organizational models can be changed and made available without the need to build a new version or redeploy.
- Different organizational models can be offered for different business organizations.

Enterprise The enterprise entity, as you have learned, is the root of an organizational model. A business will logically have one or more enterprises, which are then subdivided into organizational units (sometimes referred to as divisions or *subunits*). You will typically begin modeling a business by creating a root enterprise.

The screenshot shows the 'Create Enterprise' dialog box. On the left, there are three main input fields: 'Name' (containing 'Northwind Trading Company'), 'Description' (containing 'An enterprise to model the company business, Northwind Trading.'), and 'Type' (set to 'Hierarchical'). On the right, there is a sidebar titled 'All Addresses' which lists a single address entry: 'Address line 1: 123 Anywhere St.', 'City: Erehwon', 'Country: Spain', and 'State/Province: Valencia'. There is also a '+' button and a small checkbox next to the address entry.

Figure 7-5: Create enterprise dialog

The dialog for creating an enterprise is a Create form consisting of some properties and a relationship to the Address entity. The identity package entities which are associated to Enterprise are captured in the following class diagram:



Enterprises can be flat or hierarchical. This setting is set in the Create enterprise dialog and cannot be changed after it has been established. A hierarchical enterprise is one in which there is a clear employer/employee relation. A flat enterprise (one in which hierarchy is set to “none”) considers every user in every position equally.

After an enterprise is built, you need to describe the positions available in the enterprise. A position is typically a job assignment to which people (person entities) are assigned. It's important to note that in an organizational model, a single user may have several roles but typically one position in the enterprise. Only after the positions are laid out are you able to assign people to them.

Of the positions available, one may be described as the *lead*. The lead position is a position which describes the individual with the most authority in the enterprise or organizational unit. That is, the lead position is the leader of the team. This may not necessarily be the same as rights and privileges which are granted with roles: a user in lead position does not need to be higher in the hierarchy. This allows some flexibility when you are defining your enterprises, and allows you to create ad hoc teams for special events.

Figure 7-7:
Setting lead position

The screenshot shows a software interface with a header containing tabs: 'Summary', 'Positions' (which is highlighted in blue), 'Members', and 'Subunits'. Below the tabs is a table with three rows. The first row has columns for 'Position Name' (checkbox) and 'Descriptor...' (text input). The second row has columns for 'Position Name' (checkbox) and 'Descriptor...' (text input). The third row has columns for 'Position Name' (checkbox), 'Descriptor...' (text input), and 'Lead' (checkbox). In the third row, the 'Lead' checkbox is checked, and the value 'true' is displayed in the adjacent column. The 'Descriptor...' column contains the text 'Finance manager'.

Many people can be assigned to a single position. In some circumstances, you may want to choose one person in a position to be the most important in the position. This is called the *principal*. A principal person in a position can be assigned special tasks for the group. The position does not need to be a lead position for you to assign a principal.

Finally, you can subdivide the enterprise into smaller units, or *subunits*. Each subunit is an organizational unit which borrows (inherits) users, people, and positions from its parent: the enterprise. The hierarchical structure of the organizational model is established by adding subunits to the enterprise.



Build an organizational model with an enterprise

1. Log in to Process Experience as the `analyst@training.local` user if you are not logged in already (password: `opentext`).
 2. Select the identity homepage.
 3. Expand Identity and click the **All enterprises** list.
- Currently, there are no enterprises in the results panel.
4. Click add (+) in the results panel.
 5. Set the Name of the enterprise to **Northwind Trading Company**.
You may also add a random description.
 6. Set the type of enterprise to **Hierarchical**.
 7. In the All Addresses grid, click add (+) to build a new address for the enterprise.

8. *Expand the row to see the fields of the address.*



The All Addresses table is an example of a repeating group container for a “to-many” relationship.

9. *Set the address to be: 6 Carrer de la Safor, Sant Pau, Spain, 46015.*

10. *Next to the State/Province field, click the browse button and select the province you created in a previous exercise: Valencia.*

11. *Click Select.*

12. *Click Create.*

The enterprise is presented in the All enterprises results panel.

13. *Open the new enterprise in the layout.*

14. *Select the Positions tab.*

15. *Click the add (+) button to add the following positions to the enterprise. You do not need a description. Leave the Lead set to false:*

- Finance manager
- Director (Lead is true)
- Warehouse manager
- Sales manager
- Business analyst

16. *Select the Members tab.*

All of the positions should be presented in a repeating group container.



If the list of positions does not automatically refresh in the Members tab, return to the Identity Homepage, re-select and open the enterprise from the results panel.

17. *Return to the Identity Homepage.*

18. *Select the All organizational units list.*

19. *In the results panel, click add (+) to add an organizational unit.*

20. *Set the name to Northwind Home Office and the type to None.*

21. *In the All Addresses list, click add (+) to insert an address into the repeating group container.*

22. *Set the address to: 1599 Park Ave., Springfield, United States, Ohio, 45503.*

23. *Next to the Create button, click the options arrow and select Save and create another.*

24. Set the Name to *Northwind Distribution Center* and the type to *None*.
25. Add an address to the list: *415 Keyser Ave., Scranton, United States, Pennsylvania, 18504*.
26. Click **Create**.



These two organizational units which you just modeled do not have subsidiaries (i.e., child organizational units) and are intended to represent different businesses both logically and physically. This is only one way of modeling an organization.

27. Select the *All enterprises* list from the *Identity Homepage*.
28. Open the *Northwind Trading Company* enterprise.
29. Select the **Subunits** tab.
30. Click the *browse* tool.
31. Select the *Northwind Home Office* and *Northwind Distribution Center* and click **Select**.

You have established an organizational hierarchy between Northwind and two of its physical locations.

Worklists

Identity package and organizational models are important for distributing work amongst teams and team members. You can distribute work (often called *tasks*) to users based upon their user name, group, or role, but organizational models allow more flexibility for distributing tasks to of users regardless of their group or role.

A worklist is an identity package construct which matches work tasks to members from an organizational model (such as positions and organizational units). These work tasks can originate from flow constructs such as a lifecycle or activity flow, or they can be assigned manually.



You will learn about the Lifecycle, Activity Flow, and Assignee building blocks in later chapters.

If you define worklists with business processes, users with the appropriate level of authority can connect a worklist to a team. For example, if you have a worklist for complex cases and another worklist for simple cases, each can be linked to a different team with the appropriate level of skills to help resolve the case.



Build a worklist for a team

1. Open the Identity Homepage from Process Experience.
 2. Select the All worklists list.
 3. In the results panel, click the add (+) button.
 4. Set the Name to Billable Account Setup.
 5. Set the Description to: Setup accounts for new customers as billable or pre-paid.
 6. Click Create.
 7. Locate the new worklist in the results panel and open it.
 8. Select the **Units** tab.
 9. Click the browse button.
- Work must be assigned to organizational units.
10. Select the Northwind Home Office org unit and click **Select**.
 11. Select the **Lead position(s)** tab.
 12. Click the browse button.
 13. Select the Finance manager position and click **Select**.



Worklists can be used in a lifecycle, which is presented later in this course.

Identity package administration

There are some aspects to administering the identity package which you should be aware:



Administration tasks are not discussed in any meaningful way in this course. If you want to know more about administration in Process Suite, consider course 3-4903: Process Platform Administration.

Roles There are a number of roles which come with the identity package. These are “metadata roles”: roles which are used to administer or work with the identity package and not the roles which are modeled in OTDS. These roles are:

- Identity Administrator
- Identity User
- Identity Push Service

The Identity Push Service role is considered to be a “dummy” role responsible for synchronizing data between OTDS and the identity package in Process Suite.

Managing synchronization with OTDS Synchronization with OTDS is not true two-way synchronization: only user and group data is pushed automatically from OTDS to Process Platform, but not role information. In order to synchronize roles, you need to run an *OTDS Configurator* tool. An administrator will run this tool from a command line. The OTDS Configurator may also be necessary if you are installing and deploying a CAP file built and packaged in a completely different Process Platform environment. The reverse is also true: if you are removing/undeploying a CAP file from your Process Platform environment, you may need to run the OTDS Configurator to remove the roles which that package required.



If you are upgrading a package in Process Platform, be careful if you are renaming your roles: a package which contains roles already deployed may need the OTDS Configurator to update changes to role names. Names cannot be updated with the OTDS Configurator: the relationships are lost and need to be remapped.

Tracking building block

The Tracking building block adds information to an entity to track when and by whom the item was created and last modified. Each entity can include only a single Tracking building block.

Tracking is different from history in that properties and relationships may be tracked, whereas history does not.

Tracking information is managed through two relationships to the user entity (Created By and Last Modified By) and two date/time properties (Created Date and Last Modified Date).



You cannot see or edit the relationships and properties, although they may be visible in forms, lists, and rules.

The actors on the item (that is, the creator and modifiers) are identified in the tracking building block with a Person relationship. This is the same Person entity as used in the identity package, which will provide you access to other aspects of the actor, such as address(es), language(s), phone numbers, and so forth.

Tracking on child entities

Tracking can also be added to child entities. If it is added to the child, an option is available to capture the child tracking information as part of the parent entity.

If tracking is added to an entity:

- When creating a list, you can add CreatedDate and LastModifiedDate properties and two relationships to the User entity. For example, it might be helpful to show the user who last modified an order and when it was modified. You can also create a list filter based on creation date and last modified date or based on the relationships.
- When creating a form, you can add CreatedDate and LastModifiedDate properties and two relationships to the User entity. For example, a form for creating an order can include the date when it was created and who created it.
- When creating a rule, you can access the CreatedDate and LastModifiedDate properties using expression language.



Use the following expression language syntax:

```
item.Tracking.CreatedDate  
item.Tracking.LastModifiedDate
```

**Add tracking to the Order entity**

1. Open the Order entity from the Warehouse Application.
2. Click **Add > Tracking**.

No other details are necessary.

3. Click **Add**.
4. Select the All Orders list.
5. Click **Configure**.

6. In the list of available properties, select Tracking > Last Modified Date.

Some properties added with Tracking building block

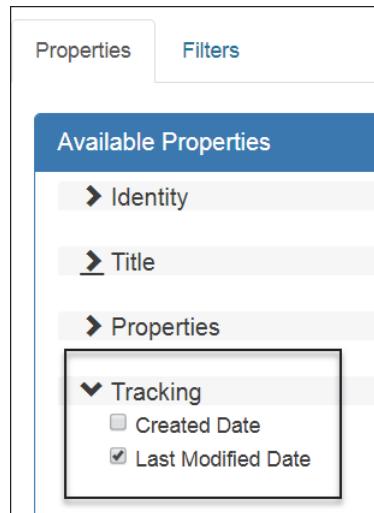


Figure 7-8:

7. Select LastModifiedBy > Properties > FullName.
8. Set the column label to "Last Modified By".
9. Save and close the list.
10. Validate and publish the solution.
11. Refresh the Process Experience page. Make changes to an order and examine it in the All Orders list.

Tracking properties in the All Orders list

The screenshot shows the 'Order List' page. At the top left is a green circular icon with a white document symbol. To its right is the page title 'Order List'. Below the title is a toolbar with icons for search, add, and refresh. The main area is a table with the following data:

Order Number	Customer	Ordered On	Sales	Requested ...	Last Modified Date	Last Modified By
1	CORPT	3/6/2018 11...	Leverling, J	3/20/2018	3/8/2018 12:33 PM	Anna Lyst
2	FOOTP	3/6/2018 11...	Peacock, M		3/27/2018	

Security building block

As you build your solution using entities, you may want to control who has access to which data, properties on forms, and so on. You can use the Security building block to restrict access based upon the Process Experience user's role. A role is an arbitrary description of what functions and duties the user has the ability and right to perform. It is customary that a user will have one or more roles depending upon their function. For example, Manager and Employee may be two roles in the system, but an Employee in the Human Resources department will have different functions than an Employee in Accounting. Consequently, there may be department-specific roles which further determine what the employees are able to do in Process Experience.

Roles are assigned both at the OTDS level and in Process Platform. This establishes a distinction between *access roles* and *functional roles*. An access role is built in OTDS and synchronized to Process Platform. It describes the authority which its members (e.g., users) have to access a system. For example, a user who needs access to Process Platform must have an access role defined in OTDS. A functional role is defined in Process Platform. It is a high-level role which defines the access that a user has to certain functions, such as applications in the application palette of the user start page. Functional roles also define the access a user has to an entity and the parts of that entity.

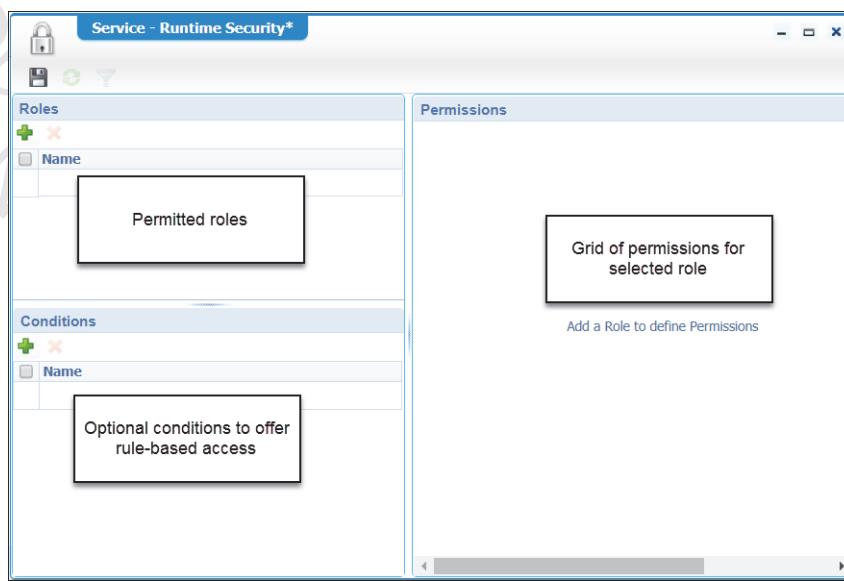
When you add the Security building block to an entity, you need to indicate the role to which the restriction will apply. Before the application can be used in Process Experience, those roles must be mapped to actual Process Experience users. A system administrator or security manager will normally perform this task in Process Platform.



A functional role is a Process Platform document, just as an entity is a Process Platform document.

The Security building block is optional and can only be applied once. It does not have any details: it does not have a display name, a name, or a description. The Security building block defines the authority a user has to parts of an entity - its properties and lists - the type of access the user has: the ability to create, read, update, or delete entity data, and any optional conditions which may offer rule-based access. The Security building block is added and then configured with roles, permissions, and optional conditions. Add the Security building block like you would any other entity building block, then select it from the entity editor and click the Configure button.

Figure 7-10:
Configuring the Security building block



Access Control List and permissions Each entity can have only one Security building block added to it. The Security building block allows the builder to establish a relationship between the entity and any number of roles. The nature of that relationship is realized in the permissions which the role is allowed with the entity. This is called the *Access Control List (ACL)*.

There are certain permissions which each role is allowed in the ACL:

- Create: create an instance of the entity.
- Read/View: view, but not change, the data in an entity instance.
- Update: change the data of an entity's instance.
- Delete: delete the entity instance.

As well, permissions can be established on building blocks, such as:

- Assignee: specifies the user responsible for the entity. You will learn more about assignee in an upcoming chapter.
- Business workspace: enables entities with extended ECM capabilities and helps users to manage workspace and content in the applications.
- Content and file: enables users to add documents (or, in the case of the Content building block, multiple documents) and manage those documents through various actions exposed by the building block.
- Deadline: set or update a deadline.
- Discussion: permit comments.
- Email and email template: users can send email with a specific template.
- Lifecycle: enable specific permissions for individual tasks in a lifecycle. For lifecycle tasks with user input, use security to permit certain user actions to be exposed.
- Lists: you can restrict who is allowed to use (i.e., read) a list in Process Experience by offering access on a role-basis to each list.
- Properties: each property of the entity may have read and/or update capabilities. This allows you to control fine-grained access to individual properties of the item.
- Title: when the title is user-generated, this authority will allow you to control who is allowed to set and read the title of an item.
- Tracking: adds information to an entity to track when and by whom an instance was created and last modified. Each entity can include only a single Tracking building block.
- Web service: enables the user to use a find-type of web service operation.
- Rules: you can restrict who is allowed to use (i.e., execute) a rule in Process Experience by offering access on a role-basis. This function allows you to restrict access to dynamic buttons.

Permissions			
Name	<input type="checkbox"/> Read	<input type="checkbox"/> Update	<input type="checkbox"/> Other
Entity Product	<input type="checkbox"/> View		<input type="checkbox"/> Create <input type="checkbox"/> Delete
Properties			
ProductName	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
CategoryID	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
QuantityPerUnit	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
UnitPrice	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
UnitsInStock	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
UnitsOnOrder	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
ReorderLevel	<input type="checkbox"/> Read	<input type="checkbox"/> Update	
Discontinued	<input type="checkbox"/> Read	<input type="checkbox"/> Update	

Figure 7-11: Configurable permissions on Product entity

Process Experience follows a permissive model. That is, everyone has access to the solutions installed unless that access has been restricted by implementing the Security building block.

When configuring the Security building block, use the plus (+) button under the list of roles to add a role to the list. You can choose from any existing role in Process Platform, or build a new role document. Select the role from the list and assign permissions on the right side.



Any new role document which you create will be available to other documents in the workspace, such as business process model and case model documents.



Add security to the Product entity

1. Open the Product entity from the Warehouse Application.
2. Click **Add > Security**.
3. Click **Add**.
4. Select the Security building block from the list.
5. Click **Configure**.
6. Only Product Managers should be able to delete products from the database.
On the Roles side of the ACL editor, click the plus(+) button to add a role. The current roles available are only for using the identity package.
7. Click **New > Role**.
8. Set the Name of the role to **Product Manager**.

9. Set the Description to **Manager of Product Catalog**.
10. Set the Type to **Functional**.



The role editor is discussed in course 4-4913: Process Modeling for Process Platform. User interfaces refers only to XForm user interfaces built in Process Platform. It does not include forms or layouts.

11. Click **Save**.
12. In the Save dialog, keep the Name and Description of the role.
13. In the Location field, click the Browse button.

The Warehouse Application folder structure appears.

14. Expand **Warehouse Application**.
15. Right-click **Warehouse Application** and select **New Folder**.
16. Set the name of the folder to **Roles**.
17. Select the **Roles** folder and click **OK**.

New roles will be stored in the Warehouse Application/Roles folder.

18. Click **OK** in the Save dialog.
19. Close the Role editor.

Your new role should now be available in the role selection dialog.

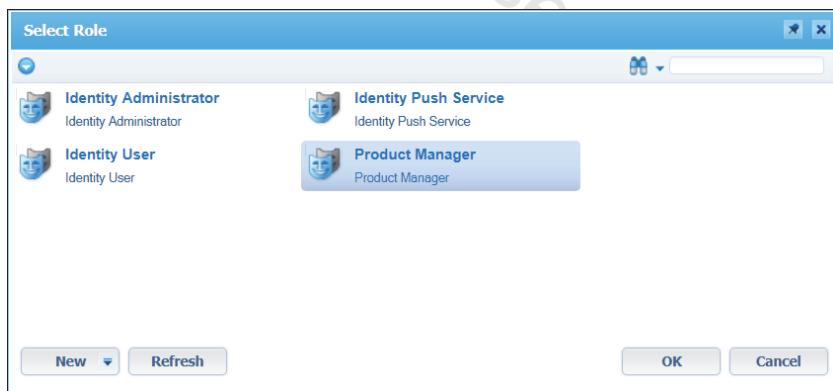


Figure 7-12:
New role in role selection dialog

20. Select the **Product Manager** role and click **OK**.

The Product Manager role is added to the list of roles in the ACL.

21. Select the **Product Manager** role on the left.
22. Select all the permissions on the right.

Only Product Managers should have the ability to work with products. Use the column headers, Read, Update, and Other, to easily select all the permissions in each column.

23. Save the configuration window.
24. Close the editor.
25. Save and close the Product entity.
26. Validate and publish the Warehouse Application.

Validation will also be performed on the new Product Manager role.



After you publish to the organization, return to Process Experience and test your work. Access to the Product entity should be unavailable. This is because your user has not been mapped to the Product Manager role.

27. Open the application palette and launch the User Manager application.



You will use the User Manager extensively in course 4-4913: Process Modeling for Process Platform.

28. Select Users - Roles as the mapping and click **Show All**.

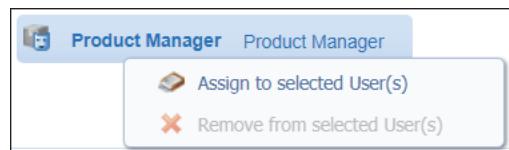
The left list will display the set of users and the right list will display the list of roles, of which the Product Manager is one available role.

29. From the list of users, select user Anna Lyst (analyst@training.local).

A number of roles will be highlighted on the right. This is the list of roles currently assigned to user analyst.

30. From the list of available roles, right-click the Product Manager role and select **Assign to selected User(s)**.

Figure 7-13:
Assign Product Manager
role to analyst user



There is no need to save your changes with the User Manager application. Once a role is assigned to a user, it is automatically assigned. You do not need to save, validate, or publish your changes.

31. Close the User Manager application.
32. Return to Process Experience and refresh the page. You should have access to the Product entity and its list now.

After a role has been added to Process Platform, you may want to synchronize that role back to OTDS. For this direction of synchronization, you must use the OTDS configurator.



Roles must be published before they can be synchronized to OTDS.



One-way configuration to OTDS can only be accomplished from a command line.



Use OTDS configurator to push new roles to OTDS

1. Click Start > Run.
2. Enter cmd.exe.
3. Switch directories to the Process Platform runtime directory:
cd C:\Program Files\OpenText\ProcessPlatform\Development\bin
4. Run the OTDS configurator.
otdsconfigurator.cmd

```
Administrator: C:\Windows\system32\cmd.exe - otdsconfigurator.cmd
c:\Program Files\OpenText>cd "ProcessPlatform\Development\bin"
c:\Program Files\OpenText\ProcessPlatform\Development\bin>otdsconfigurator.cmd
Checking package role relations for removal from OTDS...
14 package roles checked.
Checking package roles for updating or inserting into OTDS...
-
```

Figure 7-14:
OTDS Configurator

5. After the configurator has completed running, open a browser window to the OTDS Admin page.
<http://localhost:8080/otds-admin>
6. Log in as the OTDS administrator **otadmin@otds.admin**, password **opentext**.
7. Click **Partitions**.
8. Click Actions at the end of the Process Platform Roles row, and select View Members.

9. Select the **Groups** tab.

These groups represent roles in Process Platform.

10. Verify that your *Product Manager* role has been inserted.

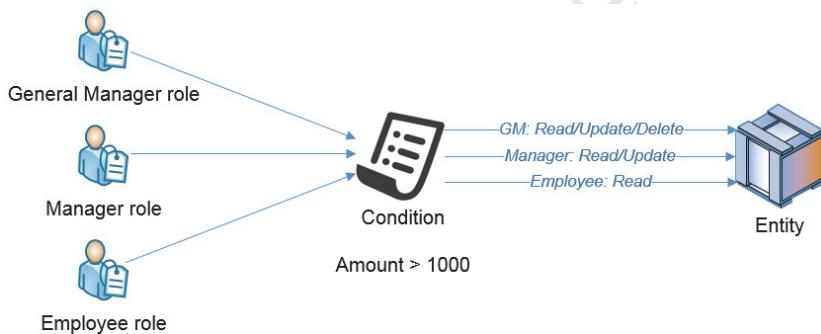
Figure 7-15:
Role synchronized to OTDS

	Group Name	Display Name
☐	Cordys@Work#Administrator	Cordys@Work
☐	Cordys@Work#Analyst	Cordys@Work
☐	Cordys@Work#Deployer	Cordys@Work
☐	Cordys@Work#Developer	Cordys@Work
☐	Northwind Warehouse Application#Product Manager	Northwind Wa
☐	OpenText Entity Assignee#Entity Assignee User	OpenText Er
☐	OpenText Entity Deadline#EntityDeadlineExecutor	OpenText En

Add optional conditions to an ACL Restrictions on entities can be rule-based. These are described as **ACL conditions**. The conditions are built in exactly the same way with the same editor as conditions for an entity's rule. Permission is granted to the entity only when the condition is satisfied.

Conditions are reusable on a per-entity basis: when a condition is added, it can be reapplied for different roles so as to allow different access for different users.

Figure 7-16:
An example of a security condition reused across different roles



In the following example, you will use conditions to restrict access to products in the Seafood category.



Add a condition to limit access to an entity

1. Open the Product entity from the Warehouse Application if it is not already open.
2. Select the Security building block.
3. Click **Configure**.
4. In the Conditions section of the security editor, click the add (+) button.

Figure 7-17:
Security condition editor

The screenshot shows the 'Condition' dialog box. At the top, it says 'Condition'. Below that is a 'Name' field containing 'UntitledCondition'. Underneath are tabs for 'Basic' and 'Advanced', with 'Basic' selected. A dropdown menu below the tabs shows 'If all of the following are true'. The main area is currently empty.

5. Set the name of the condition to *restrictSeafood*.
-
- The name of a security condition must not contain spaces and must begin with a letter or underscore.
-
6. Set the inclusivity to *all*.
 7. Set the condition to read: **Category not equal to Seafood**.
 8. Click **OK**.
 9. Select the **Product Manager** role in the security editor.
 10. Adjacent to the **Update** option on the **UnitPrice** property, look for and click the filter icon to apply a condition on the property.

Figure 7-18:
Select a condition for property permission

The screenshot shows a table titled 'Properties' for the 'UnitPrice' column. The table has three columns: 'Properties', 'Read', and 'Update'. The 'Update' checkboxes for 'ProductName', 'CategoryID', 'QuantityPerUnit', and 'UnitPrice' are checked. The 'Update' checkbox for 'UnitsInStock' is also checked. The 'Update' checkbox for 'UnitsOnOrder' and 'ReorderLevel' is checked. The 'Update' checkbox for 'UnitPrice' is checked. A circular callout highlights the filter icon next to the 'Update' checkbox for 'UnitPrice'. A tooltip says 'Select a Condition for this Permission'.

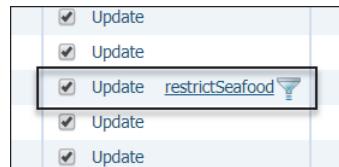
Properties			
ProductName	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
CategoryID	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
QuantityPerUnit	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
UnitPrice	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
UnitsInStock	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	Select a Condition for this Permission
UnitsOnOrder	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	
ReorderLevel	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Update	

11. Select the *restrictSeafood* condition.

Figure 7-19:
Apply a condition



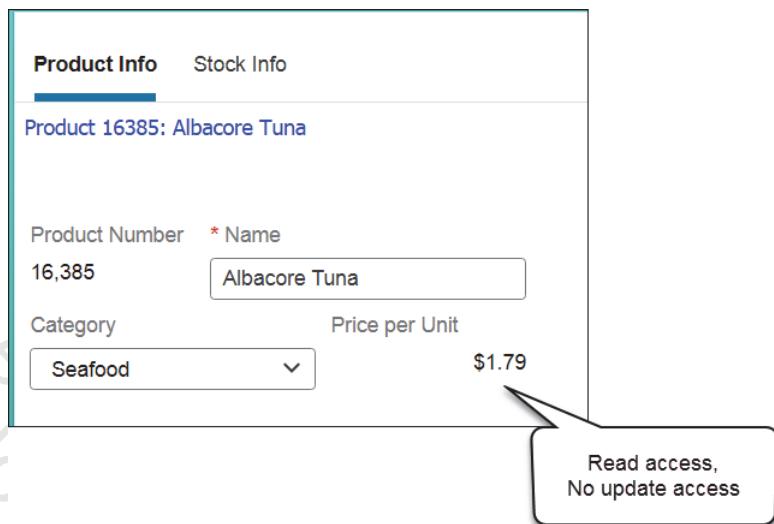
Figure 7-20:
Condition applied to a permission



12. Save the security configuration and close the editor.
13. Save and close the Product entity.
14. Validate and publish your changes to Process Experience.
15. Open Process Experience in a separate browser window or refresh your Process Experience page.
You should be logged in to Process Experience as the analyst user by virtue of OTDS single sign-on.
Remember that the Product Manager role was associated to the analyst user in the User Manager application in Process Platform.
16. Select the Product Catalog list from the home page.
17. Add a product to the catalog with the Seafood category.

If your condition was built correctly, you should not be able to edit the price of an item in the Seafood category.

Figure 7-21:
Condition applied in
Process Experience



Summary

Having completed this chapter, you should be able to:

- Add business-relevant data to the identity package in Process Experience
- Extend an existing solution to include the identity package
- Build hierarchical and “flat” organizational models using the identity package
- Create a team worklist using the identity package
- Audit the creator and users who have modified an entity using Tracking
- Add permission-based security to an entity
- Write and apply a condition to a security permission

Exercises

Identity package

1. Replace the Country property in the Customer entity with a relationship to the Country entity in the identity package.

Reminders:

- Use the relationship to browse the list of countries in the entity package.
- Consider a create option for adding new countries.
- Migrate your data to the new field before deleting the old property.
- Update the appropriate form(s). Consider the child entities of Customer, too.
- Update the appropriate list(s).

Security building block

1. Add an Account Manager role to your Process Platform solution. Publish the role, then assign the Account Manager role to the analyst user using the User Manager application.

2. Add the Security building block to the Customer entity. The Account Manager has full access to the entity.

3. Add a security condition to the Customer ACL: the Account Manager does not have access to the address, city, region, postal code, or country relationship if the customer client code begins with 'BE'.

Note: you will need the function `startsWith(arg)` to complete this challenge.

Test your work by creating a business customer with a client code starting with 'BE'.

4. Use the OTDS Push Configurator tool from the command line to synchronize the Account Manager role to OTDS.

5. PerishableProduct, because it is a subtype of Product, inherits its Security configuration from Product. Open the PerishableProduct entity and override (replace) the inherited Security configuration. Grant read and update access to the properties and use access to the list(s) specific to the PerishableProduct entity.

6. Commit your work to the repository with the description: "Chapter 7: Identity Package".

8. Entity lifecycle

Objectives

On completion of this chapter, participants should be able to:

- Describe the Lifecycle building block
- Add the Lifecycle building block to an entity
- Add building blocks to the task list in order to support lifecycle
- Model a lifecycle in the editor
- Add the Activity flow building block to an entity
- Add building blocks to the task list in order to support an activity flow
- Model an activity flow in the editor
- Test items in a lifecycle and an activity flow

Overview

Every item which is created from an entity exists in different stages from creation to archival: these stages describe the item's *lifecycle*. As the item progresses through its lifecycle, its properties may change in relevance. The Lifecycle building block captures the different stages of an item and presents them to users with different layouts. In this chapter, you will learn how to build a lifecycle and attach different layouts to it in order to capture an item from its inception.

An *activity flow* is similar to a lifecycle in some regards, but its design is simpler and does not presuppose any prior knowledge of business process management diagrams. Activity flows can be used in place of a lifecycle when the flow is straightforward.

What you will build in this chapter

- Lifecycle building block: you will add a lifecycle to your entities to manage their states from inception onward. This will also necessitate adding other building blocks, such as layouts, forms, and so on.
- Activity flow building block: similar to the lifecycle, but a more basic construction.

Timing

Lecture: 90-105 minutes

Exercises: 120-135 minutes

References

More information about this topic is available:

- Process Suite Community ([> Community > OpenText Process Suite Community](http://www.opentext.com))
- Process Platform 16.3 Entity Modeling Guide

About lifecycle

As previously mentioned, the lifecycle describes the progression of an item from its inception to its eventual end (e.g., archival, deletion, etc.). As the item progresses through its lifecycle, different properties of the item will take on different relevance. For example, when an order is first created, the order date and customer may be the most relevant properties. While an order is being fulfilled, however, the order date becomes less relevant than the required date. These properties are captured with different forms on different layouts. The layouts are then attached to different phases of the lifecycle so as to always display the most relevant data to the relevant users at the appropriate time. In this way, you will have easy access to the data which is most relevant to you at each stage of the item's lifecycle.

The Lifecycle building block is analogous to a case model in Process Platform. Nonetheless, a lifecycle can be used to model simple case-based solutions. Case-based solutions tend to be knowledge driven, unstructured, and collaborative. By comparison, process-based solutions are more structured, standardized, and task driven. Process-based solutions are built exclusively in Process Platform, independent of entities. The Lifecycle building block can be used to represent a knowledge-driven approach, while simultaneously depending upon and triggering business process models in the course of the lifecycle's evolution.

Creating a new item is typically a trigger for the lifecycle. Every item, whether it includes a lifecycle or not, has a pre-defined descriptive state, such as 'created' or 'modified', which the process engine will assign to it automatically. Whenever a new instance of a lifecycle has been attached to an entity, then a new instance of the lifecycle model will be connected to a new item. If that new item is in the 'created' state (all new items are), then the item is pushed into the initial state of the lifecycle.

Elements of a lifecycle A lifecycle can be depicted as the evolution of an item from inception to archival, as the following example for the Customer entity suggests:



This type of diagram, commonly referred to as a *state diagram*, is an effective means for describing entity lifecycles, and it presents significant insight into the design of the Lifecycle building block:

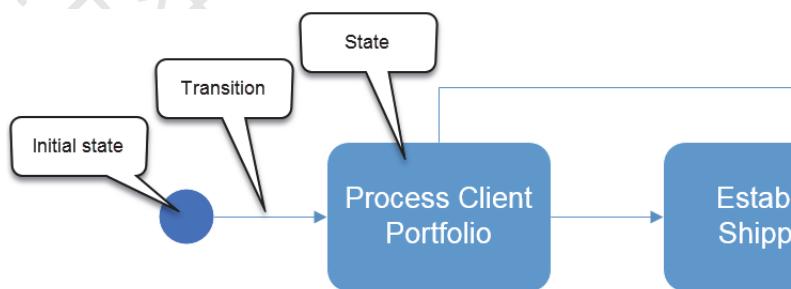
- Lifecycles are built with constructs common to, and consistent with, state diagrams.
- Lifecycles are built similarly to cases with Process Platform.
- Lifecycles are built with the same basic tools used to build processes in Process Platform.



It is a recommended practice to draw state diagrams of entity lifecycles before modeling them in Process Platform.

The preceding diagram presents the essential elements of a state diagram which apply to the Lifecycle building block in Process Platform:

Figure 8-2:
Elements of state
diagram/lifecycle



Initial and final state Every diagram, just like every lifecycle, should have a clearly-defined start and end point. These are the *initial* and *final states*. Every model must have a start point (the *initial state*) and an ending (the *final state*). In a lifecycle, they only serve a figurative purpose: they are not modeled into concrete objects. They do, however, have significance in terms of processing nested lifecycles, as you will learn later in this chapter.

States and activities Every state diagram will contain a number of states. The states represent the abstract phases, or stages, of the lifecycle. In each state, there will be one or more activities which can be performed, relevant to the state. In the example above, while processing the client portfolio, an account team may issue a welcome package to the new customer while the client team is entering relevant shipping data into Process Experience. As you will learn later in this chapter, it is possible to nest another state diagram inside a state. Knowing an item's current state is useful to determine the exact phase that the item exists in at a given point in time. This can be useful when building rules: an item's state can be composed in a rule's condition.

When designing a state, you can choose the activity which will be executed. You can either choose to execute a human activity or a business process activity. To execute a business process activity, you must first design a business process model in Process Platform.



You will learn how to build business processes in course 4-4913: Process Modeling for Process Platform.

A human activity is a task which must be completed by a person (i.e., a user in Process Experience). The activity is added as a task in a user's inbox in Process Experience. These activities are typically tasks, attributed by the targeted user's role, to complete a form or forms in a layout. By default, all activities in a lifecycle are human activities.

Transition Transitions describe the progression from one state to the next. An item may only progress from one state to another through a single transition. You cannot be in two different states simultaneously. Transitions are normally triggered by certain events which occur in the state of the item's lifecycle. For example, in the diagram above, once the customer has been created, and the account has been processed, you may choose to establish shipping rules for the customer or skip straight to reviewing their account with the finance team. Not all customers will necessarily have special shipping info.

In the Lifecycle building block, these events must be described in the state in order to trigger a transition. You can trigger a transition by either:

- Completing an activity: if the activity/activities in the state has/have been completed successfully, the item can be transitioned to the next state of its lifecycle.
- Firing a user-imposed action: a user can perform some specific action which will trigger the transition. This is normally performed by clicking a button in an action bar.
- Satisfying a condition: a user can set some specific property or change a targeted field which will trigger a rule to be executed. If the rule condition evaluates to true, you can force the transition from one state to the next. Conditions are built similarly to rules with the basic and advanced rule editor.

Activity connectors The above example presumes that every activity in every state will be performed successfully with no unusual circumstances. You can anticipate exceptional situations by including activity connectors in your lifecycle design. An activity connector will allow users to perform any of the following functions:

- Add an ad hoc task to an activity
- Add a relevant task based upon the state
- Plan future tasks when an item transitions
- Add related tasks to an existing task

Nested lifecycles It is possible to nest one lifecycle inside another. The nested lifecycle is added as an activity to a state in the parent lifecycle. When the item reaches this state in its lifecycle, it will begin executing the nested lifecycle. At this point, the parent lifecycle will be paused while the child lifecycle is executing. Only after the child lifecycle reaches a final state will control be released back to the parent so that you may transition to the following state.

Lifecycle building block

Like some of the solitary building blocks which you have modeled thus far, the Lifecycle building block does not require any basic information. It does not have a display name or name. Only one lifecycle may be added to an entity. When you add the Lifecycle building block, Process Platform creates a child entity, called *task list*, that contains the list of tasks. You may add other building blocks to that child entity, such as forms, layouts, and lists. Only those layouts which are built in the task list may be used in the lifecycle.

Since the activities which are added to states in the lifecycle may be presented to users as tasks, the child entity which is created is named the LifecycleTask. A task is work which a Process Experience user must perform (i.e., a human activity). Tasks are presented to users in the inbox (i.e., My Inbox). The inbox is an organized “to-do” list of work tasks which have been assigned to the user by their user name, their role, or their assigned team. The inbox is presented in Process Experience as a task panel in a layout.



Add a Lifecycle building block to the Customer entity

1. Open the Northwind Workspace.
2. Open the Customer entity in the Warehouse Application.
3. Click **Add > Lifecycle**.
4. Click **Add**.



When you add a lifecycle to an entity, two entries are added to the entity: Lifecycle and Task list.

Configuring the lifecycle tasks

The Lifecycle building block contains two components: its state diagram and its building blocks (in the task list child entity). After you add a lifecycle to an entity, you must add building blocks to its associated task list (e.g., forms, layouts, etc.) and you must configure the lifecycle's state diagram.



The lifecycle state diagram editor is actually a simplified case model diagram editor for Process Platform. You will build process models with a similar editor in course 4-4913: Process Modeling for Process Platform.

When a lifecycle is added, the child entity - task list - is also added automatically. The task list is provided an identity, a relationship to its parent entity, and a task. The task provides the connection between the lifecycle states and the generation of work task content in the user's inbox for the Task panel.



You must never delete the default properties of the task list: its identity, relationship, or task.

Because the task list is essentially a child entity, you may add other building blocks to it, including forms, layouts, action bars, and security. These building blocks can be associated to the item at different stages during its lifecycle.

Designing forms with lifecycle

When a Lifecycle building block is added to an entity, and the task list is added, new options are added to the form editor so that elements of the lifecycle may be included on the form. The form editor will present a set of components for both the parent entity and for the forms added in the task list.

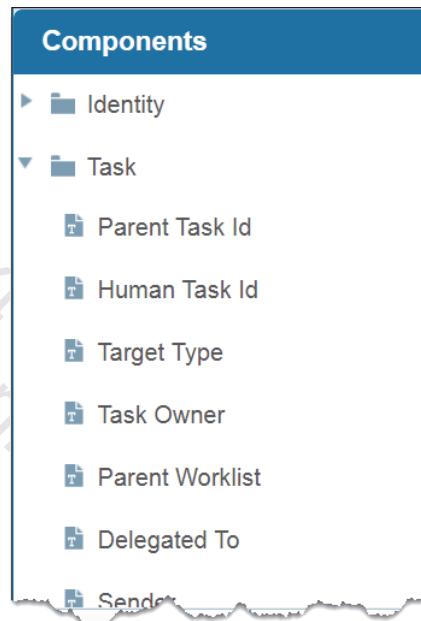
In the parent entity, elements of the lifecycle may be included in a form. These elements are common for any entity with an added lifecycle, regardless of the name of the entity. These elements include:

- Instance identifier: a unique identifier that refers to an instance of the lifecycle's state diagram model.
- State: the current state of the lifecycle model. An item can only be in one state at a time.
- Parent state: if the lifecycle model includes a nested model, and execution is currently in the nested model, then the state component will return the name of the state in the sub-model, and the parent state component will return the name of the immediate parent state which has been paused for execution of the nested model.

- Top-level state: a lifecycle model may have many nested models. This component will display the name of the topmost paused state in the lifecycle model.
- Last performed task: the activity which was performed most recently in the model.
- Last transition event: the most recent user event which triggered a transition in the model.

Elements from the lifecycle task list (below).

Figure 8-3:
Components from the Lifecycle building block added to a form



The task list includes many other components which may be added to forms. These components are related to the task engine in Process Platform, and therefore contain references to the task inbox and task generation for users. These components are common for all lifecycle task lists:

- Parent and human task IDs
- Task: properties of the generated task, which include (but are not limited to) the sender of the task, priority, due date, start date, task owner, subject, etc.
- Entity: all the components specific to the parent entity, its properties, relationships, title, forms and subforms, etc.
- Any new decorator building blocks after the lifecycle was added (e.g., any forms, layouts, titles, action bars, etc.).

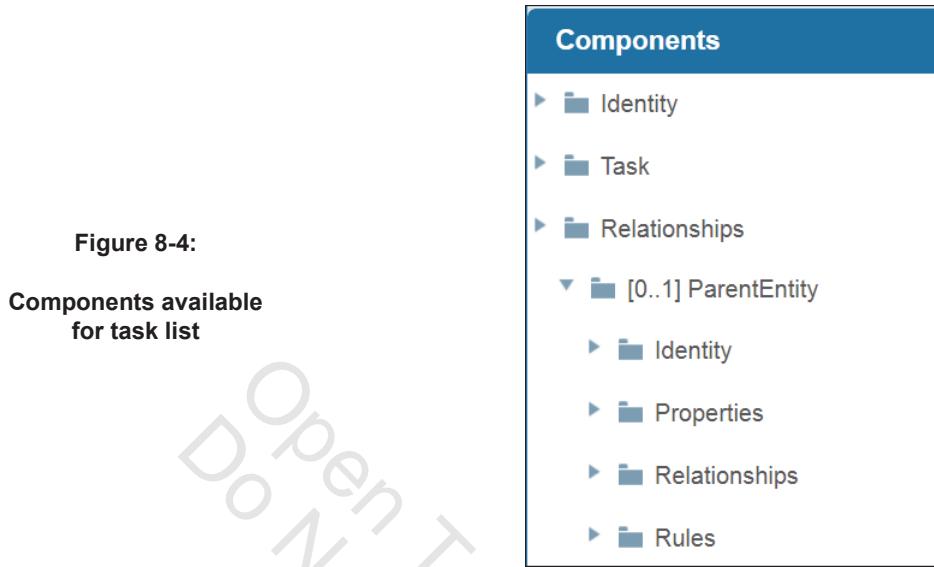


Figure 8-4:

Components available
for task list



Add building blocks to Customer task list

To this point, you have added several building blocks to entities. In this example, you will add many of the same building blocks to the task list which will complement the lifecycle.

1. Open the Customer building block if it is not open already.
2. Select the Task list building block.
3. Click **Configure task**.
4. Add an action bar.
 - a. Click **Add > Action bar**.
 - b. Set the display name to **Customer lifecycle action bar** and the name to **abarCustLifecycle**.
 - c. Click **Add**.
 - d. Click **Configure**.
 - e. Choose the following task buttons: **Claim, Revoke, Start, Stop, Pause, Resume, Skip, and Complete**.



Consider placing some or all of these buttons into button groups.

-
- f. Save and close the action bar.

You will discover a set of buttons which are made available for the tasks of the lifecycle. Each activity in each state of the lifecycle becomes a task which users must complete. The tasks, as you will learn, can be assigned to specific users, or to users based on their role, team, or organizational unit.

Tasks are distributed to an inbox: a virtual depository for receiving and processing tasks. The actions which a user can perform on a task are captured in the buttons of the action bar:

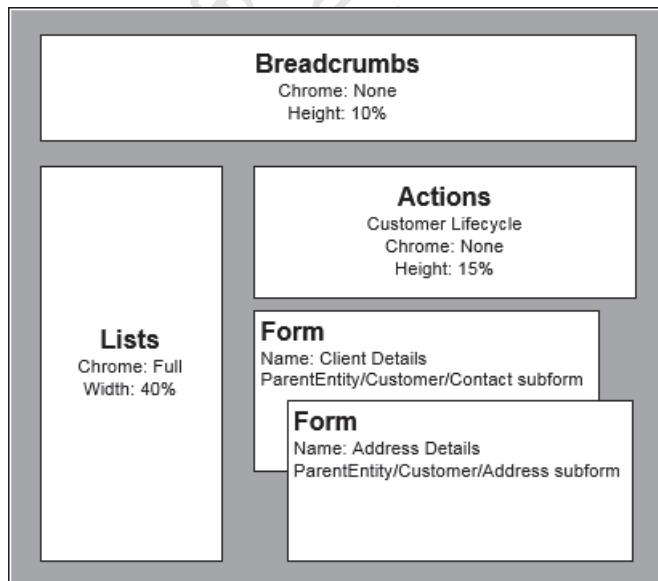
- Claim and Revoke: *claiming* a task implies that the user has accepted the task from the group inbox, without necessarily starting work on it. The task is removed from the inbox and placed in the claimant's personal inbox and marked with the user's identity. Once a task has been claimed, deadline timers may start. A user can return the task to the group inbox by *revoking* the task.
- Start, Stop, Pause, and Resume: a user *starts* a task to begin working on the task. The time spent working on the task is recorded along with the task. A user can *stop* work on the task (without revoking it), *pause* work, and *resume* work at a later date/time.
- Assign, Delegate, and Forward: a task claimant does not necessarily have to complete the work. The work of the task can be distributed to other users/groups by the claimant. When a task is *assigned* to a new user, the task is added to the new user's personal inbox. *Delegating* a task to a user or group makes the new user/group responsible for completing the task, but the task still remains in the delegate's personal inbox. *Forwarding* a task shares the task details with other users, but does not make them responsible for completing it, nor does it remove it from the claimant's inbox.
- Suspend and Skip: Users can *suspend* a task from executing. Only a work list manager or team lead can suspend the execution of a task in the work list or team folder. Users can also *skip* a task. You can skip tasks before they are completed if required. Once the task is skipped, it becomes a non-workable task.
- Modify dates: some tasks are given strict deadlines for starting and/or completing. For example, a task administrator may establish a deadline for the task to be claimed from the inbox and started. Similarly, a deadline may be established for the completion of the task. These dates can be modified in real-time, but the modifications (and user) are recorded for audit purposes.
- Add task: if there are several activities which are available in the state, a user can choose which activity/activities must also be completed in processing the state. The activities must be part of the state diagram.
- Complete: once work on the task has been finished, it is important to mark the task as *complete*. This issues an event to the state machine engine and will trigger the next activity or transition to the next state.

5. Add a layout named Process Account Layout.
 - a. Click **Add > Layout**.
 - b. Set the Display Name to **Process Account Portfolio**.
 - c. Set the Name to **ProcAccountLayout**.
 - d. Select **Full Layout**.
 - e. Click **Add**.
 - f. Click **Configure**.
 - g. Add a lists panel to the left and a form panel to the right, both with full chrome settings, and a breadcrumbs and an actions panel to the top with no chrome.
 - h. Select the customer lifecycle action bar for the actions panel.
 - i. Set the form name to *Client Details*.

The form you need for this panel is inherited from the parent entity: Customer.

- j. Scroll down in the form settings and expand **ParentEntity**.
- k. Select **Customer**.
- l. From the form options, choose the **Contact subform**.
- m. Layer another form panel on top of Client Details and add the Address subform, also inherited from the ParentEntity Customer, as Address Details.

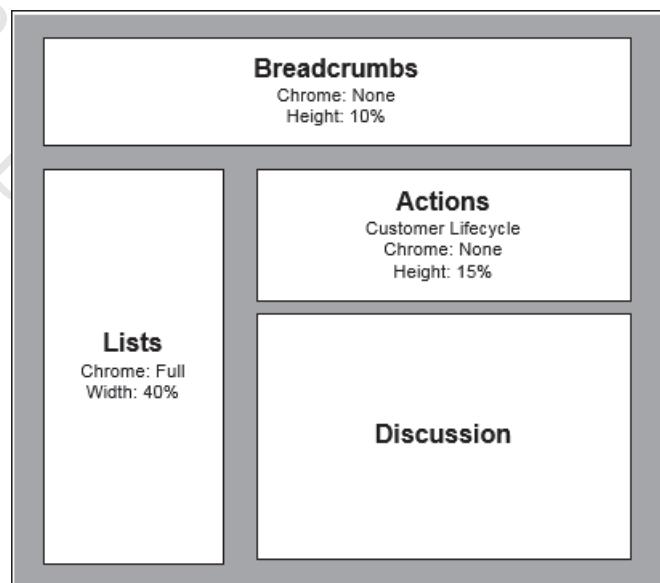
Figure 8-5:
Design of Process
Account Portfolio layout



- n. Save and close the layout.
6. Click **Add > Discussion**.

You will include a discussion about a state in the lifecycle which is independent from the entity's discussion.

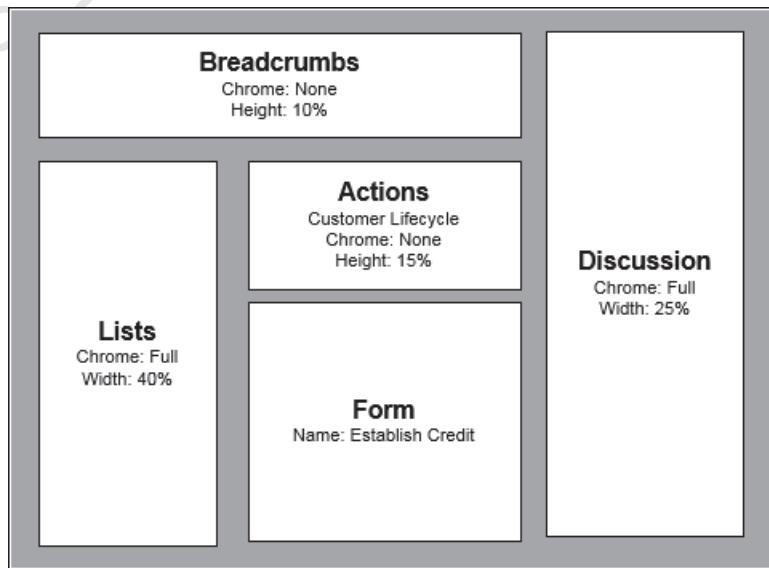
7. Click **Add**.
8. Add a layout named *Establish Shipping Layout*.
 - a. Click **Add > Layout**.
 - b. Set the Display Name to **Establish Shipping** and Name to **EstablishShippingLayout**.
 - c. Select **Full Layout**.
 - d. Click **Add**.
 - e. Click **Configure**.
 - f. Add an actions and breadcrumbs panel to the top, a lists panel to the left, and a discussion panel to the right.
 - g. Select the customer lifecycle action bar for the actions panel.



- Figure 8-6:**
Design of the Establish
Shipping layout
- h. Save and close the layout.
 9. Add a simple form and layout.
 - a. Click **Add > Property**.
 - b. Set the display name to **Billing Type** and name to **BillingType**.
 - c. Set the property type to Boolean.
 - d. Set the display name of the true value to "Pre-Paid", the false value to "Invoice", and the none value to "COD".
 - e. Click **Add**.
 - f. Click **Add > Form**.
 - g. Set the display name to **Establish Credit** and the name to **formEstablishCredit**.
 - h. Click **Add**.
 - i. Click **Configure**.

- j. Drag the customer's ID (Relationships > [0..1] ParentEntity > Identity > Id), client code, and billing type fields to the form.
- k. Save and close the form.
- l. Click **Add > Layout**.
- m. Set the display name to **Establish Credit Layout** and name to **EstablishCreditLayout**.
- n. Select **Full Layout**.
- o. Click **Add**.
- p. Click **Configure**.
- q. Add a lists panel to the left, a form panel to the center, a discussion panel to the right and an actions and breadcrumbs panel to the top.
- r. Select the customer lifecycle action bar for the actions panel.
- s. Set the form in the form panel to the Establish Credit form.

Figure 8-7:
Design of Establish Credit layout



- t. Save and close the layout.
10. Save and close the task list (LifecycleTask).
11. Save the Customer entity but do not close it.

The lifecycle model designer

The lifecycle model is designed with a special state model diagram editor in Process Platform.



There are many features of the state model editor which, for simplicity sake, cannot be addressed in this course. For this reason, there are many aspects of the editor which will not be described here. The sample which will be built in this chapter is a primitive model which one can build. Therefore, only the elements which you need to build the sample model will be presented. If you want to learn more about building models, you should consider course 4-4913: Process Modeling for Process Platform.

The state model editor is composed of a toolbox and a model canvas. The toolbox contains elements, called *constructs*, which may be added to a state (read: lifecycle) model. As the modeler, you can simply drag and drop the constructs from the toolbox onto the canvas. Some of the relevant constructs include:

- Initial State
- State
- Final State
- Activity
- State Entry: to signify the starting point inside a state
- Start Lifecycle: a start point if you choose to build a lifecycle without states (just activities)
- Close Lifecycle: an ending point for a lifecycle that does not include states
- Text: add some arbitrary, descriptive text to a diagram. The text has no functional purpose and is not used in run time.

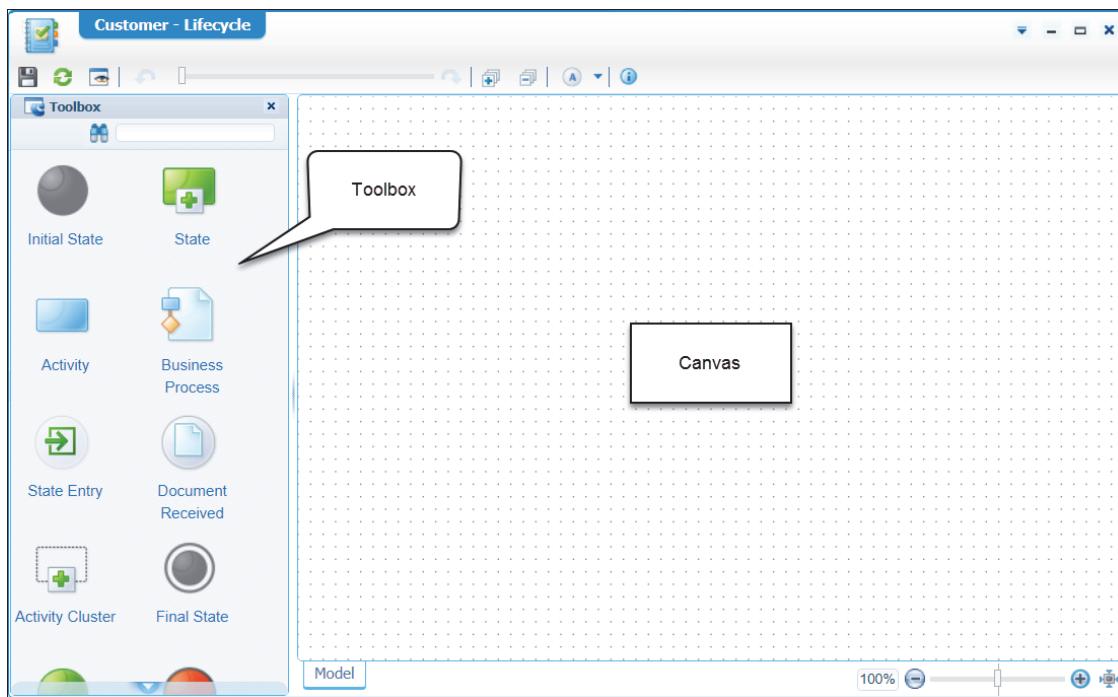


Figure 8-8: Lifecycle model editor

Designing a model In order to create a lifecycle model in the editor, drag constructs from the toolbox to the canvas. As you hover over constructs on the canvas, the context-sensitive graphical editor will offer you options for other, connecting constructs. For example, if you hover over a blank canvas, a set of constructs will appear which you can select to add to the canvas. If you add an Initial State construct, a set of constructs appear which may be connected to an Initial State (i.e., a transition, a state, or a text annotation). Finally, if you add a state, a set of constructs appear which may be connected to the state (i.e., a transition, another state, a final state, or a text annotation). Simply selecting these pop-up constructs is a quick and efficient method for designing a lifecycle model.

Figure 8-9:
Constructs available on a
blank canvas



Figure 8-10:
Constructs available
from an initial state

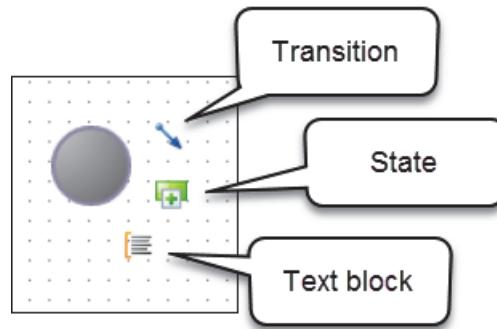
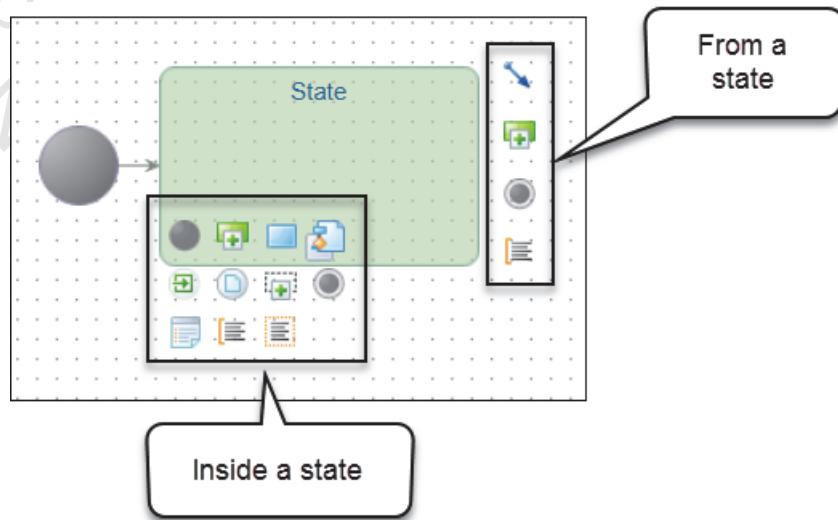
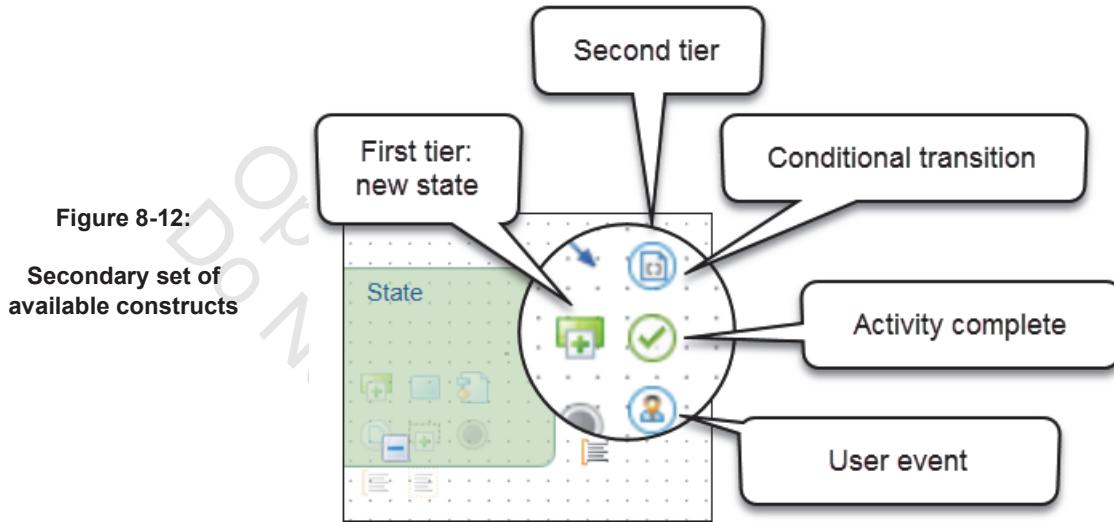


Figure 8-11:
Constructs available
inside and outside a state



Alternatively, you can simply drag constructs to the canvas and connect them together (*wire them*) with transition arrows.

As well, some of the pop-up constructs will have a subset of other, related constructs. For example, if you connect a state to another state, a subset of constructs will appear which will assist in determining the type of transition between the two states: a conditional event, an activity completion event, or a user-type event. Hover over each construct to determine its name.



Design a lifecycle state model diagram

1. Open the Customer entity if it is not open already.
2. Select **Lifecycle**.
3. Click **Configure**.

The Lifecycle Model Editor is displayed.

4. Drag an *Initial State* construct onto the canvas.
5. Hover over the *initial state* construct until the pop-up menu displays.
6. Select **Add State**.

The state is automatically connected to initial state with a transition arrow.

7. Change the name of the state to **Process Client Portfolio**.
8. Drag a *State Entry* construct and drop it inside the state.

The state entry construct is a cue to begin the activities in the state. The alternative is that the user must select activities manually.

9. Select and hover over state entry until the pop-up menu displays.
10. Select **Add Activity**.

The activity is automatically connected to the state entry with a transition arrow.



A letter (A) on the transition arrow is a message to the engine that the transition will occur automatically. Transitions may be *automatic*, *manual*, or *intermediate*.

11. Change the name of the activity to **Process Portfolio**.
12. Select the state and hover over it until the pop-up menu displays.
13. Hover over Add State without selecting it: a second-tier pop-up menu displays.
14. Select **Add User Event**.

Figure 8-13:
Add new state with user event transition



A second state is added to the diagram with a user-type transition.

15. Rename the second state to **Establish Shipping**.
16. Drag a State Entry construct and drop it into the Establish Shipping state.
17. Attach an activity to the new state entry construct.
18. Set the name of the activity to **Discuss Shipping**.
19. Hover over the shipping state until the pop-up menu displays.
20. Hover over the Add State icon without selecting it: a secondary pop-up menu displays.
21. Select **Add Activity Completion Event**.

The lifecycle automatically transitions to the next state after the activity is marked complete.

22. Rename the third state to **Establish Credit & Finance**.
23. Drag a State Entry construct and drop it in this state.
24. Attach an activity to the new state entry construct.
25. Set the name of the activity to **Choose Credit Option**.
26. Hover over the credit and finance state until the pop-up menu displays.
27. Hover over the Add Final State without selecting it: a secondary pop-up menu displays.
28. Select **Add Activity Completion Event**.

29. Select the Process Client Portfolio state and hover over it until the pop-up menu displays.
30. Select the Add State Transition > Add User Event arrow.
31. Connect the Transition arrow to the Establish Credit & Finance state.

This describes the situation of a Customer account which does not need to discuss shipping.

You may need to drag and reposition the new transition arrow in order to properly view the diagram.

Your diagram should resemble the following when complete:



Figure 8-14: Sample customer lifecycle model diagram (in progress)

32. Save the case model editor, but do not close it.

Attaching a layout to an activity After you have designed the lifecycle model and added building blocks to the task list, you must attach layouts from the task list, and any user events, to the lifecycle model. This step will configure the lifecycle model with any forms, layouts, rules, and action bars you have built.

In the lifecycle model editor, every construct has a set of properties. You can access the properties by right-clicking the construct and selecting properties or by pressing F8. The properties panel opens at the bottom of the model editor. Each construct has a number of properties, organized by tabbed panes.

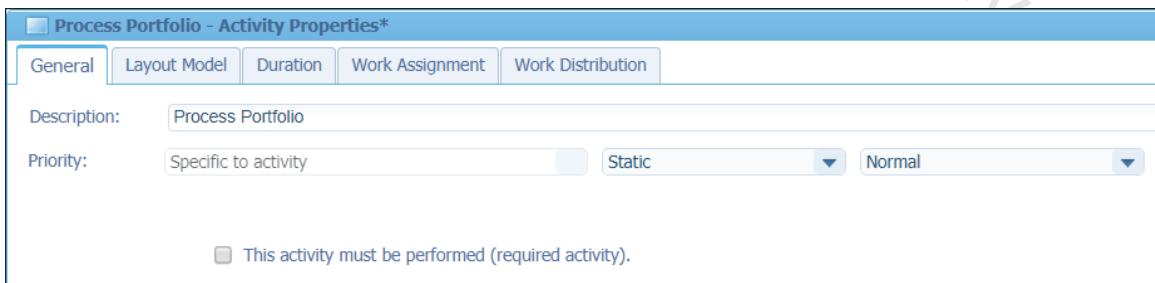


Figure 8-15: Properties panel in model editor

For your purposes, the most relevant tabbed panes include:

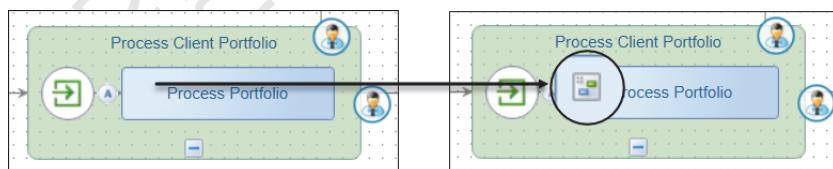
- General: this tabbed panel includes the name of the construct, a check box to mark the activity as required (in case there are many activities in a state), and the activity priority (which may be assigned statically or programmatically based upon the incoming message). If you choose the whole model, this panel contains the name of the root state.



State constructs will also include a field named the *Applicability Service*. The applicability service allows the user to choose the most applicable subset from the set of possible follow-up activities.

- Layout Model: this tabbed panel allows you to choose a layout to match an activity. It is only available on activity constructs. When a layout has been added to an activity, the icon on the model editor will change accordingly.

Figure 8-16:
Layout attached to
activity in state diagram



- Duration: this tabbed panel allows you to specify a deadline for completing the activity or state. The deadline may be a static amount of time (e.g., 3 days, 8 hours, etc.) or a programmatic value read from a message. You can also apply a business calendar to the calculation.



This is not the same as the Deadline building block, which will be discussed in the following chapter. You will build business calendars in course 4-4913: Process Modeling for Process Platform.

- Work Assignment: this tabbed panel allows you to choose the party responsible for completing the activity. You may assign the work to a user, role, organizational unit, or worklist. The assignment may be a static or programmatic based upon a message.



This is the Worklist you created in the previous chapter, but not the same as the Assignee building block, which will be discussed in the following chapter.

- Work Distribution: this tabbed panel allows you to add optional algorithms to dispatch the work associated to the activity. By default, the work is distributed based upon settings inherited from the lifecycle. You may, however, add a static or programmatic dispatch algorithm.



A dispatch algorithm is a document which can be prepared in Process Platform, just as entities, roles, or homepage layouts are documents.



Programmatic functions, such as dispatch algorithms or programmatic work assignments, are typically written with expression language.



You must have the model editor open to begin the following example, you must have designed the model, and you must have added the decorator building blocks to the Lifecycle.



Attach layouts to activities in a lifecycle model

1. Select the *Process Portfolio* activity from the model.
2. Right-click the activity and select **Properties**.
3. Select the **Layout Model** tab.
4. Click the browse tool to lookup a layout.



The only layouts available are the ones which you added to the task list.

5. Select the *ProcAccountLayout* and click **OK**.

The activity icon will change to reflect that the Layout has been attached.

6. Select the *Discuss Shipping* activity.
7. In the *Properties* panel at the bottom, select the *Layout Model* tab.
8. Use the lookup button to attach the *Establish Shipping* layout to the activity.
9. Attach the *Establish Credit* layout to the *Choose Credit* activity.
10. Select the transition between the *Process Client Portfolio* and *Establish Shipping* states.
11. In the *Properties* panel at the bottom, set the description to **Discuss Shipping** and the Event name to **Shipping**.

12. Select the transition between the *Process Client Portfolio* and *Establish Credit* states.
13. In the Properties panel at the bottom, set the description to **Establish Credit** and the Event name to **Credit**.
14. Save and close the model editor.

Setting transition events As you have learned when building the state model diagram, there are three ways in which a state may transition to another. The state model diagram is referred to as an “event-driven model”; this means that in order to effectuate a transition from one state to the next, an event must occur. The possible events are:

1. Activity Completed event (✓). Once the activity has been completed, the lifecycle automatically transitions to the next state in the model.
2. Conditional event (⌚). The state and/or item must satisfy a specific condition (written with the rule editor in either basic or advanced mode) in order to transition to the next state.

When working with expression language in a lifecycle, you can write language which will check values of the item, or check values of the task list.

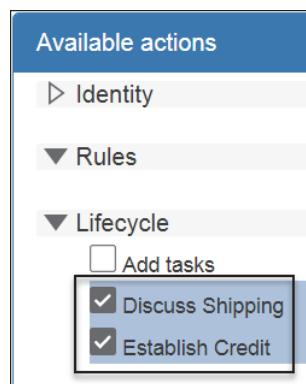
Some task-level properties you may choose to query include the task duration, priority, assigned team, and so on. If you are querying the task in the lifecycle, you should use the *Lifecycle* keyword. For example:

```
item.Lifecycle.CurrentState
```

3. User event (👤). The user must click a button in order to effectuate a transition from one state to the next. The buttons are placed on an action bar building block.

When you choose the user event-type of transition, you must specify the name of the event and an arbitrary description. The event name is used to create a button in the action bar editor (under the lifecycle group) of the parent entity.

User event transitions available as buttons in action bar editor





Configure user event transitions in the model

1. Open the Customer entity from the Warehouse Application if it is not open already.
2. Select the Task list.
3. Click **Configure task**.
4. Select the customer lifecycle action bar.
5. Click **Configure**.
6. Expand the **ParentEntity**.
7. Select **Discuss Shipping** and **Establish Credit** under the Lifecycle group.
8. Save and close the action bar.



The action panel in the layouts should already use the action bar and, consequently, will use the new lifecycle buttons.

9. Save and close the task list.
10. Select Security and click **Configure**.
11. In the Permissions section of the security editor, grant permission for the Account Manager to execute both the Discuss Shipping and Establish Credit transitions.
Select the Account Manager role and select **Execute** for Discuss Shipping and Establish Credit. Add tasks is optional.
12. Save and close the security configuration editor.
13. Save and close the Customer entity.
14. Validate and publish the Warehouse Application to the organization.



Test the customer lifecycle

1. Open Process Experience in a separate window and log in as user **analyst** (if necessary).
If you are already logged in to Process Experience, refresh the browser to get the latest build of your solution.
2. Create a new business customer item with some sample data.



You must build a new customer: any existing customer items which you may have will use the older version of customer from your solution (i.e., the one without lifecycle).

Creating the item will trigger the lifecycle. The lifecycle model you designed places the new item in the Process Customer Portfolio state and automatically triggers (via the state entry construct) the corresponding activity. This activity is now considered to be a *task*. Tasks are available in the *inbox*.

3. On the home page, click **My Inbox > All Tasks**.

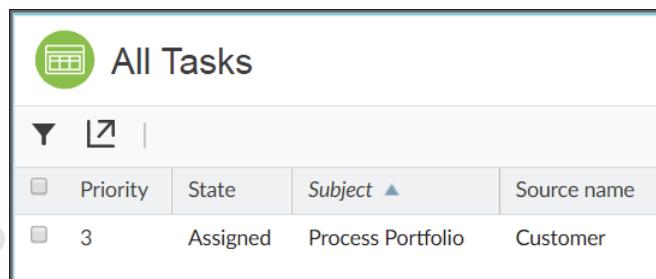


Figure 8-18:
New task in the inbox

4. Select the task and click **Open**.

The layout should be the process customer layout you created earlier, with action buttons consistent with processing the lifecycle.

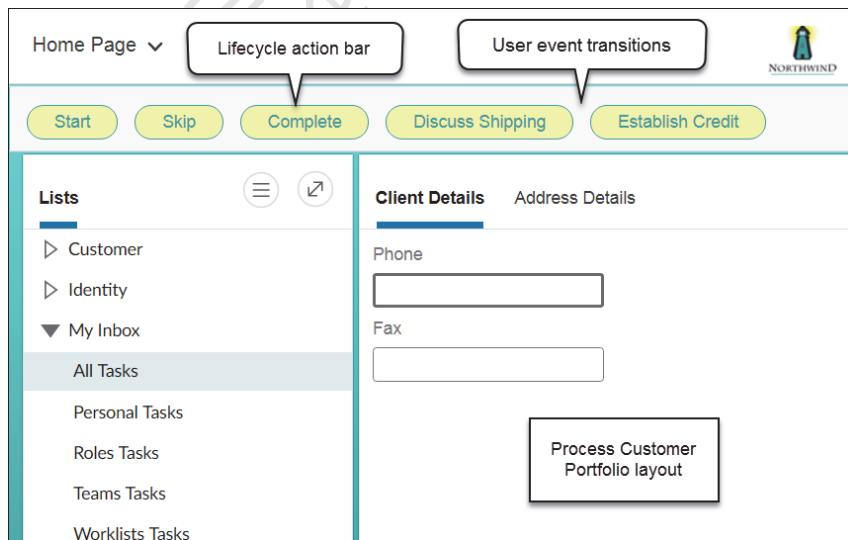


Figure 8-19:
Testing lifecycle with
new customer

5. Click **Start**.
6. Add some values to complete the customer details, such as a phone number and address.
7. Click **Complete**.

Once this task is complete, you need to select the state to which the item will transition.

8. Click **Discuss Shipping**.

9. Return to the home page and select **My Inbox > All Tasks**.

The item should occur in the entry activity of the second state: Discuss Shipping.

10. Select the task and click **Open**.

11. Examine the form and the layout: they should reflect the shipping layout you created earlier.

You do not need to start a discussion.

12. Click **Complete**.

13. Return to the home page.

14. Click **My Inbox > All Tasks**.

The lifecycle should have transitioned automatically to the next state: Establish Credit.

15. Select the item and click **Open**.

The layout should reflect the current state: choosing the credit option. You should be presented with the pre-paid/invoice form.

16. Select the Pre-paid option and click **Complete**.

17. Return to the home page.

The customer lifecycle is now complete, and the item has transitioned to the final state. There are no tasks remaining in the inbox.



You may want to create another customer item and test the establish credit transition. When the transition is followed, the item will skip the shipping discussion.

Follow a lifecycle in the entity

You can track the progress of an item in the lifecycle using a layout. A special panel, *Lifecycle Progress*, is added to the parent entity layout design when the lifecycle is added to the entity.



Track the progress of a lifecycle

1. Open the Business entity in the Warehouse Application.

Remember that Business is subclassed from Customer entity.

2. Select the workWithBusiness layout and click **Configure**.

3. Drag the Lifecycle Progress panel and drop it on the top of the layout, so that it may be visible across all panels.

4. Set the height of the Lifecycle Progress panel to approximately 15%.

5. Save and close the layout.

6. Save and close the Business entity.

7. Validate and publish the Warehouse Application.
8. Refresh Process Experience.
9. Create a new Business customer in Process Experience.
10. Select Customer > Client List.
11. Select the new customer from the list and click Open.

The new customer is in the first state of the lifecycle.

Figure 8-20:
First state of the
Customer lifecycle in
Lifecycle Progress panel

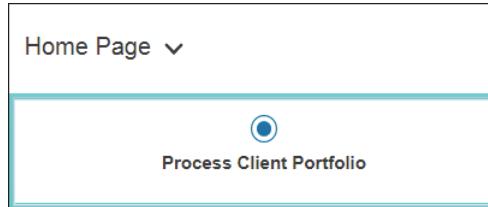
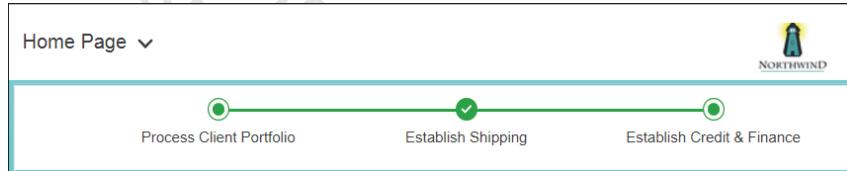


Figure 8-21:
Lifecycle complete for
Customer item



The Lifecycle Progress panel is only available in layouts for the parent entity, not in layouts for the task list.

Activity flow building block

The lifecycle construct is capable of handling complex business flows with several activities available at every stage which can be added manually in real-time depending upon user experience and circumstances.

As you have learned, there are several steps involved in creating a lifecycle. To simplify, these are: adding the building block, creating building blocks in the associated task list, and modeling the lifecycle in the editor.

Some lifecycles, however, can be very simplistic with a limited number of activities and automatic transitions from one activity to the next. In these circumstances, you may choose to use the activity flow building block instead of the lifecycle. An activity flow is like a lifecycle or a business process model, but does not require any prior knowledge of BPM design.

Components of the activity flow The activity flow is much like the lifecycle in that it is composed of two parts: the task list and the model. Activity flows and lifecycles use the same task list, in fact. The real difference between a lifecycle and an activity flow is the model. Whereas the model for a lifecycle can be a complex wiring of states, activities, and transitions, activity flows are straightforward progressions. Activities in the activity flow are, in truth, governed through layouts. You can think of an activity flow as a progression of role-based layouts.

Similar to lifecycles, when the activity flow is added, it also adds a task list which will contain all the forms, layouts, and other building blocks necessary to support the flow. Unlike lifecycles, however, you can add many activity flows to an entity. These activity flows can be triggered in one of two ways:

- When an instance of the item is created. This is similar to a lifecycle. When an item is created in an entity which contains a lifecycle, the lifecycle is instantiated and evaluated whenever a new item is created.
- Based upon a rule. The condition of the rule is written with the rule editor in basic or advanced mode with expression language and, once that condition evaluates to true, the activity flow is started.



The rule condition must be written with the same style editor used with rules. You cannot refer to a pre-written rule or a business process.



Add an activity flow to the Product entity

1. Open the Product entity from the Warehouse Application.
2. Click **Add > Activity flow**.
3. Set the display name to **New Product Processing** and the Name to **processNewProducts**.
4. Click **Add**.

The activity flow and its associated task list are both added to the entity.

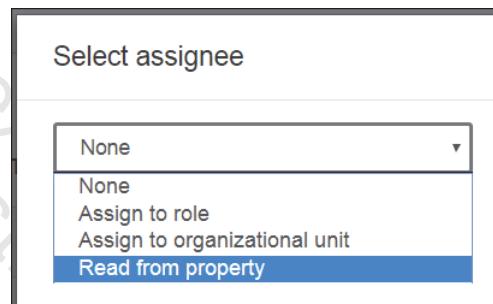
5. Save the entity but do not close it.

Activity flow editor

The activity flow editor is a table of named activities. Each activity (task) in the activity flow can be given the following properties:

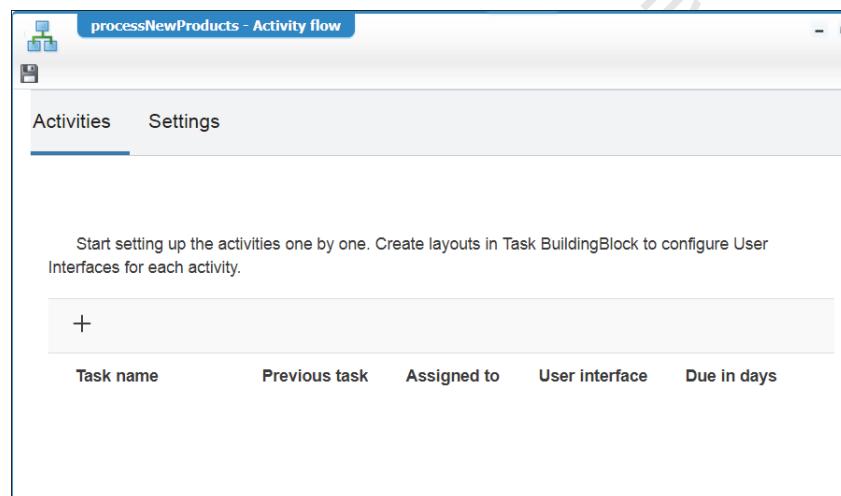
- Task name: an arbitrary description of the activity in the flow. It is recommended that these names are succinct; they will be used to choose the transition for the next activity.
- Previous task: the task name of the task which precedes it. In this way, you are creating an ad hoc order to the tasks.
- Assigned to: activities in a flow can be assigned or not. If this value is set to None, then the activity is available for everyone in the organization. Otherwise, the task can be assigned to a role or an organizational unit. The task may also be assigned programmatically, by reading a value from a specific property and making an assignment based upon the property.

Figure 8-22:
Assignees for an activity flow's task



- User interface: the display name of a layout from the task list which will be used as the interface for the user assigned.
- Due in days: the integer number of days in which this task must be completed. You can only express this value in integers, and you can only express the value statically in days: you cannot add a programmatic value.

Figure 8-23:
Activity flow editor



The Settings page of the activity flow editor allows you to indicate if the activity flow is executed when an item is created, or if its execution is rule-based.

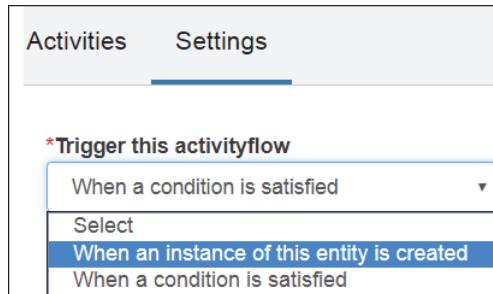


Figure 8-24:

Activity flow triggers

- The condition for triggering an activity flow can only be written in expression language. It does not have a basic mode and cannot be a BPM.



Add building blocks to the task list for the activity flow

1. Open the Product entity from the Warehouse Application if it is not already open.
2. Select Task list.
3. Click **Configure task**.
4. Add a new action bar.
 - a. Click **Add > Action bar**.
 - b. Set the display name to be **Flow Action Bar** and the name to **aflowBar**.
 - c. Click **Add**.
 - d. Click **Configure**.

You will notice that the action bar for activity flows has the same set of tasks as a lifecycle.

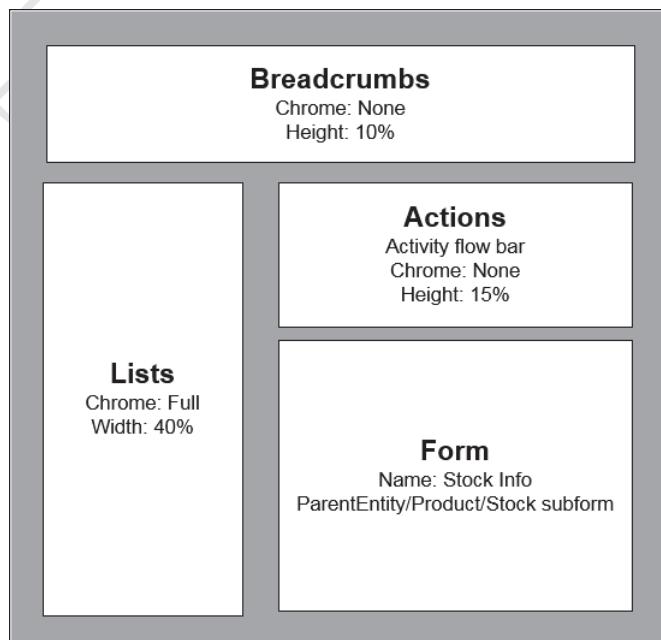
- e. Scroll to the bottom. The activity flow actions can be located there.
- f. Add the following buttons from the Task category: Claim, Revoke, Start, Stop, Skip and Complete.
- g. Save and close the action bar.



Activity flows offer their own button, *Initiate activity flow*, which you will discover under the ParentEntity/Activity flows heading. This is presented as a button for the purpose of manually triggering a flow.

5. Add a discussion building block to the task list.
 - a. Click **Add > Discussion**.
 - b. Click **Add**.
6. Add a layout for the first task.
 - a. Click **Add > Layout**.
 - b. Set the display name to **Stock Product** and the name to **aflowStockProduct**.
 - c. Select **Full Layout**.
 - d. Click **Add**.
 - e. Click **Configure**.
 - f. Add a breadcrumbs and actions panel to the top, a lists panel to the left, and a form to the right with the Path linked to **ParentEntity > Product > Stock** subform.

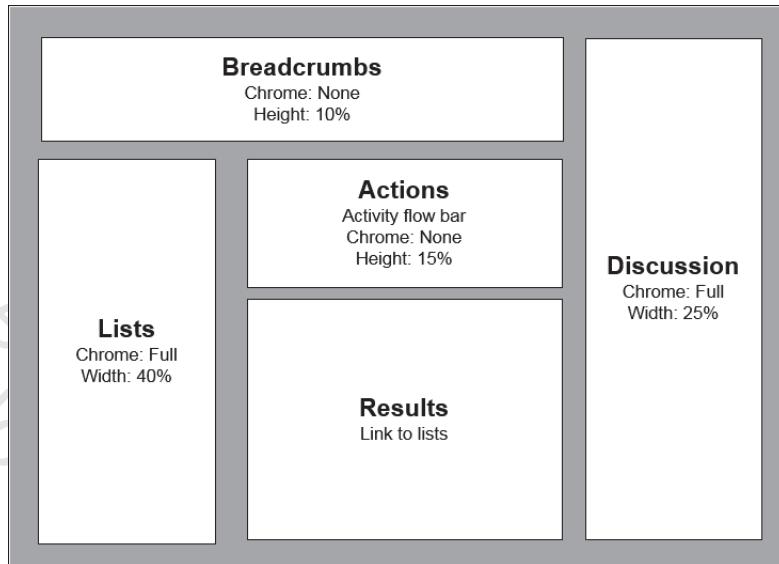
Figure 8-25:
Design of Stock Product layout



- g. Save and close the layout.
7. Add a layout for the second task.
 - a. Click **Add > Layout**.
 - b. Set the display name to **Product Awareness** and the name to **aflowProductAware**.
 - c. Select **Full Layout**.
 - d. Click **Add**.
 - e. Click **Configure**.

- f. Add a breadcrumbs and actions panel to the top, a lists panel to the left, a results panel to the center linked to the list panel, and a discussion panel to the right.

Figure 8-26:
Design of Product Awareness layout



- g. Save and close the layout.
8. Save and close the task list.
9. Do not close the Product entity.



Model the activity flow

1. Select the new product activity flow on the Product entity editor.
2. Click **Configure**.
3. Select the **Settings** tab.
4. Select **When an instance of this entity is created** as the trigger.
5. Select the **Activities** tab.
6. Add a new activity (i.e., click the '+') as a row to the flow grid.
7. Set the task name to **Place initial stock order**.
8. In the Assigned to column, click the browse button.
9. In the Select assignee dialog, select **Assign to role**.
10. Use the browse button to select the Product Manager role.
11. Click **Select**.
12. Click **Done**.
13. In the User interface column, click the browse button.
14. Select the **aflowStockProduct** layout and click **Done**.
15. Set the due date to be 5 (days).

16. Add a new row to the activity flow.
17. Set the following values to the new task:
 - Task name: Product awareness campaign
 - Previous task: Place initial stock order
 - Assigned to: None (blank)
 - User interface: *aflowProductAware*
 - Due in days: 10
18. Save and close the activity flow.
19. Select Security and click **Configure**.
20. Select the Product Manager role.
21. Enable **Execute** on Initiate activity flow, to grant permission for the product managers to manually execute an activity flow.
22. Save and close the security editor.
23. Save and close the Product entity.
24. Validate and publish the Warehouse Application.



Test an activity flow

1. Return to Process Experience and refresh the browser to collect the published changes.
2. Add a new product.
3. Add some default values for the product and click **Create**.
4. From the home page, select the **My Inbox > All Tasks**.
5. Select your new product and click **Claim**.
6. Open the product.

You should be presented with the first layout of the activity flow: stock product.

7. Set some values for the initial stock values.
8. Click **Complete**.
9. Return to the home page.
10. Locate the new product in the inbox.

It has progressed to the next activity in the flow.

11. Open the product.

You should be presented with the second layout of the flow: the product awareness campaign layout.

12. Click **Complete**.

The activity flow is complete and the new product will no longer be available in the inbox.

Summary

Having completed this chapter, you should be able to:

- Describe the Lifecycle building block
- Add the Lifecycle building block to an entity
- Add building blocks to the task list in order to support lifecycle
- Model a lifecycle in the editor
- Add the Activity flow building block to an entity
- Add building blocks to the task list in order to support an activity flow
- Model an activity flow in the editor
- Test items in a lifecycle and an activity flow

Exercises

1. Build a lifecycle on the Product entity which will support products being stocked in the system. Your lifecycle should include the following features:

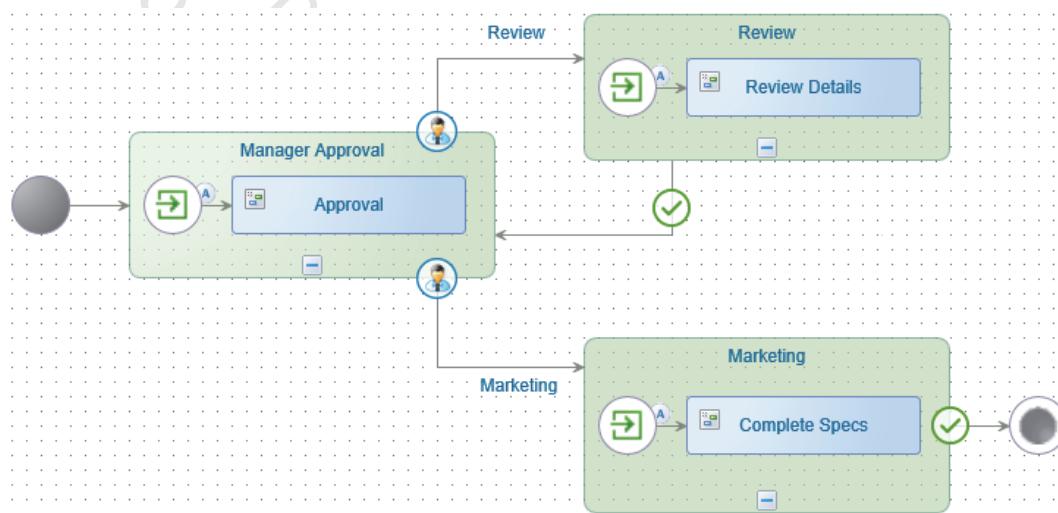
- Three layouts - Manager Approval, Product Review, and Marketing - for each activity in the state model diagram. You may design the layouts as you choose.

Hint: Use the breadcrumb and lifecycle progress panels in the parent layout to demonstrate navigation through the lifecycle.

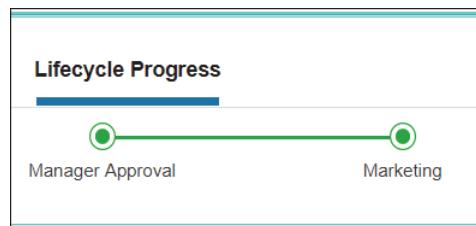
Hint: Use an actions panel in the layout that includes the lifecycle tasks (e.g., Claim, Complete, Suspend, etc.) so the user may mark the task.

Hint: update your security configuration to allow the lifecycle and its user event transitions.

- A lifecycle state model diagram which maps the activities to the layouts:



- Remember when testing, after the project has been published, you must create a new instance of the Product entity (i.e., one which uses the new lifecycle building block), and there will be two tasks for every new product in your inbox: one for the activity flow and one for the lifecycle.



2. Commit your work to the repository with the description: "Chapter 8: Lifecycle".

Open Text Internal Use Only
Do Not Distribute

9. Flow support

Objectives

On completion of this chapter, participants should be able to:

- Configure Process Platform to connect to an email server
- Add the Email building block to an entity to enable it for the email service
- Write message templates with the Email template building block
- Configure email in the Process Experience administration console
- Send email from an entity instance in Process Experience
- Impose a deadline on an entity using Deadline building block
- Distribute entity instances using the Assignee building block

Overview

When using flows such as lifecycle and activity flow in an entity solution, there are building blocks which you can add in order to support the flow. These building blocks are email, deadline, and assignee.

What you will build in this chapter

- Email building block: you will add email capabilities to your entities, which includes the ability to send and receive emails in a layout. Email must be properly configured, however, and generated emails must be designed from an email template (using the Email Template building block).
- Deadline flow building block: impose deadlines on the completion of entity instances in activity flows or lifecycles and decide how to treat overdue items.
- Assignee building block: distribute work to other users or groups of users using the Assignee building block.

Timing

Lecture: 75-90 minutes

Exercises: 75-90 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform 16.3 Entity Modeling Guide
- Regular expressions: <http://www.regular-expressions.info/>

The Email building blocks

Sending email from an entity solution requires configuration beforehand as well as setting up the email messages. Users in Process Experience can send, receive, reply to, and forward emails from an entity's instance.

Attachments which are added to or received from an email must be added through the Content building block.



You will use the Content building block in the following chapter.

Emails must be sent from pre-built email templates. Therefore, sending an email from an entity requires three major setup elements:

1. Configure Process Platform to communicate with the email server. This is a task which may be performed by a system administrator. Configuration must only be performed once. As soon as the configuration is established and the service container is running, then emails can be sent through the connection.
2. Add the Email building block. This building block must be added to the entity from which you want to send or receive emails, whether or not that email includes attachments (content).
3. Build an email template. This step requires adding the Email Template building block to an entity and building the message with text and expression language.

After this setup has been performed, you must also include the Email panel in a layout so that users in Process Experience can manage email from within your application.



You will learn how to add service groups and service containers in course 4-4913: Process Modeling for Process Platform.



Configure the email connector in Process Platform

1. Start the Mercury Mail server if it is not running already.

A shortcut to start the mail server is provided on the desktop. Double-click the shortcut to start the mail server.

Start Mercury mail server

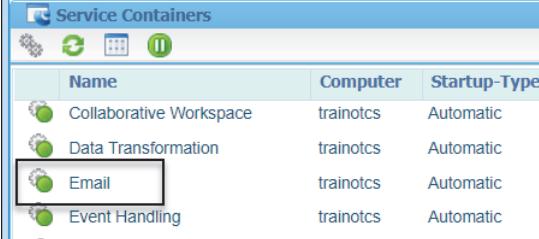


Figure 9-1:

2. In Process Platform, switch to the System organization.
3. Launch the System Resource Manager application from the application palette.
4. Right-click the Email service container in the list and select **Properties**.
5. Change the Startup Type from Manual to **Automatic**.
6. Select the **E-mail** tab.
7. Set the following configuration on the **Incoming** sub-tab:
 - Server Protocol: **pop3**
 - Host Name / IP address: **trainotcs**
 - Port number: **20110**
 - Set SSL: **false**
8. Set the following configuration on the **Outgoing** sub-tab:
 - Host Name / IP address: **trainotcs**
 - Port number: **2025**
 - Authentication required: **false**
 - HTML (Content Type): **false**
 - Set SSL: **false**
9. Save your changes.
10. When you see the confirmation dialog, select **Restart Service Container** and click Yes.
11. Check the service containers to make sure the Email container starts correctly.

If the connector is running properly, it will be marked with a green circle, as shown in the figure below.

Figure 9-2:
Email connector running properly



Name	Computer	Startup-Type
Collaborative Workspace	trainotcs	Automatic
Data Transformation	trainotcs	Automatic
Email	trainotcs	Automatic
Event Handling	trainotcs	Automatic



If you want to configure automatic polling for a mailbox, you would add XML in the Mailboxes tab. For the purposes of this exercise, you will not configure this. Furthermore, messaging data will be stored in a database and added to the Database Configuration tab. For the purposes of this exercise, the database has already been configured and added.

12. Close the System Resource Manager.

13. Return to the Warehouse organization.

Email building block

The Email building block is a placeholder in the system to indicate that email is available for the designated entity. When the Email building block is added to an entity, it requires an email configuration and it creates a child entity. This child entity contains identity information, an email log, and content.

The email log in the child entity keeps a record of the emails which are sent and received from items of the entity. The content attribute contains the attachments which are received from inbound emails. The content attribute is a Content building block, which is intended to store multiple files.



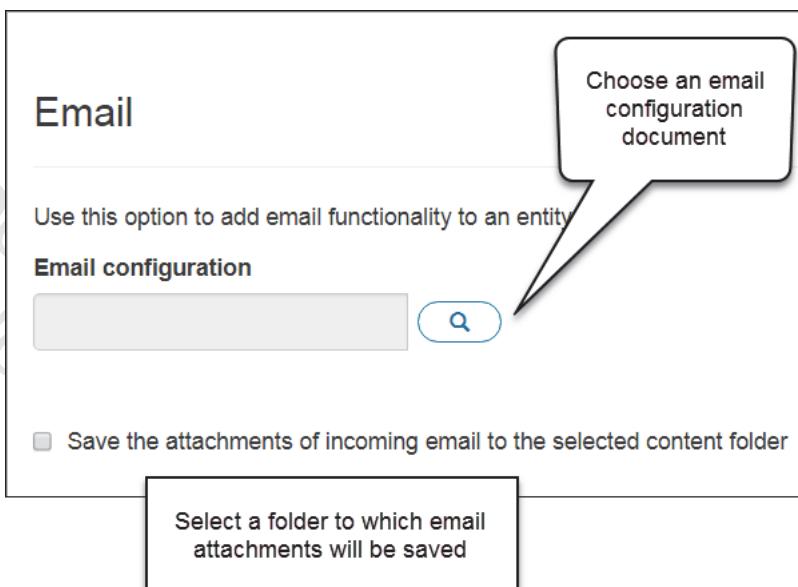
You will learn more about the Content building block in the next chapter.



You should never modify or delete any of the attributes of the email child entity.

Linking the email configuration to the email building block	The email configuration which you established must be connected to the solution, so that entities of the solution can take advantage of the email connection. This is accomplished in an <i>Email configuration document</i> . The configuration document is little more than a placeholder in the project for the email configuration to be linked. This link is established in the Process Experience administration console.
--	---

Figure 9-3:
Email building block details



Add the Email building block and configuration document

1. Switch to the Warehouse organization if you are not in it already.
2. Launch the Collaborative Workspace and open the Northwind Workspace.
3. Open the Product entity from the Warehouse Application.
4. Click **Add > Email**.
5. Click the browse button.

You must choose an email configuration document, but you must create one first.

6. Click **New > Email Configuration**.

You will see a message that you cannot specify the email configuration here. This message is normal: the document is a placeholder for linking the email configuration to the solution through the Product entity. There are no settings.

7. Click **Save**.
8. Set the Name to **MercuryMailConfiguration**.
9. Click the browse button next to Location.

10. Right-click the Warehouse Application and select **New Folder**.
11. Set the name to **Email**.
12. Select the **Email** folder and click **OK**.
13. Click **OK** again on the save dialog.
14. Close the configuration document.

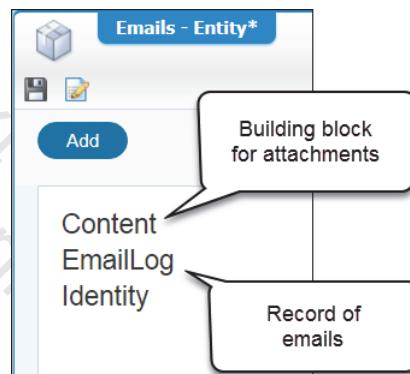
You will set the properties of the configuration document in Process Experience administration console.

15. Select the new **MercuryMailConfiguration** document and click **OK**.

The document is added to the Email building block.

16. Click **Add**.

Figure 9-4:
Contents of the child entity for Email building block



Email template building block

You can choose to pre-write email messages into templates so that your users need only choose the template, perhaps make some minor changes, and send them. Email templates are required when the email being sent is automated (cf, Deadline building block).

You can add as many email templates to an entity as you wish, but you must add the Email building block before you will be allowed to add any email templates.

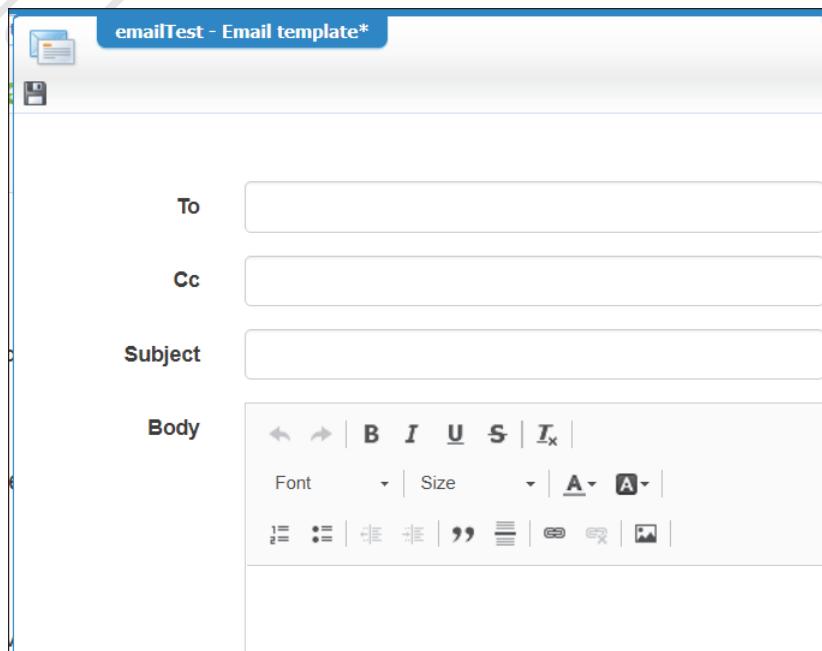
The email template editor is a rich text editor, as you may commonly find with any editor for composing email messages. The email templates, however, may include a mixture of rich text and expression language so that the contents of the messages may contain the relevant details of the associated entity instance. The expression language you add to an email template may even extend to text patterns (e.g., regular expressions). Text patterns may be useful for expressing phone numbers or postal codes when preparing a template from values.



Regular expressions require their own logic and syntax and are a style of programming language. You can learn about regular expressions at:

<http://www.regular-expressions.info/>

Figure 9-5:
Email template editor



Add an email template to the Product entity

1. Open the Product entity from the Warehouse Application.
2. Click **Add > Email template**.
3. Set the display name to **Product Test** and the name to **emailTest**.
4. Click **Add**.
5. Click **Configure**.

6. Set the following values for the email template:

- To: ralba@opentextls.com
- Subject: Product Test
- Body:

This is a test of the Product entity email service.
 Product number: {item.Identity.Id}
 Product name: {item.Properties.ProductName}

7. Save and close the email template.

8. Save the Product entity but do not close it.

Adding the email and email template building blocks is not enough to use emails in Process Experience. You must also support the email service by adding it to other building blocks in Process Platform.



Support the email service

1. Open the Product entity from the Warehouse Application if it is not open already.
2. Select the workWithProduct layout.
3. Click **Configure**.
4. Drag the Emails panel to the right-side of the layout. Set the chrome to full.
 Resize the panels as appropriate.
5. Save and close the layout.
6. Select Security.
7. Click **Configure**.
8. Select the Product Manager role.
9. Scroll to the bottom of the security configuration editor.

Roles		Permissions	
Product Manager		Rules	
		propSetRecieveShipment	
		pbDiscontinue	
		Activity flow	
		Initiate activity flow	
		Lifecycle	
		Add tasks	
		Review (Review)	
		Distribute to Marketing (Distribute)	
		Email	<input type="checkbox"/> Send Email
		Email template	
		emailTest	<input type="checkbox"/> Read

Figure 9-6:

Security privileges for granting access to email and email templates

10. Select **Email > Send Email**.

11. Select **Email template > emailTest > Read.**
12. Save and close the security configuration editor.
13. Save the Product entity.
14. Validate and publish the Warehouse Application.



Configure the email document in the Process Experience administration console

1. Open a new browser window.
2. Use the favorites bar to select the Process Experience administration console for the Warehouse organization.

You should be automatically logged in by virtue of OTDS single sign-on.

3. Select the NorthwindWarehouseApplication from the left.
4. Examine the Configurable Elements from the center panel.

The Email Configuration document you created has added a new configurable element to this list: MercuryMailConfiguration.

5. Select MercuryMailConfiguration.

The configuration properties are presented in the right panel.

Configuration Properties	
From	Enter the sender's email address
Display name	Enter the display name for the sender's email address
<input type="checkbox"/> Use the full name of the logged in user as the display name	
Reply to	Enter the email address to which a recipient sends replies
Email profile user ID	Enter the user ID of the profile for accessing the mail server

Figure 9-7:
Configuration properties of email configuration document in Process Experience administration console

6. Set the following properties for the mail configuration:
 - From: Admin@opentextls.com
 - Display name: check Use the full name of the logged in user.
 - Reply to: cho@opentextls.com
 - Email profile user ID: Admin
7. There is no need to save your changes. They are implemented instantly and automatically.



Test the email service

1. Open a browser window to Process Experience, Warehouse organization, or refresh the window if you have it open already.
2. Click **Home Page > Home Page**.
3. Select **Products > Product Catalog**.
4. Select any of your products and open.

The Emails panel should be present on the right-side of your layout.

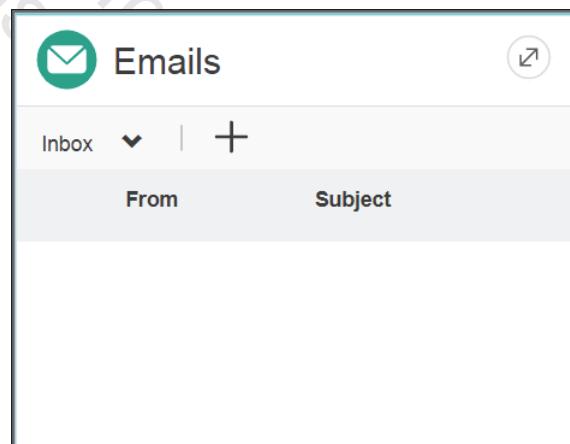


Figure 9-8:

Emails panel

5. Click the compose (+) button.

The email template dialog should appear.

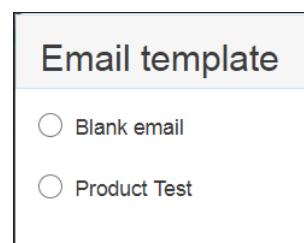


Figure 9-9:

Email template dialog

6. Select *Product Test*.

A draft of the template should be composed on the right.

7. Click *OK*.

8. Confirm that the message is being sent to *ralba@opentextls.com*.

9. Click *Send*.

10. Click *Inbox > Sent* in the *Emails* panel.

11. Confirm that the message you sent appears in this list.



Messages sent and received in the inbox are stored in the EmailLog: a building block of the Email child entity you discovered earlier in this chapter.

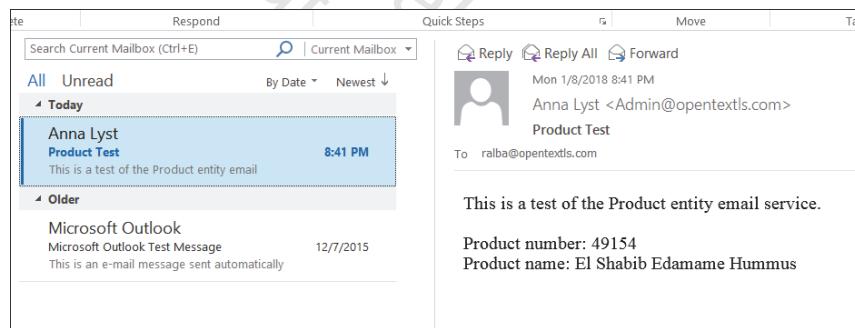
12. Click Microsoft Outlook on the Windows taskbar.

13. Choose *ralba* as the Profile Name and click *OK*.

Use opentext as the password if you are prompted.

14. Confirm that you received the test email, that the email was received from the user logged in to Process Experience (Anna Lyst), and that the contents reflect the values of the item you selected.

Figure 9-10:
Successful email test



Deadline building block

Now that you are able to send email, you can take advantage of building blocks which use the email service. The email template is one such building block. Deadline is another.

The Deadline building block is used to establish a date and/or time in which an entity or task of a lifecycle must be started or completed. A deadline on a start date enforces the date/time in which the task must be claimed and started. A deadline on a completion enforces the date/time in which the task must be finally completed and released.

The consequences of not completing a task within the deadline parameters depend upon the deadline itself. Some deadline violations may escalate the task or send an email to a supervisor. You must have the email service properly configured and added to the entity in order to accommodate this functionality. You must also have an email template composed and ready for the deadline scheduler to send the email.

You may add as many Deadline building blocks to an entity as you wish, in order to capture different deadlines and different consequences. Each deadline has a display name, name, and description. After adding the Deadline building block, you must configure its policies and escalation.



Deadline building blocks measure independently of due dates in activity flows or lifecycles. A deadline could run concurrently with activity flow/lifecycle due dates.

Deadline policies

A deadline is composed of two policies: start and stop. These policies reflect when the deadline timer begins (start policy) and when it ends (stop policy). These times are measured against a deadline time, which begins running when the start policy condition is met. The deadline time can be measured statically or programmatically by reading a property to determine a value.

Figure 9-11:

Deadline policies

Deadline definition		Alerts and escalations	
Start policy*			
when an instance is created			
Stop policy*			
when an instance (lifecycle) is completed			
Default deadline			
<input checked="" type="radio"/> Static		<input type="radio"/> Read from property	
Days	Hours	Minutes	
<input type="text"/>	0	0	<input type="text"/>

Start and stop policies can commence on a number of predefined events:

Start policies There are three start policies a user can select:

- When an instance is created. As soon as the instance is created, the deadline timer starts. This means that if a due date has been set on a lifecycle, that the deadline timer and lifecycle due date timer can run concurrently.
- When an instance enters a certain state. This refers to states in a lifecycle. You must have a lifecycle added to the entity in order to use this policy. The deadline building block assesses the states in an attached lifecycle model and presents the user with a list of states to select. When the instance enters into that state as the lifecycle is being executed in the run-time, the deadline timer commences. This is analogous to setting a due date on a state in a lifecycle. When you select this policy, the stop policy is automatically set to “when an instance exits a selected state”.
- When a certain condition is met. If you choose this option, you must build an expression using the expression editor, in the same way that you built a rule in a previous chapter. The expression can be built in basic or advanced mode. If the condition evaluates to true, then the deadline timer will start. As well as the entity’s properties, you also have access to properties of the lifecycle, activity flow, and deadline building block itself: this includes properties such as status, state IDs, the deadline due date property, and so on.

Stop policies There are a number of stop policies a user can select:

- When an instance (lifecycle) is completed. The timer stops as soon as the lifecycle reaches a final state.
- When an instance exits a selected state. This option is selected automatically when the user selects “when an instance enters a certain state” start policy.
- When a certain condition is met. You must build an expression using the expression editor in basic or advanced mode. If the expression evaluates to true, then the deadline timer stops.

Due dates vs deadline There are some parallels between due dates in activity flows and lifecycles (or lifecycle states) and deadlines. Timers for both can commence when an instance is created or when a state is entered. The difference between them is how alerts and escalations are handled.

With due dates, an event is fired when a due date threshold is passed. By default, that will mean the flow will be flagged in the inbox as overdue. You can also build functionality which will intercept the past due event and build your own escalation pattern.

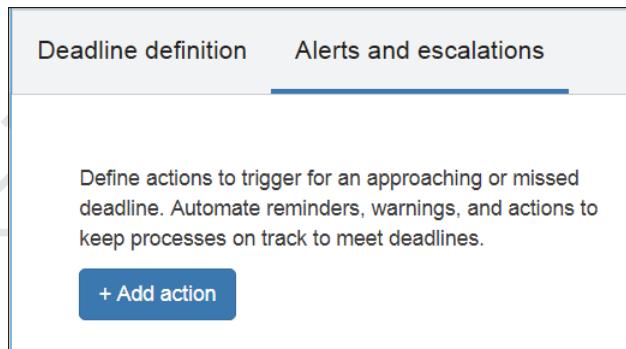
With deadlines, however, the alerts and escalations are provided out of the box as part of the Deadline building block. You will build alerts and escalations next.

Alerts and escalations When a deadline is transgressed, there are certain actions you may want to perform. These actions could be notifying a supervisor, creating a new task, or distributing the work to another team member. In this way, a deadline can be a useful tool to ensure work is completed if, for example, the claimant is on vacation and forgets to release tasks back to the communal inbox.

Actions can also be added before a deadline is met. If a deadline is impending, you can choose to create an action to remind the claimant that the deadline is approaching.

There is no theoretical limit to the number of actions which can be performed for a deadline, but it is a recommended practice to limit the number of actions to between 1 and 3.

Figure 9-12:
**Deadline alerts and
escalations**



Actions are added to the alerts and escalations tab sequentially, in a grid format. This does not necessarily mean that the actions will be fired in the order in which they're added: each action must be assigned a schedule. The scheduler which is associated to the deadline will consider the deadline start and stop policies as well as the alert and escalation actions which are included.

 An alert action cannot fire if the deadline's stop policy has been met.

Add action

Action

Process*
 

Schedule*
 Days

Figure 9-13:
Trigger process action added to alerts and escalations tab

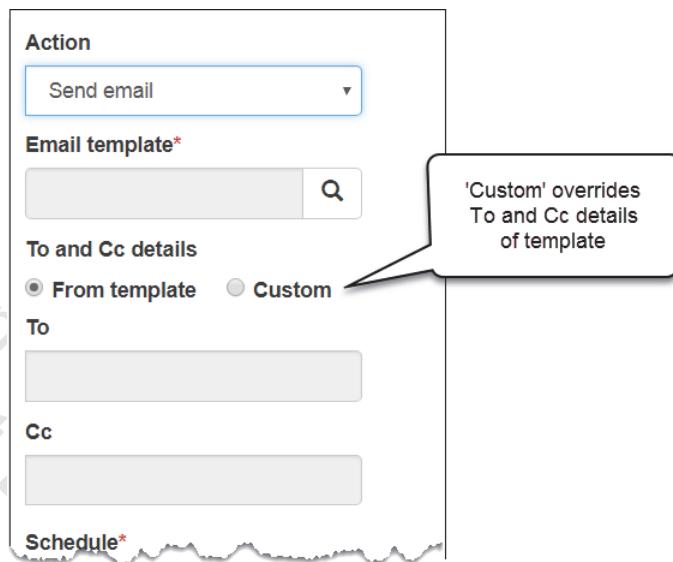
Action There are two actions which can be added to a deadline: a message to trigger a business process (and the associated business process) or send an email (from an associated template). If you trigger a business process, you can build the business process model on the fly and add it to the deadline action. Email templates must be built beforehand (using the Email template building block) and available for selection.



You will build business process models in course 4-4913: Process Modeling for Process Platform.

If you choose to send an email, you must also populate the “To” and “Cc” lines. You can populate these from the selected template or you can override the values using the ‘custom’ option, but you cannot edit the content of the template.

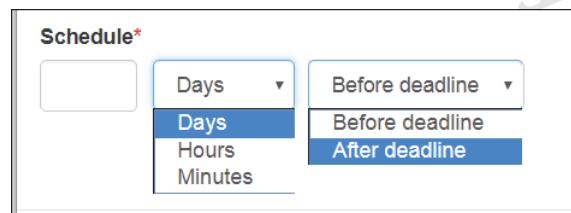
Figure 9-14:
Send email action with
associated attributes



Schedule Actions can occur before or after a deadline expires. Scheduling an action before the deadline expires is useful for issuing a reminder. The scheduling tool for an action is three simple, literal, static expressions:

- An integer to represent the time period of the schedule.
- A drop-down to select the time measure: days, hours, or minutes. You cannot combine these measures (e.g., 2 days and 4 hours), so you must choose the time measure which is the lowest common measure.
- Whether the action is triggered before or after the deadline.

Figure 9-15:
Action schedule





Actions are only fired once per schedule, and actions cannot be scheduled to repeat. If you want to repeat an action, consider triggering a business process model with a loop as the action. The process model can start according to the schedule and repeat based upon the loop details.

Business process models can also be useful for starting consequential escalations. For example, if a task is escalated to the manager after one week, and must be escalated to the director if another week passes, build a business process which will handle this type of escalation and trigger it from the deadline.



Impose a deadline on an entity

1. Open the Product entity from the Warehouse Application.
2. Click **Add > Email template**.
3. Set the display name to **Overdue email message** and the name to **emailOverdue**.
4. Click **Add**.
5. Click **Configure**.
6. Add the following details to the message:
 - To: `ralba@opentextls.com`
 - Subject: `Product flow overdue`
 - Body:
 The following product is now overdue.
 Product number: `{item.Identity.Id}`
 Product name: `{item.Properties.ProductName}`
 Current state: `{item.Lifecycle.CurrentState}`
7. Save and close the email template.
8. Click **Add > Deadline**.
9. Set the display name to **Product Timer** and name to **dlineProduct**.
10. Click **Add**.
11. Click **Configure**.
12. Set the Start policy to **when an instance is created** and the Stop policy to **when an instance (lifecycle) is completed**.
13. Set the default deadline to **3 minutes** (no days, 0 hours).
14. Click **Alerts and escalations**.
15. Click **Add action**.
16. Set the action to **Send email**.
17. Use the browse button to select the **emailOverdue** template.
18. Set the schedule to **1 Minutes After deadline**.
19. Click **Save**.
20. Save and close the deadline.

21. Select **Security**.
22. Click **Configure**.
23. In the security configuration editor, select the **Product Manager** role.
24. In the security configuration grid, enable the **Read** option on `emailOverdue` and enable the **Update** option on the deadline (`dlineProduct_DueDate`).
25. Save and close the security configuration.
26. Save and close the **Product** entity.
27. Validate and publish the **Warehouse Application**.
28. After publishing is complete, switch to the **Process Experience** browser window and refresh it.
29. Create a new product in **Process Experience**. Provide some default values and click **Create**.
30. From the home page, click **My Inbox > All tasks**.



Even though a deadline has been added to the entity, it will not display in the Due Date column of the All Tasks list: this value is restricted to due dates added to the lifecycle states/action flow activities.

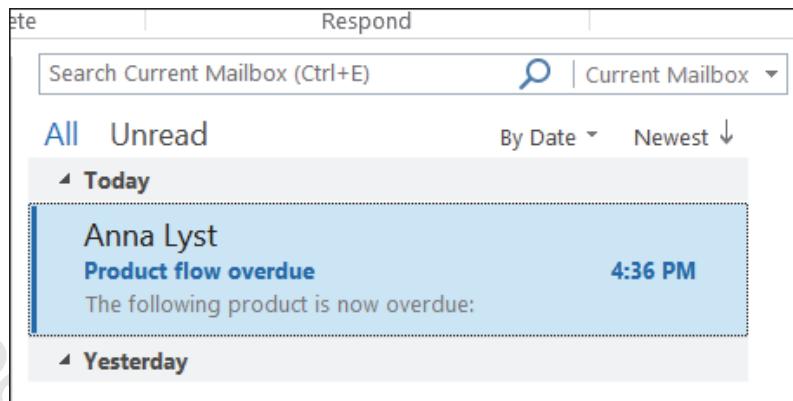
31. Wait for three minutes so that the deadline will expire. Refresh the inbox periodically.
32. After three minutes, the deadline will expire and trigger the actions. Wait one additional minute.
33. Click **Products > Product Catalog**.
34. Locate the new product and open it to the product layout.

You should have a lifecycle progress panel in the layout to show you where the item exists in the flow.

35. Open Microsoft Outlook. If you need to log in, use user `ralba` and password `opentext`.

Once the deadline expires, you should receive an email one minute later.

Figure 9-16:
Overdue email in inbox



If you have the history block added to the entity (and the history button is available in the action bar), you can examine the history to see the application of a deadline.

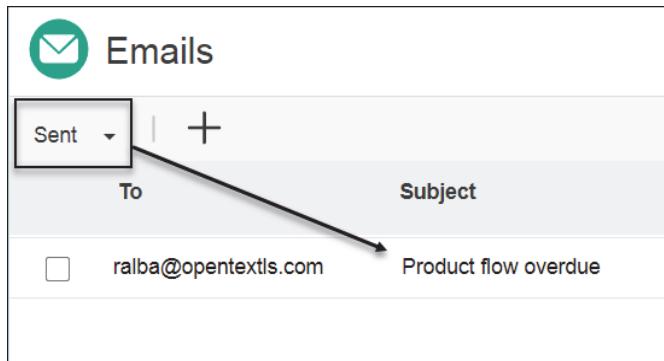
Modified On ▾	Action	Modified By
2018 4:35 PM	Property 'EscalationStatus' was changed from 'Pending' to 'Processed'.	analyst@tr
2018 4:35 PM	An escalation(1 minute after deadline) email notification has been triggered for the deadline 'dlin...	analyst@tr
2018 4:35 PM	Item '000C2915C91BA1E7BD2AA8433988C20F.65539.16385' was created under Email.	analyst@tr
2018 4:35 PM	Draft Item '000C2915C91BA1E7BD2AA8433988C20F.65539.16385' was created under Email.	analyst@tr
2018 4:33 PM	History for Item 'Product 65539: Holy Guacamole Spicy' was viewed.	analyst@tr
2018 4:33 PM	Item 'Product 65539: Holy Guacamole Spicy' was accessed.	analyst@tr

Figure 9-17: Examining escalation in item history



Emails sent from a deadline escalation action are recorded in the email building block's email log.

Figure 9-18:
Escalation email maintained in email log, displayed in Emails layout panel



Assignee building block

The Assignee building block enables you to specify who is responsible for an item. An item can be assigned to a workgroup, such as a role and organization unit from the identity package, or to a specific named user. This is a different concept from work assignment, as you have seen with activity flows and lifecycles: an assignee is responsible for *all instances* of a particular entity.

The Assignee building block specifies the default assignee, whether e-mail notifications are sent to assignees, and the e-mail template to use for notifications. Each entity can contain only a single Assignee building block. This is because, as mentioned above, an assignee is responsible for all instances.

Assignees may be assigned statically or programmatically by reading values from a property. If the assignee is set statically, you may choose only a single, specifically named user, a single role, or an organizational unit.

Assignee actions on action bars Including the Assignee building block makes the following actions available for an action bar. You can also configure permissions on these actions in the security configuration editor.

- **Assign** - Assigns responsibility for an item that is currently unassigned. If assignee is set to a role or organization unit, this action enables a user (with appropriate permission) to assign an item to a user from that workgroup.
- **Claim/Revoke** - Accepts responsibility for an item that is currently unassigned or assigned to one of the participant's workgroups, setting the assignee to the participant specifically. This action requires claim permission to be granted in the security building block. Moving an item back to the workgroup requires the Revoke action.



If the entity also contains a Discussion building block, the person who revokes an item is prompted for comments to explain the revocation.

- Forward - This action assigns responsibility for an item that is currently assigned to either the participant individually or to one of the participant's workgroups to another specific participant or workgroup. This action displays a form prompting for the workgroup or individual to be set as the assignee.

These actions, as you can determine, are similar to those that one might find with a lifecycle. However, when the Assignee building block is attached to an entity, these actions become available on all instances of the entity, even if that entity does not have a lifecycle, and on all phases of the entity instance, even if the lifecycle is complete.



An entity with no clearly defined process to build into a lifecycle can benefit from the Assignee building block accompanied with another block, such as Deadline or Tracking.

Assignee configuration

The Assignee building block is composed in two sections: the default assignment and notifications. As previously mentioned, the default assignment can be to a person (user), role, or organizational unit. Notifications can be sent when a new assignee is assigned or when the assignee changes. Notifications are sent in the form of an email template.

Figure 9-19:
Assignee configuration

The screenshot shows the configuration interface for the Assignee building block. At the top, there is a header bar with a user icon and a save button. Below the header, the title "Assignee - Assignee*" is displayed. The interface is divided into two main sections: "Set default assignment" and "Notify assignee through email".

Set default assignment: This section contains a dropdown menu labeled "Assignment type" with the option "User" selected. There is also a small empty text input field.

Notify assignee through email: This section has two subsections. The first subsection, "When the entity is assigned or delegated, send an email to:", contains two checkboxes: "New assignee" and "Former assignee". Each checkbox has an associated "Email template" input field with a search icon. The "New assignee" input field is empty, while the "Former assignee" input field contains the placeholder "Email template".

Assignment type When you choose a role as the assignment type, you will be presented with a lookup tool and field which will allow you to lookup a role from the Process Platform system. A role, you may recall, is stored as a document (a role document) in Process Platform. Instead of a static role, you may choose a dynamic role, which is assigned programmatically using expression language.

When you choose a user or an organizational unit as the assignment type, however, these are not presented as Process Platform documents. Instead, you are presented with a multi-line text field, and you must use expression language to read a property to determine the user or organizational unit.



Assignments can be changed in Process Experience. A Process Experience user does not need to use expression language to select a new user, role, or organizational unit for changing an assignment.

Notifications When an entity is assigned to a new assignee, or when the assignee changes, a notification email can be sent to the new assignee and/or the old assignee. You must select the recipient's of the notification, and the email template for each notification. The templates are instances of the Email template building block.



Include an assignee for an entity

1. Open the Product entity in the Warehouse Application.
2. Click **Add > Assignee**.
3. The building block takes no settings. Click **Add**.
4. Click **Add > History**.
5. Use the browse button to browse for a history log.
6. Click **New > History Log**.
7. Set the display name to **Product History Log** and name to **logProducts**.
8. Set the compression type to **GZip**.
9. Click **Save**. Use the location browser to save the log to Warehouse Application/Security. Click **OK** to save the log, then close it.
10. Select **logProducts** from the list and click **OK**.
11. Check **Record access events** and click **Add**.
12. Click **Add > Email template**.
13. Set the display name to **Product Return assignment** and name to **emailReturnProduct**.
14. Click **Add and continue**.
15. Set the display name of the second email template to **Change Return notification** and name to **emailChangeReturn**.
16. Click **Add**.

17. Select the Product Return assignment email template and click **Configure**.
18. Set the following properties for the email template:
 - To: ralba@opentextls.com
 - Subject: Product Return Assignment
 - Body:

The following product has been designated for return:
{item.Properties.ProductName}
19. Save and close the email template.
20. Select the change return email template and click **Configure**.
21. Set the following properties for the email template:
 - To: ralba@opentextls.com
 - Subject: Change Return Notification
 - Body:

{item.Properties.ProductName} return designation has changed.
22. Save and close the email template.
23. Select the Security building block.
24. Click **Configure**.
25. Select the Product Manager role.
26. Scroll down to the Assignee section.
27. Grant Read, Update, and Execute access to all aspects of Assignee security:
 - Read: AssigneeIdentity, GroupIdentity
 - Update: AssigneeIdentity, GroupIdentity
 - Execute: Assign, Claim, Revoke
28. Grant Read access to the return and change product email templates.
29. Save and close the security configuration editor.
30. Select the Assignee building block.
31. Click **Configure**.
32. For Assignment type, select **User**.

User assignment is always dynamic and requires expression language.
33. Use expression language to select the current user:

```
user.Properties.Name
```

Do not use curled brackets: it is assumed that the contents of this field are written only in expression language.
34. In the notifications section, select **New assignee**.
35. Use the lookup tool to select the return product email template.
36. Enable **Former assignee**.

37. Use the lookup tool to select the change return email template.
38. Save and close the Assignee editor.
39. Select the default action bar and click **Configure**.
40. Under the *Email* heading, select *Send Email*.
41. Under the *Assignee* heading, select *Assign*, *Claim*, and *Revoke*.
42. Under the *History* heading, select *History*.
43. Save and close the action bar.
44. Save and close the entity.
45. Validate and publish the Warehouse Application.



Test assignment in Process Experience

1. Refresh the Process Experience home page.
2. Add a new product. Provide some default values and click **Create**.
3. Select the **Products > All Products** list.

By default, the new product was assigned to the current user (User = {user.Properties.Name}). This means that the item has already been claimed by the active user: the Revoke button should be present in the action bar, but not Claim.

The first test will demonstrate that the item has automatically been assigned.

4. Click **History** in the action bar.
5. Examine the history contents. You should discover that the item was automatically assigned to the active user upon creation.

2018 11:17 AM	Item 'Product 81926: Restocking' (000C2915C91BA1E7B97AD5C0662C0205.81926) was crea...	analyst@training.local
2018 11:17 AM	Item '000C2915C91BA1E7BD5DF9DBEF29820F.81926.16386.16386' was created under Dea...	analyst@training.local
2018 11:17 AM	Item '000C2915C91BA1E7BD56CACCB06B420F.81926.16386' was created under DeadlineIns...	analyst@training.local
2018 11:17 AM	Item '000C2915C91BA1E7BD016E187FEAC20D.81926.49158' was created under LifecycleTask.	analyst@training.local
2018 11:17 AM	Item 000C2915C91BA1E7B97AD5C0662C0205.81926 is assigned to user analyst@trainin...	analyst@training.local
2018 11:17 AM	Item '000C2915C91BA1E7BD016E187FEAC20D.81926.49157' was created under LifecycleTask.	analyst@training.local
2018 11:17 AM	Item '000C2915C91BA1E7BD0162CA000FC20D.81926.49156' was created under Activityflow.	analyst@training.local
2018 11:17 AM	Draft Item '000C2915C91BA1E7BD016E187FEAC20D.81926.49157' was created under Lifec...	analyst@training.local

Figure 9-20: Item assigned upon creation, history

6. Close the history dialog.
7. Click **Home Page > Home Page**.
8. Select **My Inbox > Personal Tasks**. These are just the tasks which have been claimed or assigned.

The new item should be labeled as Assigned.



The source may be listed as “Product”, and may not include the product name. Use the Start Date column as an indicator of the product.

The next test is to re-assign the task to another workgroup or user.

9. Select **Products > All Products**.
10. Open the product you created.
11. Click **Revoke**.
12. Click **Assign**.

The assignment dialog should appear.

You may assign the item to a workgroup or user. Workgroups include roles or organizational units. You created an organizational unit previously with the identity package.

13. Click **Specific user**.

A list of the system users is displayed.

14. Click **Work group**.
15. Select the **Organizational units** tab.

A list of the organizational units you added to the identity package is displayed.



The Roles tab allows you to assign the work to a Process Experience role.

16. Select **Northwind Home Office**.
17. Click **Select user**.
18. Click **Select**.
19. Select the product from the list and click **Send Email**.
20. Select the Product Return email template and click OK.
21. In the New email dialog, accept the defaults and click Send.
22. Verify that the email was sent and that user ralba received it in Microsoft Outlook.

Summary

Having completed this chapter you should be able to:

- Configure Process Platform to connect to an email server
- Add the Email building block to an entity to enable it for the email service
- Write message templates with the Email template building block
- Configure email in the Process Experience administration console
- Send email from an entity instance in Process Experience
- Impose a deadline on an entity using Deadline building block
- Distribute entity instances using the Assignee building block

Exercises

1. Configure the Business entity for email.

You can reuse the same email configuration document that was built during class.

Hint: build an email template to test email functionality.

Hint: remember to add the Emails panel to the layout.

Hint: remember to replace security and configure it for use with your entity.

2. Add a deadline on the Business entity to impose a 3-minute time limit on establishing customer credit.

The deadline will commence once the Customer lifecycle enters the Establish Credit state.

Send a reminder of the task (1 minute before deadline) and an overdue email if the state is not completed within the time allotted (2 minutes).

Hint: build an email template for each email message.

3. Commit your work to the repository with the description: “Chapter 9: Flow Support”.

Open Text Internal Use Only
Do Not Distribute

10. Integration and advanced features

Objectives

On completion of this chapter, participants should be able to:

- Add files to entity instances in Process Experience
- Describe web services and their role with entities
- Explain web service operations exposed on entities and relationships
- Expose an entity as a web service
- Explain external entities
- Import entities from database tables

Overview

In this course, you have built entities, associated them together with relationships, and added building blocks to them. You have added lists to visualize instances of entities, forms to create new items, and layouts to organize your data. You have also added rules to insert business logic into your solution and security to control who has access to entities, and what type of access is included. You have developed flow constructs such as lifecycle and activity flow for items across their evolution, and connected them with an email server for sending preconfigured message templates. You are able to publish these entities and their building blocks to Process Experience and test your solution at a rudimentary level.

In this chapter, you will examine other aspects of data-driven design, and the transition from entity modeling to larger development efforts in Process Platform. You will examine the interaction of entities with files and web services, and how to integrate your entity solution with business process management.

Finally, you will investigate how to take existing data and dynamically create entities in Process Platform using database tables.

What you will build in this chapter

- File building block: use this building block to attach a single file to an entity instance.
- Content building block: use this building block to attach several files in a defined folder structure to each entity instance.
- Web service building block: convert an entity into a web service which includes operations such as create, read, update, and delete, as well as configurable “find” operations.

Timing

Lecture: 75-90 minutes

Exercises: 75-90 minutes

References

More information about this topic is available:

- Process Suite Community (<http://www.opentext.com> > Community > OpenText Process Suite Community)
- Process Platform 16.3 Entity Modeling Guide

Attach files and content to items

Some instances of entities may require files to be attached to them, such as reports, images, or marketing files. These files may be in any type of format. This can be useful when saving attachments from inbound email messages.

File building block

The File building block adds an indicator to the entity that some type of file may be attached to the entity. By default, the file will be stored in the Process Suite document store. The document store can be a different database or OpenText Content Server. The document store depends upon the configuration which you're using in your Process Suite solution. The document store is configured in Process Platform.

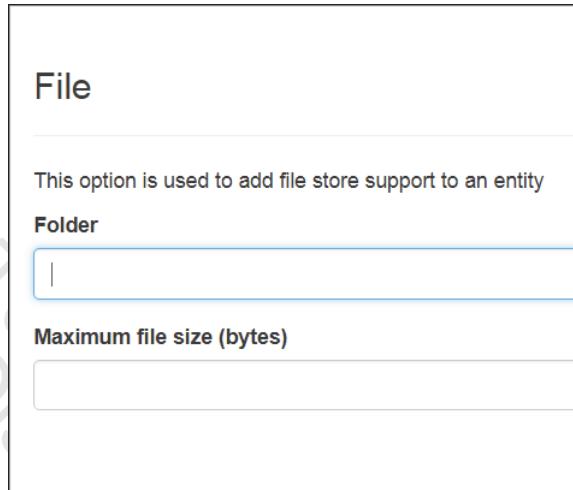
By default, the document store is a database which has been allocated space on your database server. However, you can install and configure an instance of OpenText Content Server ("Content Server") and use Content Server as the document store. When you are using Content Server as your document store, and it is properly configured with OTDS, users can seamlessly move between Process Platform, Process Experience, and Content Server using single sign-on functionality.



Your system administrator will configure the Process Suite document store in Process Platform.

Only one File building block may be added to an entity. Including the File building block simply allocates space on the database for a file to be attached to instances of that entity. The File building block only allows one file to be attached to an item. You may name a folder, so that files will be organized in a particular folder in the document store when they are saved, and you can allocate the maximum file size (in bytes) to be saved, but you cannot assign a name or display name.

Figure 10-1:
Properties for File
building block



Add a File building block to an entity

1. Open the Warehouse Application in the Northwind Workspace, if it is not already open.
2. Double-click the Customer entity to open it in the editor.
3. Click **Add > File**.
4. Set the Folder to **Profiles**.
5. Click **Add**.
6. Select Security.
7. Click **Configure**.
8. Select the Account Manager role.
9. Scroll to the bottom of the security configuration editor.

There are several configurable options for file security. These options are directly related to functions available in Content Server, even if you are not using Content Server.

The screenshot shows a table-based security configuration interface. The columns represent different entity types or categories. The rows represent specific permissions. The 'File' row contains a list of actions: Download, View versions, View properties, Search, Audit, Upload, Check out, Check in, Copy attachment, Link attachment, Update properties, Share in repository, and Public URL. The rightmost column contains two checked checkboxes: 'Execute' and 'Execute'. The top row has two empty checkboxes.

				<input type="checkbox"/> Execute
				<input checked="" type="checkbox"/> Execute
File	<input type="checkbox"/> Download <input type="checkbox"/> View versions <input type="checkbox"/> View properties <input type="checkbox"/> Search <input type="checkbox"/> Audit <input type="checkbox"/> Upload <input type="checkbox"/> Check out <input type="checkbox"/> Check in <input type="checkbox"/> Copy attachment <input type="checkbox"/> Link attachment <input type="checkbox"/> Update properties <input type="checkbox"/> Share in repository <input type="checkbox"/> Public URL	<input type="checkbox"/> Delete versions <input type="checkbox"/> Open in Brava viewer		

Figure 10-2: Security options for File building block

10. Enable the following permissions:

- Download
- View versions
- View properties
- Search
- Upload
- Check out/Check in

11. Save and close the security configuration editor.

12. Save and close the entity.

13. Validate and publish the Warehouse Application to the organization.

File building block at run-time

When an entity is published to the run-time for Process Experience, there is no need to add anything to any of the existing forms or layouts. By virtue of the File building block having been added, the Process Experience engine will automatically include a button so that a user can browse and attach a file to an entity instance when creating an item.



Figure 10-3:

Upload function and browse button automatically added

When updating an item which has the File building block attached, the Process Experience engine will include upload and download buttons so the user can add more or download existing files. If there are no files attached to the item, the download button is disabled.

If the File building block is added to an entity, the action bar will include buttons associated with file handling.

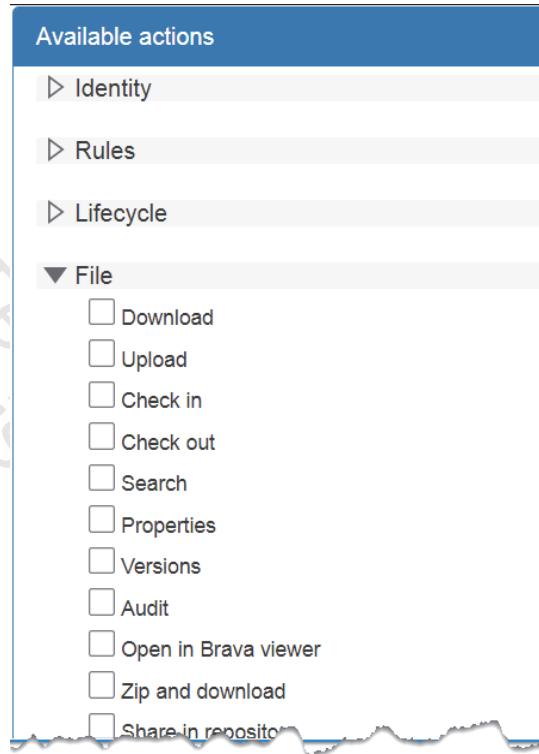


Figure 10-4:

File maintenance buttons available in action bar configuration



When designing your entities, be aware of the purpose a file serves in your solution. You can use the File building block to attach one (and only one) file. You can also use a property with a multi-line text or image data type if that is more relevant for your solution.

Content building block The Content building block is similar to the File building block, in that it allows Process Experience users to attach files to an entity. However, whereas the File building block only allows one file to be attached, Content allows several files.

When the Content building block is added to an entity, it creates a child entity - called Contents - which includes properties such as identity, title, display organization (to organize your attached files), and layout (to display the attached files). The child entity also include the File building block. If you recall, a parent entity may have several child entities. Therefore, by virtue of the File building block being attached to the surrogate child entity, content permits many files to be added to an item with explicit organization options.



To display the list of files which are available with the content, create a layout in the parent entity which contains the content and/or a preview panel.



Do not change or delete any of the building blocks which are added automatically to the Contents child entity.

Only one Content building block may be added to an entity. By default, files will be stored in a root path, which is determined in expression language. You can also set a limit on the number of files that can be uploaded, but not on the size of each of file (as the case with the File building block).

Figure 10-5:
Content building block details

Content

Use this option to create a child entity with a list of files. The child entity includes Layout building blocks. To display the list of files, in the parent entity, create a Preview panel.

Root path
\${system.organization}/\${solution.name}/\${contentbuildingblock.path}

Maximum number of files that can be uploaded
No limit

The root path, by default, is expressed as:

```
 ${system.organization}/${solution.name} /  
 ${contentbuildingblock.path}
```

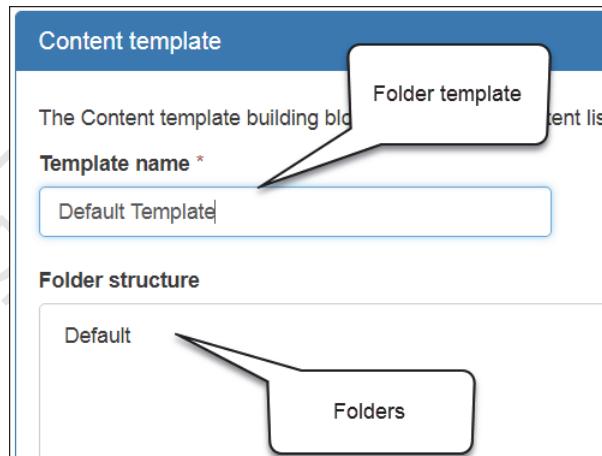
In this course, for example, that path would realize as:

```
 Warehouse/NorthwindWarehouseApplication/Content
```

Folder templates Files added to the Content building block can be arranged and organized in a custom folder structure. The default structure is referred to as the *content template*. The template is established in the Content child entity, in a special building block called the content template.

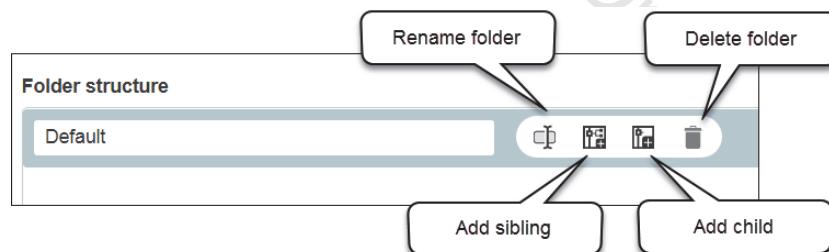
Figure 10-6:

Content template



The content template and corresponding folder structure can consist of any number of folders and sub-folders. Tools provided in the content template allow you to add new folders to the structure.

Figure 10-7:
Folder structure tools



Add the Content building block to the Product entity

1. Open the Product entity from the Warehouse Application.
2. Click **Add > Content**.

3. Accept the default value for Root path and do not express a file maximum.

When you do not express a maximum on the number of files, the system automatically sets the maximum at “no limit”.

4. Click **Add**.
5. Click **Configure child entity**.
6. Select Content Template.

The content template helps to organize the content by folders.

7. Click the **Default Template**.
8. Change the template name to **Product Specifications**.
9. Select Default in the Folder structure list and select the Rename tool.
10. Name the folder **ProductSpecs**.
11. Select the ProductSpecs folder and click **Add Child**.
12. Set the sub-folder name to **Nutrition**.
13. Select the ProductSpecs folder and click **Add Sibling**.
14. Set the folder name to **Images**.
15. Save and close the content template.
16. Save and close the Contents child entity.
17. Select the workWithProduct layout.
18. Click **Configure**.
19. Drag a Contents panel and layer it on top of the Emails panel.
20. Save and close the layout.



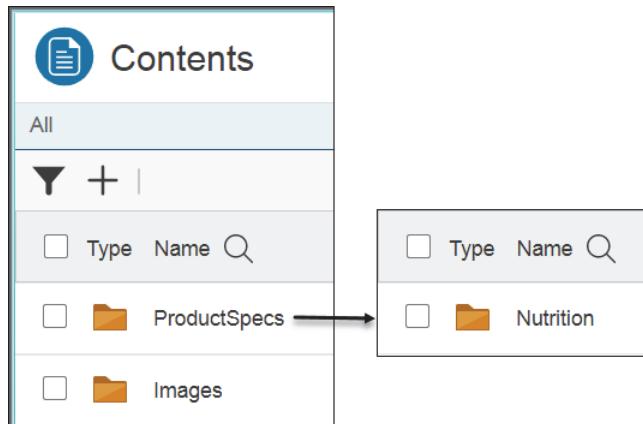
You must set security for the File building block, but not for Content. If you want to set Security for the Content block, add a Security building block to the child entity of Content.

21. Save and close the Product entity.
22. Validate and publish the Warehouse Application to the organization.

In Process Experience, the Contents Panel should be presented as displayed in the screen capture in the following figure.

Figure 10-8:

Contents panel for Product entity



Entities and web services

Process Platform contains its own Enterprise Service Bus, a communication hub used between mutually interacting software applications in a service-oriented architecture (SOA). When solutions are built using this Service Oriented Architecture, functionality is exposed by means of web services. When people and systems perform actions such as retrieve data, write data, and calculate, they use web services, by means of user interfaces (forms, layouts, etc.), to perform these actions. To facilitate this, connectivity needs to be established so that a web service operation request is sent to the correct systems, applications, users, etc. The web service can be an internal service built for your application as well as an external service that you use in your application. Most of the interactions between services in Process Platform are facilitated through web services.

Web services in Process Platform are based on Simple Object Access Protocol (SOAP) messages. Web service messages are written as SOAP requests and responses. The connectivity that is used to facilitate the routing of web service requests can be compared to sending a written letter to a company:

- The requested information is written. In terms of a web service, this is the input, written in the body of a SOAP request.
- The letter goes into an envelope with the destination address. In web services, this is the SOAP request itself.
- The envelope is put in a mailbox. In this context, the SOAP request is delivered onto the SOA grid.
- The envelope is routed to the final destination on the address. This refers to the XML namespace of the destination.
- The letter is read and a response is sent back. The output, a SOAP response, is prepared and returned.

Web services use input and output messages to communicate. Service consumers and providers use a messaging system in order execute services.

A consumer puts his service request in an envelope, puts the name and address on the envelope according to a certain protocol and puts it into a message delivery system, which ensures that the message is delivered to the right spot. Analogously, the body of a SOAP request contains the message, the SOAP request follows a specific format, the message is delivered to its destination through the SOA grid built into Process Platform, and a formatted SOAP response is returned.

When examining a SOAP request you can distinguish two important parts:

- The header (SOAP:Header): Contains the sender and receiver details.
- The body (SOAP:Body): Contains the web service operation and XML namespace that will be used. The operation contains the input message for the service.

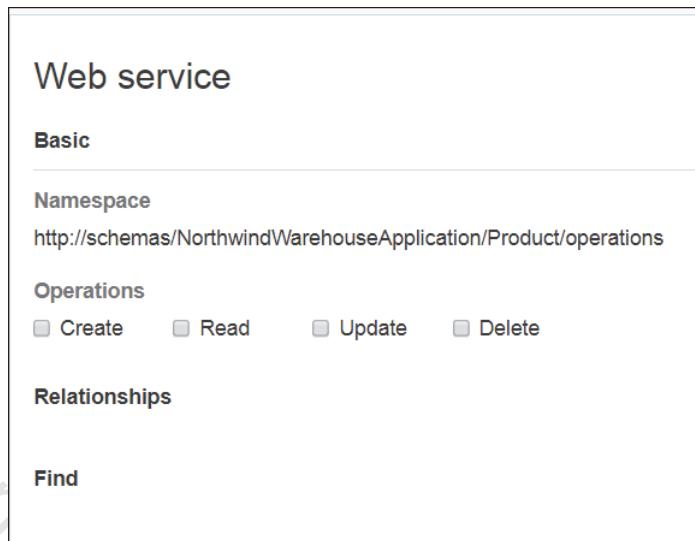
Web services provide a standard means for software applications running on a variety of platforms and frameworks to work together. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions, thanks to the use of standard language like XML. Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticated added-value services.

Web service building block

To integrate your application with other systems or parts of Process Platform you can use web services. With regards to entities, you can add the Web service building block as a means of wrapping your entity and exposing it to other applications in Process Platform as a web service. You can use the Web service building block to attach web service operations to the entity, so that users can execute certain functions on your entity. The following operations can be exposed as a web service on an entity:

- Read: Reads a single entity
- Create: Creates a new entity
- Update: Updates an existing entity
- Delete: Deletes an existing entity

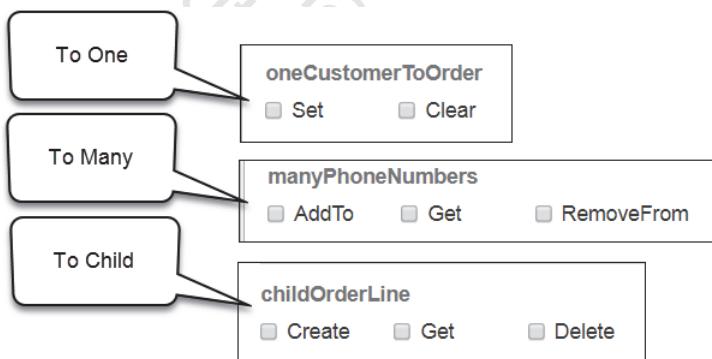
Figure 10-9:
Web service building block on Product entity



Web service operations on entities with relationships The following operations can be exposed as a web service on a relationship:

- To One relationships: Set, Clear
- To Many relationships: AddTo, Get, and RemoveFrom
- To Child relationships: Create, Get, and Delete

Figure 10-10:
Web service operations available for entities with relationships



Expose an entity as a web service

1. Open the Product entity from the Warehouse Application, if it is not open already.
2. Click **Add > Web service**.



The Product entity does not have any relationships. The only web service operations that are available are Create, Read, Update, and Delete.

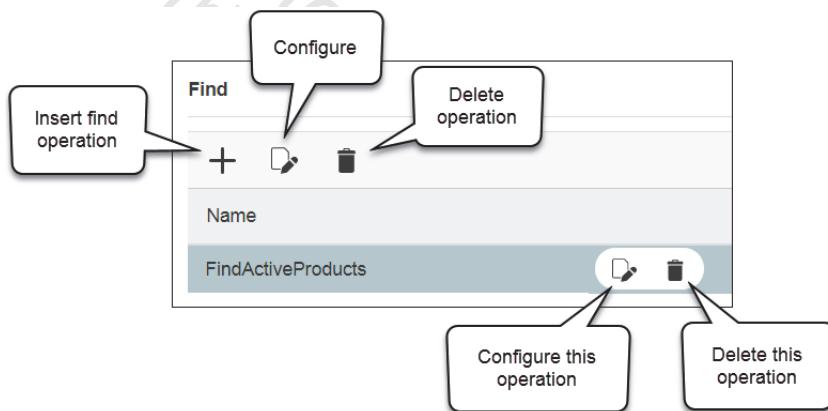
3. Select **Create, Read, and Update**.
4. Click **Add**.
5. Save and close the entity.
6. Validate and publish the Warehouse Application.



You cannot test the generated web service in Process Experience. You can, however, test the web service using the Web Service Interface Explorer in Process Platform. This is an advanced tool for testing published web services which you will examine in course 4-4913. If you choose, there is an advanced exercise at the end of this chapter which will demonstrate how to test a web service.

Find service operation Another operation on an entity web service is Find. The Find operation, which is available with all entity web services, is intended to get a list of all entity instances that fulfill specific criteria. There can be many different types of Find operations on a single web service, but each set of criteria must be built separately in a table.

Figure 10-11:
Find operation table on
web service building
block

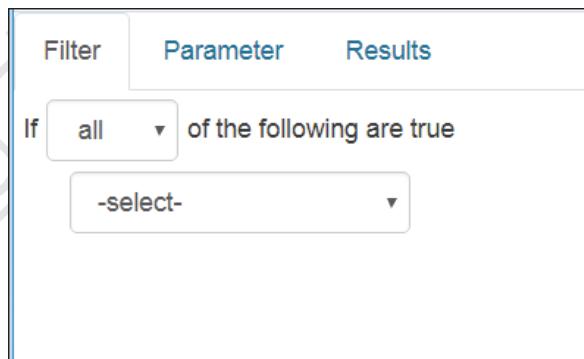


The name of the operation can be any name. If no parameters are used in the configuration of the web service operation, the request takes no arguments. The criteria to find the instances are captured in the expression defined with the expression editor.

The expression editor, however, is a little different from the one which is used for creating and editing rules:

- There is no Basic tab. Conditions are built in basic mode on a Filter tab, instead.
- There is an Advanced tab, but it remains hidden.
- On the Filter tab, you can specify for most operations whether a static value or a parameter (defined in the Parameters tab) must be used.
- A Parameter and a Results tab were added next to the Filter tab to enable the definition of parameters and sorting for the web service operation.
- The browse list for properties on the Filter tab is extended with a browse option to navigate to properties of related entities.

Figure 10-12:
Expression editor for a
Find operation



If an invalid expression is created (for example, when the data types of a property and a parameter do not match), the Advanced tab is displayed and the expression is shown so that it can be corrected there. If it is valid, switching to the Parameter tab and back to the Filter tab shows the corrected expression.



All parameters in the Parameter tab must be used in the Filter tab or an error is generated.



Add Find operations to the Product web service

1. Open the Product entity from the Warehouse Application if it is not already open.
2. Select Web service.
3. Expand the **Find** operation table.
4. Click add (+) to insert a Find operation into the table.
5. Set the name to *FindActiveProducts*.

6. Click **Configure**.
7. On the *Filter* tab, set the inclusivity marker to *all*.
8. Add a condition: **Discontinued equal to Static value No.**
9. Select the *Results* tab.
10. Set the **Sort results by** value to **ProductName ascending**.
11. Save and close the expression editor.
12. Click add (+) to insert another *Find* operation into the table.
13. Set the name to *FindProductsByStockCeiling*.
14. Click **Configure**.
15. Select the *Parameter* tab.
16. Click add (+) to insert a new parameter.
17. Set the parameter name to **OnHand**.
18. Set the Type to **Integer**.
19. Select the *Filter* tab.
20. Set the inclusivity marker to **one**.
21. Add a condition: **In Stock empty**.
22. Add another condition: **In Stock less than/equal to Parameter OnHand**.
23. Save and close the expression editor.
24. Select Security.
25. Click **Configure**.
26. Select the *Product Manager* role.
27. Locate the *Web service* section in the security configuration editor and enable **Use** permission for both find operations.
28. Save and close the security configuration editor.
29. Save and close the *Product* entity.
30. Validate and publish the application.



There is an advanced exercise at the end of this chapter for testing a web service in Process Platform.

External entities

The process for creating an entity, its properties, relationships, and other building blocks is relatively simple and straightforward. However, in most enterprise systems, these business objects already exist in some form. They may be business objects in another system, such as Case360 or MBPM, or they may be tables in a database. It is quite reasonable to assume that most enterprises will have business objects that exist in these Enterprise Information Systems (EIS) which you can leverage instead of re-creating.

In most Process Suite solutions, it would be the preferred process to assess what business object entities already exist in an EIS system, and import those *external entities* into your Process Suite solution as entity objects. In order to do this, you obviously would need to establish a connection from Process Suite to the existing EIS. Then, when you have an entity connection, you could import the external entities into Process Suite. External entities are representations of the underlying structure in the EIS and as such, you are limited to the types of building blocks you can use.

Building block limitations	The underlying structure of an external entity is determined by the EIS from which the entity was imported. This means that there are certain limitations to the building blocks which you can use. Identity and property building blocks cannot be removed or modified, for example. However, you can add building blocks to the external entities, such as rules and layouts.
-----------------------------------	---



You cannot add a form building block to any external entity imported from Case360, MBPM, or Inbox.

You cannot add a lifecycle or activity flow to an imported entity.

Some of the entity building blocks are added automatically, once the external entity is imported from the EIS. Some of these blocks, however, do not provide options which builders can configure. These depend upon the source EIS and the building block that you're trying to import.

Establishing a connection to an external entity

The first stage in importing external entities is to establish a connection to the source EIS. These sources include, but are not limited to:

- Process Platform Inbox
- Process Component Library for Process Suite
- MBPM
- Case360
- Database table



Process Component Library (PCL) for Process Suite has been deprecated, but constructs can be imported as entities.



A custom EIS connector has been provided in Process Suite 16.3 in order to model objects in disparate EIS systems.

The Process Platform Inbox and Process Component Library for Process Suite are considered systems which are internal to Process Suite. Process Suite therefore offers an easy-to-configure option for importing entities from either of these two sources. If you are trying to import entities from another source, however, you must create the connection and properly configure it. There are separate connections available for MBPM, Case360, custom EIS applications, and database tables.

Establishing an internal connection The internal connection which is provided with Process Suite allows you to import entities from either the Process Component Library (PCL) or the Process Platform Inbox.

The PCL entity connection enables access to data stored in the PCL database repository. When you employ the PCL connection, entities are automatically generated from the ToDo and Watch List objects. Attributes of these objects will also be automatically transformed into Property building blocks.

The Process Platform Inbox is a user repository for work items (“To Do Lists”) requiring user input. Whenever a case instance, process instance, or other task requires human input, a task is added to a user’s inbox. These tasks are assigned by user role, team affiliation, or name. The internal connection to the Inbox creates entities from Inbox tasks. The benefit of utilizing this connection is to provide complete integration between entity-based applications and user tasks in the Process Platform Inbox. The connection will automatically generate entities with Properties based upon ToDo Lists.



Properties from a ToDo List entity created from the Process Platform Inbox connection do not support searching or sorting.

Once the internal connection creates entities from either PCL objects or the Process Platform Inbox, you can decorate the entities with building blocks, such as lists, forms, and layouts.

A connection is captured and configured as a new Process Platform EIS connection document in the Collaborative Workspace.

Figure 10-13:
**New Process Platform
EIS connection
document**

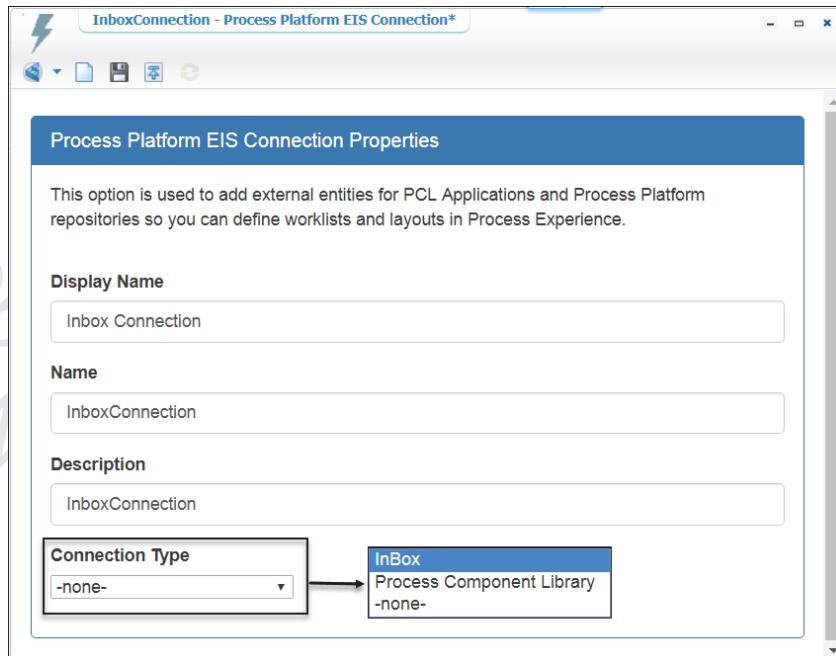
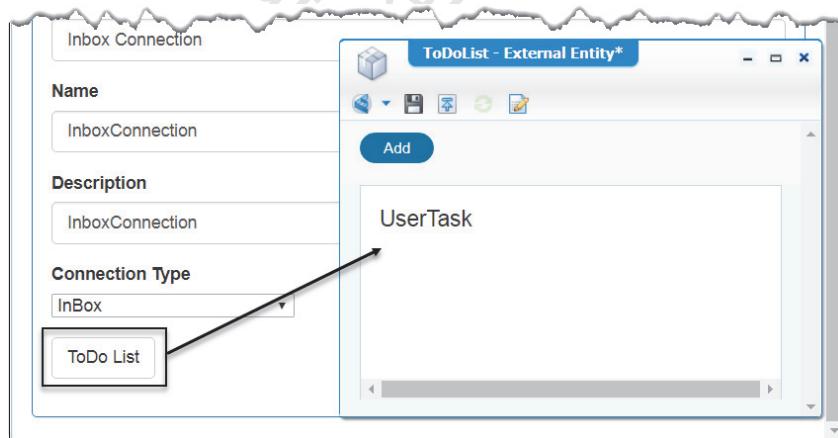


Figure 10-14:
**ToDoList external entity
generated from internal
connection (Inbox)**



**Establishing a
connection to MBPM or
Case360**

Because the Process Platform EIS connection for Process Component Library and the Process Platform Inbox is based upon data held in the Process Platform databases, the connection document does not require specific configuration. If you were to attempt a connection between Process Platform and MBPM or Case360, however, you would need to include configuration options which provide connection-specific information to your instance of either MBPM or Case360. Each of these connections requires a new, different type of connection document in the Collaborative Workspace: either a MBPM or Case360 connection document, respectively.

When you create connection documents for either system, the connection document will automatically create external entities for you based upon the information contained in either system. Depending upon the properties and their data types, you may need to manipulate the external entity in Process Platform.



A Process Platform system administrator must separately install the connection documents for creating external entities from MBPM and Case360.

Custom EIS connections

Often, the connection you must establish is to another system, which may not be an OpenText system. In these cases, you need to establish a connection to this external EIS system in order to import business objects as entities. A custom EIS connector has been provided to accommodate these circumstances.

Using a custom EIS connection is a long process which requires careful planning and thorough testing. The EIS connector framework enables application developers to develop connectors to specific external systems by implementing the interfaces provided by the framework.



To develop an EIS connector, you need a good working knowledge of the Java programming language.

The EIS connector framework provides tools for building a connector to bring information from the enterprise's other information systems as entities into the entity modeling environment and use them in an application.

The target external systems for a custom EIS connector might be enterprise applications (ERPs such as SAP) and repositories (for example, SuccessFactors or SharePoint). All of these systems hold information that can participate in an application either during building of the application or at run time during the execution.

The steps to build an EIS connector and use EIS entities in an application are summarized as follows:

1. In CWS, create a new EIS connector and provide the properties required to connect to the external system.
2. Define the entities and provide the implementation for the Create, Read, Update, Delete and List operations.
3. In the System Resource Manager, manage EIS repository configurations by configuring the properties defined in the connector.
4. Import EIS entities through the EIS repository reference by selecting the appropriate connector and corresponding repository configuration.
5. Decorate the EIS entities with building blocks.

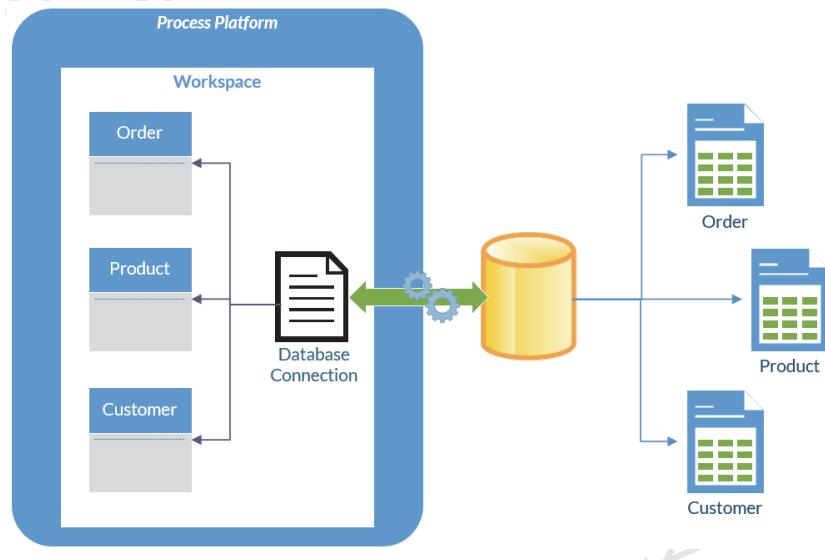
Importing entities from a database

Importing entities from a database is conceptually similar to importing entities from an external source: you must first establish a connection to the database using a connection document in Process Platform, then you can generate entities. The Entity Importer document serves two purposes:

1. Establish a connection to the database. The document will contain all the information necessary to form a connection from Process Platform to the database. This includes the type of database, the database URL, user name, password, database driver location, and name of the database.
2. Generate entities from a database connection. After the database connection has been configured and verified, you can use the Entity Importer to create entities from tables in the database.

After the Entity Importer has generated entities, you can develop solutions by decorating the entities with building blocks such as forms, rules, security, etc.

Figure 10-15:
Conceptual image of entity importer



By default, the Entity Importer works with Microsoft SQL Server, MySQL, and Oracle databases.

Database Metadata Modeler The Entity Importer is related to another Process Platform document: the Database Metadata Modeler. Metadata is “data about data”. Process Platform provides tools to explore the metadata of various back end systems. By exploring metadata, automated service generation on various back ends is supported, which speeds up service development. The Database Metadata Modeler is a document which developers can use to examine the contents of a database as object representations in Process Platform. There are other applications of the Database Metadata Modeler, including establishing a connection to the database so that the processes and cases you develop can read and write data to/from the database tables. Developers can also use the Database Metadata Modeler to wrap the database connections as web services.



You will use the Database Metadata Modeler to read and write to/from a database and wrap the connection as a web service in course 4-4913: Process Modeling for Process Platform.

SQL web service operations use a connection pool to connect to the database. All users share the same database account as they connect to the database (a technique called *impersonification*), therefore Process Platform provides auditing of its users. The web services all have an implementation type of DBSQL, which means that together with the parameters, a SQL statement is, in fact, passed to the connector.

The Database Metadata Modeler allows you to model metadata from database tables, views, or stored procedures. Furthermore, you can choose which of these objects you wish to model as metadata.

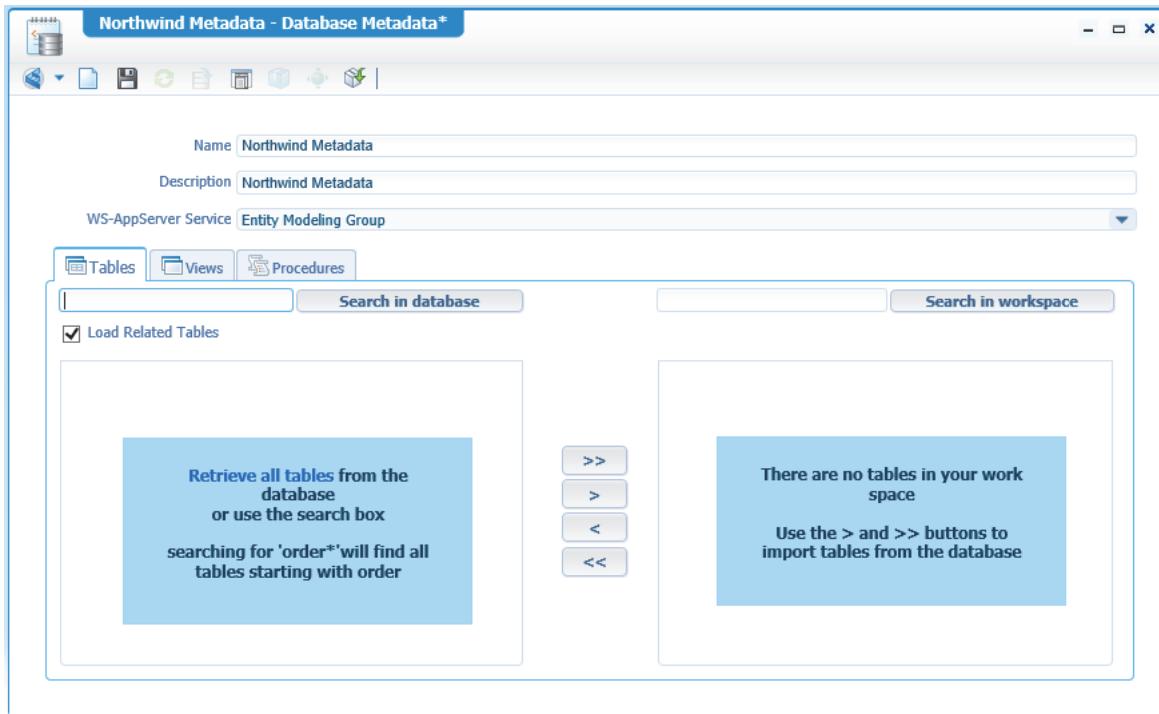


Figure 10-16: Sample Database Metadata Modeling document

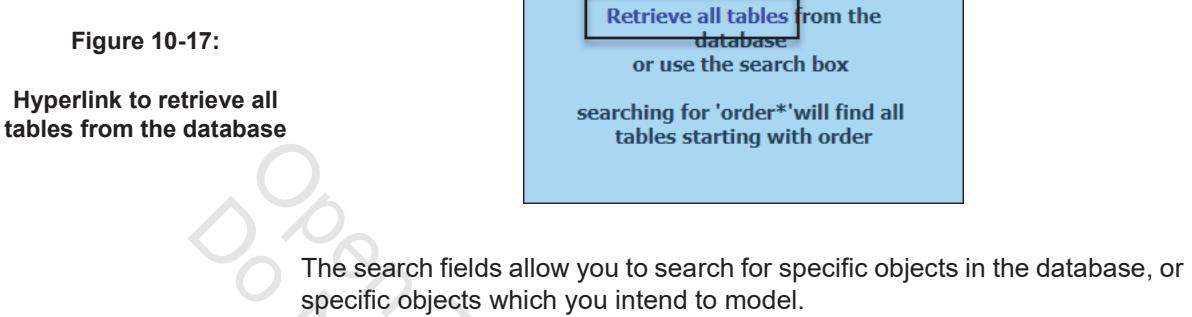
The Database Metadata Modeler document contains information about the document, such as a unique name and a detailed description, a service group, and the objects available and modeled from the database. The service group is a special document which must be built separately. Inside the service group is a service container, which contains all the connection information necessary to connect to the database.



In course 4-4913: Process Modeling for Process Platform, you will learn how to build your own service groups and service containers.

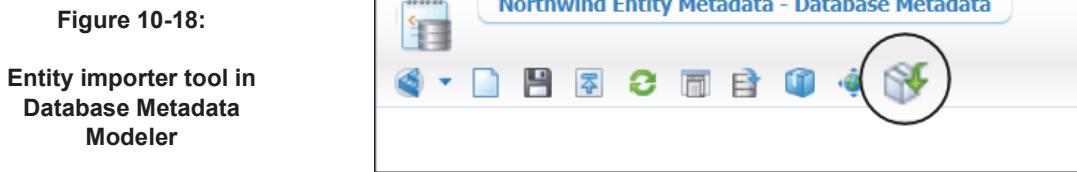
Each tab of the Database Metadata Modeler is divided into two groups: on the left are all the tables, views, or stored procedures which the modeler was able to detect in the database. On the right are all the tables, views, or stored procedures which you choose to model as metadata. To add tables, views, or stored procedures to this list (or remove them from this list), use the arrow keys between the lists to move any or all objects to/from the list of modeled objects.

Before you can add objects to the list of objects for modeling, you must retrieve them from the database. A dialog in the table list contains a hyperlink which will fetch all the tables/view/procedures from the database, and thereby allow you to begin modeling them as metadata.



The search fields allow you to search for specific objects in the database, or specific objects which you intend to model.

Entity importer The entity importer is a tool available in the Database Metadata Modeler. After you have selected tables from the database which you intend to model as entities, you can click the entity importer to start the entity generation process.



The entity importer contains a list of all the metadata objects which have been saved in the Database Metadata Modeler. To create the entities, select the metadata objects from the list and click Finish. The entities will be created automatically.

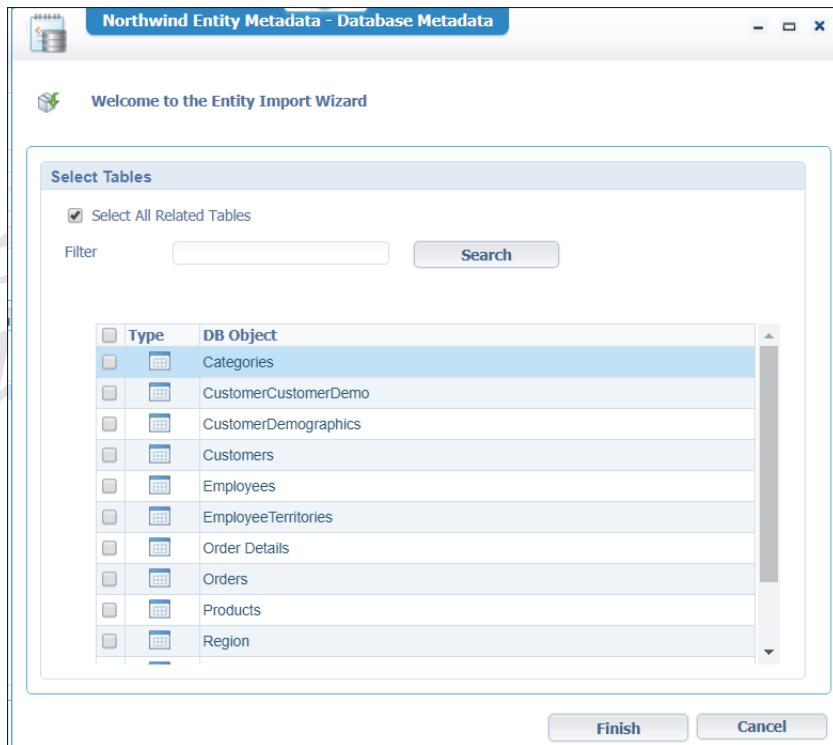


Figure 10-19:

Example of entity importer with sample metadata objects

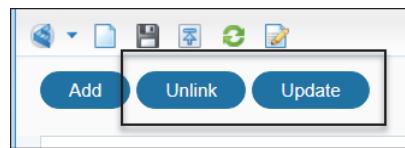


The entity importer only works with metadata objects which have been saved in the Database Metadata Modeler. You must save the objects which you intend to model first before you use the entity importer.

During the import process, the entity importer will attempt to create the entity with a property building block representing each column of the database. The entity importer will attempt to match the datatype in the table column with a matching data type in the property building block. The primary key of the database table will always be realized as an Identity building block, but the name of the column will not be transferred to the entity.

Linked entities When the entity importer finishes generating the entities, it creates the entity as a *linked entity* in the collaborative workspace. A linked entity is an entity representation of the underlying database table which is local to the workspace. If any changes are made to the underlying database table, you must update the linked entity. Similarly, if you don't want the entity to be linked to the database table, you can choose to unlink it.

Figure 10-20:
Functions for linked entities



When a solution which includes a linked entity is published to Process Experience, the connection to the underlying database table is still present; data which is read from or written to the entity instance is stored in the source database table.

It is quite common that, in the initial stages of solution development, teams will assess which business objects they currently have, and generate linked entities from them. In this way, you can start building your solution much quicker, and you can choose to unlink the entities from the source tables, so that you may decorate it with more building blocks.

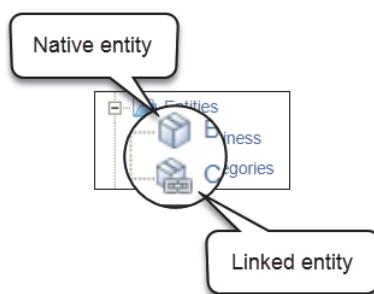


When adding a linked entity into your solution, Process Platform creates a table in its database to maintain the building blocks of the linked entity.

If you choose to unlink the entity from its source table, then the link to the original underlying database will also be severed and cannot be reclaimed: it must be recreated, but cannot be recreated as long as the unlinked entity of the same name is present.

Linked entities inherit their identity and property building blocks from their source tables. Property building blocks also inherit features such as the length of the column (if the datatype is text). If there are foreign keys present in the table, these will be mapped to relationship building blocks. The properties and relationships are named similarly to their table counterparts.

Figure 10-21:
Linked and native entities in the collaborative workspace



When an entity is linked to its source table, you may not change its base table structure. This means that you cannot change building blocks such as properties or relationships, nor can you add new ones. This is because, by nature of the link, changes to properties or relationships would affect the underlying database (as new columns or new foreign keys, respectively). If you want to add any of these building blocks, you must unlink the entity from its source table. Otherwise, linked entities may contain other types of building blocks (lists, forms, layouts, rules, etc.), but not all of them: lifecycle and activity flow are not supported in linked entities.

Included	Excluded
Action bar	Activity flow
Form	Assignee
History	Business workspace
Layout	Content
List	Deadline
Rule	Discussion
Security	Email / Email template
Web service	File
	Lifecycle
	Mobile App
	Property
	Relationship
	Title
	Tracking



Generate database metadata and import tables as entities

1. In the Northwind Workspace, right-click the Warehouse Application and select New > Folder.
2. Name the folder: **Metadata**.
3. Right-click the Metadata folder and select New > Other.
4. Select **Database Metadata**.

You can type the name of the document in the search bar to filter the documents by name.

5. Set the Name of the metadata document to **Northwind Metadata**.

The description will automatically be set to the same name.

6. From the WS-AppServer Service drop-down, select **Entity Modeling Group**.



This WS-AppServer Service Group has been built for you.
Ordinarily, you would need to build this service group yourself.

7. If it is not already selected, select the **Tables** tab.
8. Click the **Retrieve all tables** hyperlink in the left list to fetch the tables from the database.
9. Click the double arrow (>>) to add these tables to the workspace.
10. Save the Database Metadata Modeler document.

The system will automatically begin to model the database table as a metadata object.

11. Click the **Import Entities** button from the toolbar.
12. Clear the **Select All Related Tables** option.
13. Select the following tables from the list to model as entities:
 - Categories
 - Employees
 - Region
 - Shippers
 - Suppliers
 - Territories
14. Click **Finish**.
15. Close the Database Metadata Modeler document.
16. In the Northwind Workspace, expand the **Metadata** folder.

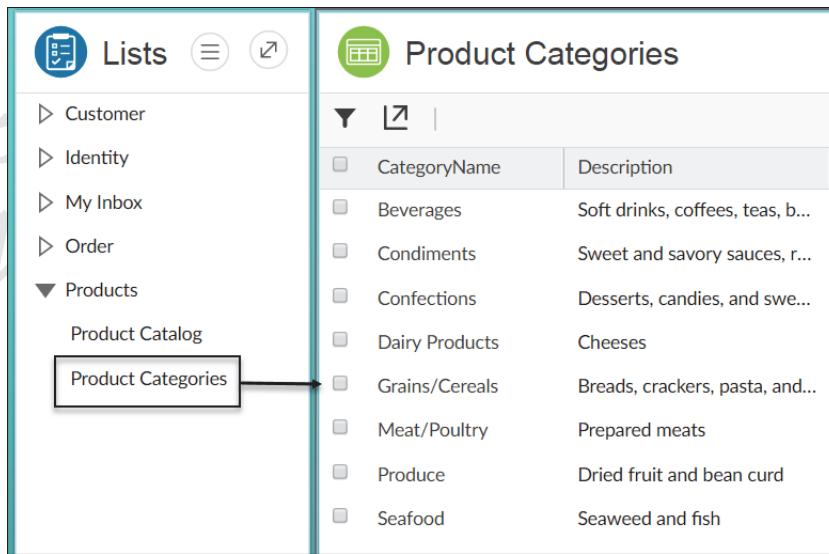
Your new entities have been added in this folder.

17. Drag each new imported entity from the **Metadata** folder and drop it in the **Entities** folder.
18. Double-click the **Categories** linked entity to open it in the entity editor.
19. Click **Add > List**.
20. Set the **Display Name** to **Product Categories** and **Name** to **viewCategories**.
21. Set the **Default Category** to **Products**.
22. Click **Add**.
23. Click **Configure**.
24. Select the **CategoryName** and **Description** for the list.

25. Save and close the list.
26. Save and close the entity.
27. Validate and publish your solution to Process Experience.
28. Return to Process Experience and refresh the browser.

You should have a new list under the Products category which is linked to the corresponding database table.

Figure 10-22:
Categories linked entity published with list



Use linked and native entities together in a solution

1. Open the Product entity from the Warehouse Application.
2. Click **Add > Relationship**.
3. Set the Display name to **Categories** and Name to **oneCategory**.
4. Set the Type **To peer** and Multiplicity **To one**.
5. Click **Browse** for the Target entity.
6. Select **Categories** and click **OK**.
7. Click **Add**.
8. Select the subformProduct form.
9. Click **Configure**.
10. Under the existing Category field, drag and drop the **[0..1] Categories** relationship.
11. Change the type to a **Drop list**.
12. Select **Product Categories** as the Browse list.
13. Select **CategoryName** as the Property.
14. Change the label to **Category**.
15. Clear the Create action.

16. Save and close the form.

Because this subform is used in other forms and layouts, the changes will propagate throughout your solution.



This relationship will replace the Category property in the Product entity. Until you have updated all of your existing products to use this new relationship, keep both the property and the relationship on your form. Once you've made changes to all of your existing products, you can delete the Category property from Product.

17. Select Security.
18. Click **Configure**.
19. Select the *Product Manager* role.
20. Locate the relationship in the security configuration editor and enable **Read** and **Update** permissions.
21. Save and close the security configuration editor.
22. Validate and publish the Warehouse Application.
23. Refresh Process Experience and try to create a new product to determine if your new category relationship is present and if it is accessing the underlying linked entity.

Figure 10-23:
Fields for property and
relationship to linked
entity

Product Number	* Name
1	Daim Orange
Category	Price per Unit
Confections	\$3.99
Category	
- Select - - Select - Beverages Condiments	

Summary

Having completed this chapter, you should be able to:

- Add files to entity instances in Process Experience
- Describe web services and their role with entities
- Explain web service operations exposed on entities and relationships
- Expose an entity as a web service
- Explain external entities
- Import entities from database tables

Exercises

1. Create web service wrappers for Customer and Order entities.

The web service wrappers should use:

- Customer: Create, Read, and Update basic operations
- Order: Create, Read, and Update basic operations
- Order > childOrderLine: Create, Get, and Delete operations
- Order > OrderToCustomer: Set operation

2. Build “find” operations for the following web services:

- Customer: find customer by client code
- Order: find all outstanding (unshipped) orders
- Order: find orders after a certain date

3. Replace the Employee ID property in the Order entity with a relationship to the Employees linked entity.

Hint: Build list(s) so you can use the linked entity in a form. What type of list will you build?

Hint: Change your form(s) so they use the relationship instead of the property.

Hint: Consider changes to your rule(s) which refer to the Employee property.

4. Replace the Ship Via property with a relationship to the Shippers linked entity.

Hint: Build list(s) and change form(s).

Hint: Consider changes to your rule(s) with the Ship Via property.]

5. Commit your changes to the repository with the description: “Chapter 10: Advanced”

Advanced: Testing web services in Process Platform

6. Follow these steps to test the web service operations you created this chapter:

- In the applications palette, open the System Resource Manager application.
- Click Show > All Service Groups.
- Right-click the Entity Modeling Group and select New Service Container.
- Select Application Server Connector and click Next.
- Set the name to Entity Web Service Connector and the Startup Type to Automatic.
- Enable the Assign OS Process option and select Application Server from the drop-down as the assigned process.
- A pop-up dialog will remind you to set the JRE parameters. Click Close.
- Click Next.
- You do not need to provide details for the Application Server Connector. Click Next.
- Set the connection point name to: Entity Web Service Connection Point.
- Click Finish.
- Right-click the Entity Modeling Group and select Properties.
- In the Web Service Interfaces table, click insert (+).

- n. Select the **warehouse** organization in the Organization/Package field. Your entity web services should be listed here.
- o. Select each entity web service interface you want to test (Method Set Entity Customer, Order, and/or Product) and click Add.
- p. After adding all your entity web service interfaces, click Done.
- q. Save the service group.
- r. Right-click the Entity Web Service Connector and select Restart.
- s. After the Entity Web Service Connector starts successfully, close the System Resource Manager.
- t. In the applications palette, open the Web Service Interface Explorer application.
- u. Type 'product' in the Keyword field and click Find. If you added the Product web service interface to your service group, then the web service operations for the Product entity should be listed here.

The screenshot shows the 'Web Service Interface Explorer' application window. At the top, there is a toolbar with icons for creating a new service, deleting, and saving. Below the toolbar, the title bar reads 'Web Service Interface Explorer' and 'Manage web service interfaces and web service operations'. The main area has a search interface with a 'Keyword' input field containing 'product', a 'Search Criteria' dropdown set to 'Web Service Operation', and two checkboxes for 'Starts With' and 'Ends With'. Below the search controls, a message says 'Search Results (6 Web Service Operations found.)'. A table lists six operations:

Web Service Operation	Web Service Interface	Application/Organization	Service Group
CreateProduct	Method Set Entity Product	warehouse	Entity Model
FindActiveProducts	Method Set Entity Product	warehouse	Entity Model
FindProductsByStockCeiling	Method Set Entity Product	warehouse	Entity Model
Product.xsd	Method Set Entity Product	warehouse	Entity Model
ReadProduct	Method Set Entity Product	warehouse	Entity Model
UpdateProduct	Method Set Entity Product	warehouse	Entity Model

- v. Right-click an operation and select Test. This example will demonstrate the ReadProduct operation. The Operation Test Tool dialog will open with a pre-built SOAP request message.
- w. Replace the PARAMETER value in the SOAP request with a valid product ID (you can find valid product IDs in the appropriate list of Process Experience).

- x. Delete the line which contains the ItemID parameter.



- y. Click Invoke. The test tool will send the SOAP request to the server and should receive a SOAP response.
 z. Double-click the Response Message to view it.
 aa. Examine the contents of the response message. You should be able to locate your properties and values.

```
<data>
<wstxns1:ReadProductResponse xmlns:wstxns1="http://schemas/NorthwindWarehouseApplication/Product/operations" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:wstxns1="http://schemas/NorthwindWarehouseApplication/Product" xmlns="http://schemas/NorthwindWarehouseApplication/Product" xmlns:wstxns2="http://schemas/NorthwindWarehouseApplication/Product">
<Product-id xmlns="http://schemas/NorthwindWarehouseApplication/Product">
<Id>16385</Id>
<ItemId>000C2915C91A1E88062437C95EEC20C.16385</ItemId>
</Product-id>
<ProductName xmlns="http://schemas/NorthwindWarehouseApplication/Product">Al-Habib Roasted Garlic Hummus</ProductName>
<CategoryID xmlns="http://schemas/NorthwindWarehouseApplication/Product"></CategoryID>
<QuantityPerUnit xmlns="http://schemas/NorthwindWarehouseApplication/Product">Case of 48</QuantityPerUnit>
<UnitPrice xmlns="http://schemas/NorthwindWarehouseApplication/Product">4.9500</UnitPrice>
<UnitsInStock xmlns="http://schemas/NorthwindWarehouseApplication/Product">UnitsInStock</UnitsInStock>
<UnitsOnOrder xmlns="http://schemas/NorthwindWarehouseApplication/Product">48</UnitsOnOrder>
<ReorderLevel xmlns="http://schemas/NorthwindWarehouseApplication/Product">ReorderLevel</ReorderLevel>
<Discontinued xmlns="http://schemas/NorthwindWarehouseApplication/Product">false</Discontinued>
<ShipmentReceived xmlns="http://schemas/NorthwindWarehouseApplication/Product">true</ShipmentReceived>
<ReceiveComments xmlns="http://schemas/NorthwindWarehouseApplication/Product" xsi:nil="true" />
</data>
```

- ab. Close the operation test tool window when your testing is complete.
 ac. You may test any operation using the Web Service Interface test tool, but be mindful that any Create operation will insert new records into the data store.
 ad. Note: you should test any Find operations to ensure that they are executing as designed.



Tips, tricks, and traps

1. Use web services to interact between entities

As you have seen in this course, building blocks govern and dictate the functionality of the entity to which they're added. For example, you cannot (easily) write a rule in the Person entity which will set values in the Order entity. For the most part, entities are self-contained. Interaction between entities requires choreographing activities.

- Wrap your entities with web service building blocks.
- Build a business process model.



You will learn about business process models in course 4-4913: Process Modeling for Process Platform.

- Choreograph the interaction between entities by calling the proper web service operations in the appropriate order as activities of the business process model.
- Trigger the business process model in a rule or lifecycle.

2. Wrap linked entities to use lifecycle and other building blocks

As you have learned with linked entities, you cannot add some building blocks to them, such as activity flow and lifecycle. This can be problematic if you want to impose stages of execution on a linked entity.

- Create an entity to serve as the wrapper of a linked entity.
- Build a “to one” relationship in the wrapper to the linked entity.
- Add the lifecycle, activity flow, or other building block in the wrapper.
- Write rules or business processes to effectuate changes from the wrapper to the linked entity.

Open Text Internal Use Only
Do Not Distribute