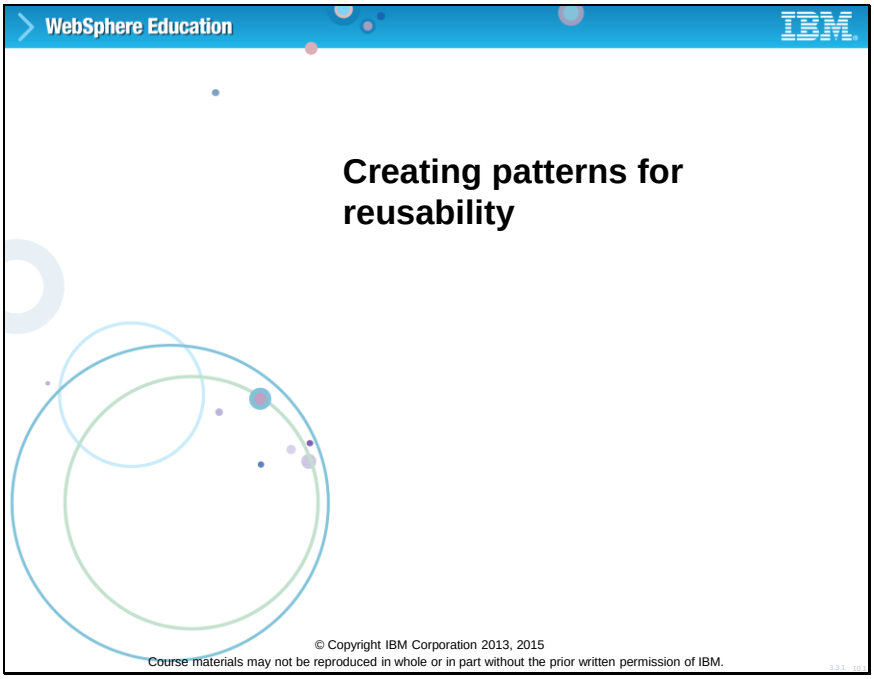
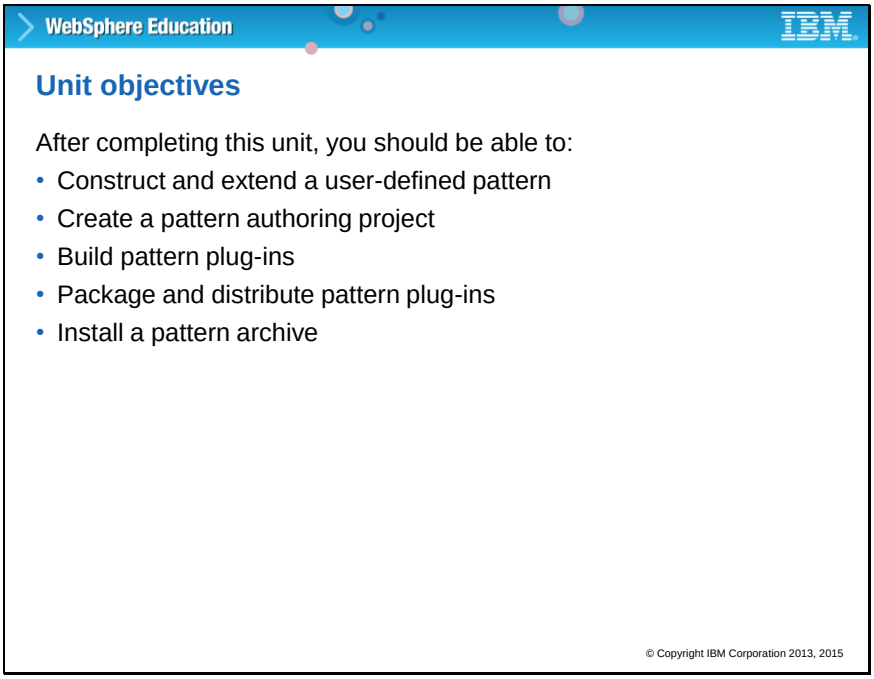
Slide 1


WebSphere Education
IBM
Creating patterns for reusability
© Copyright IBM Corporation 2013, 2015
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Slide 2

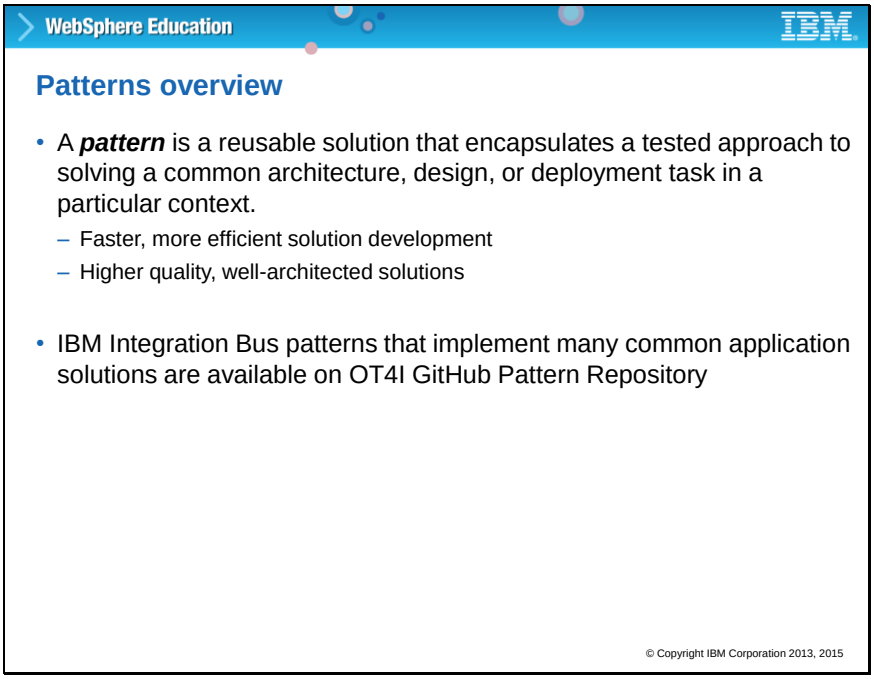


Unit objectives

This unit describes how to create and test user-defined message flow patterns in the IBM Integration Toolkit.

After completing this unit, you should be able to:
- Construct and extend a user-defined pattern
- Create a pattern authoring project
- Build pattern plug-ins
- Package and distribute pattern plug-ins
- Install a pattern archive

Slide 3



Patterns overview

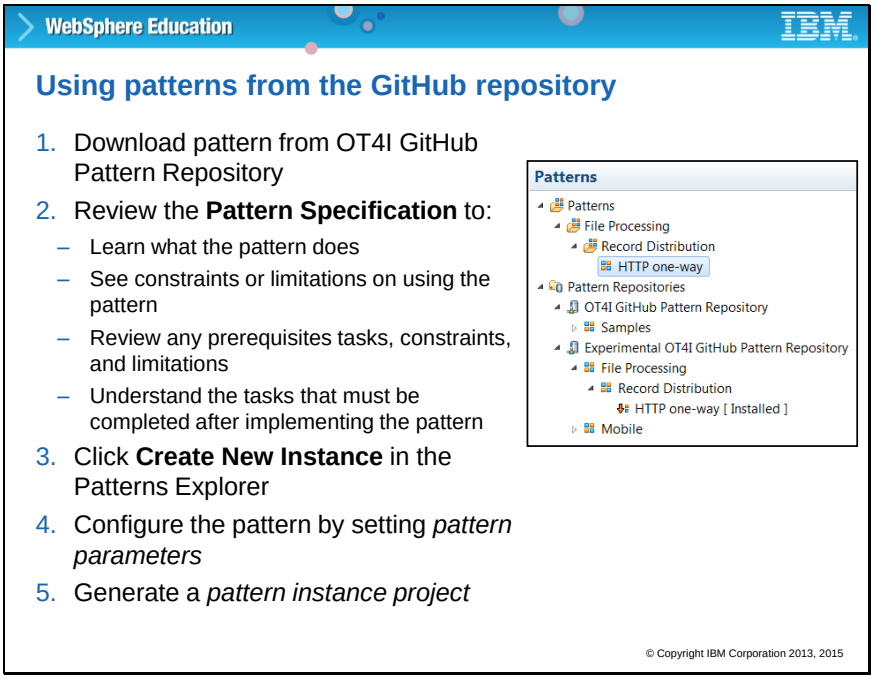
A pattern captures a tested solution to a common application problem or specific business problem.

When you start your Integration Bus development by using a pattern:
- Your applications are based on well-designed solutions that are analyzed and optimized.
- You can decrease your development time because you are not creating the project “from scratch”.
- You can learn preferred techniques for developing application solutions.

Patterns-based development takes advantage of common practices by standardizing message flows. It eases some of the anxiety that new users can experience when presented with so many node choices, and it gives a quick start to message flow design. A pattern reduces common problems by providing templates that solve common problems and promote good practices.

A catalog of Integration Bus patterns is available on an external GitHub repository. You select and download the pattern that meets your development needs to your Integration Toolkit.

Slide 4



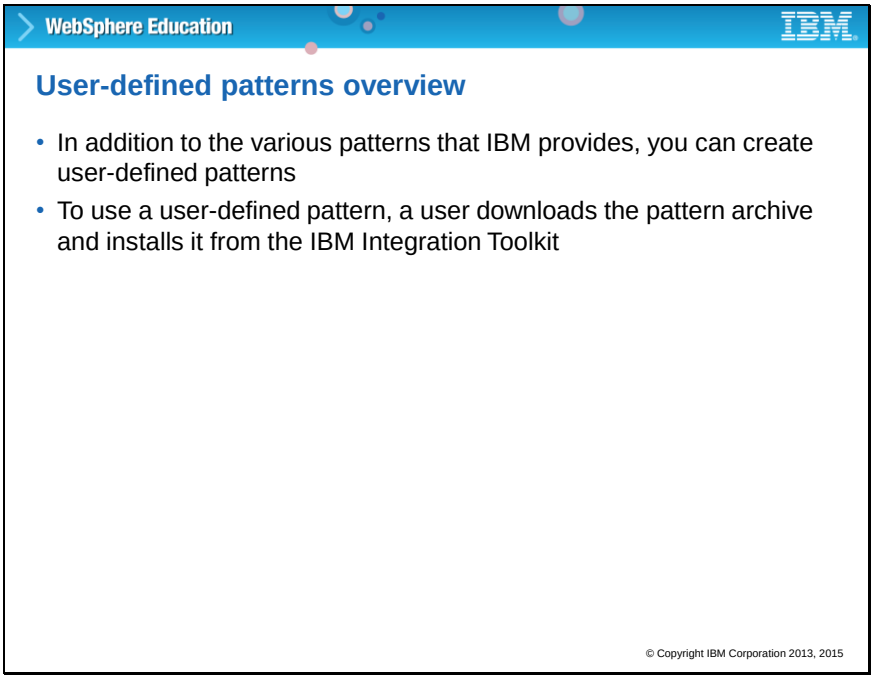
Using patterns from the GitHub repository

The Patterns Explorer lists the available patterns in the GitHub repository and those patterns that were downloaded to the Integration Toolkit.

When you elect to implement a pattern, you follow a standard defined sequence of steps. Finally, these steps include creating an instance of the template pattern, then configuring that instance, and generating the pattern instance. This slide lists the steps for getting patterns from the GitHub Pattern Repository and that uses the pattern to create a pattern instance.

These steps are used for built-in patterns and user-defined patterns after they are installed in the Patterns Explorer.

Slide 5



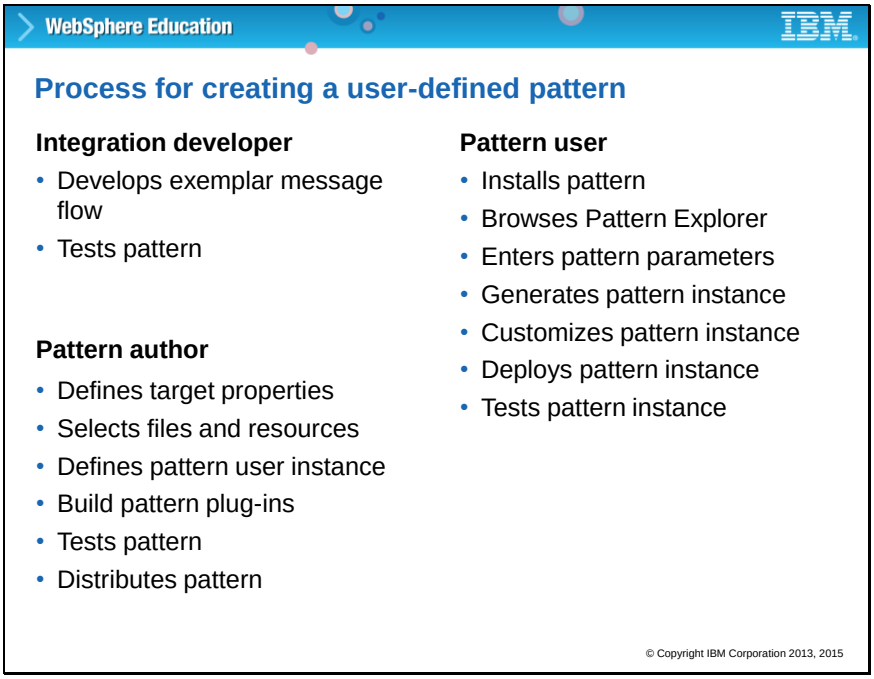
User-define patterns overview

Like many components of Integration Bus, the concept of patterns is extensible. In addition to taking advantage of a number of built-in patterns, you can also create user-defined patterns.

After you create a user-defined pattern, you can package and distribute it so that another developer can download the user-defined pattern to their Toolkit installation.

User-defined patterns can help you to extend good practices within your organization by providing predefined templates from which to start Integration Bus application development.

Slide 6



Process for creating a user-defined pattern

The Integration Bus developer, a pattern author, and a pattern user complete the three stages of creating a user-defined pattern.

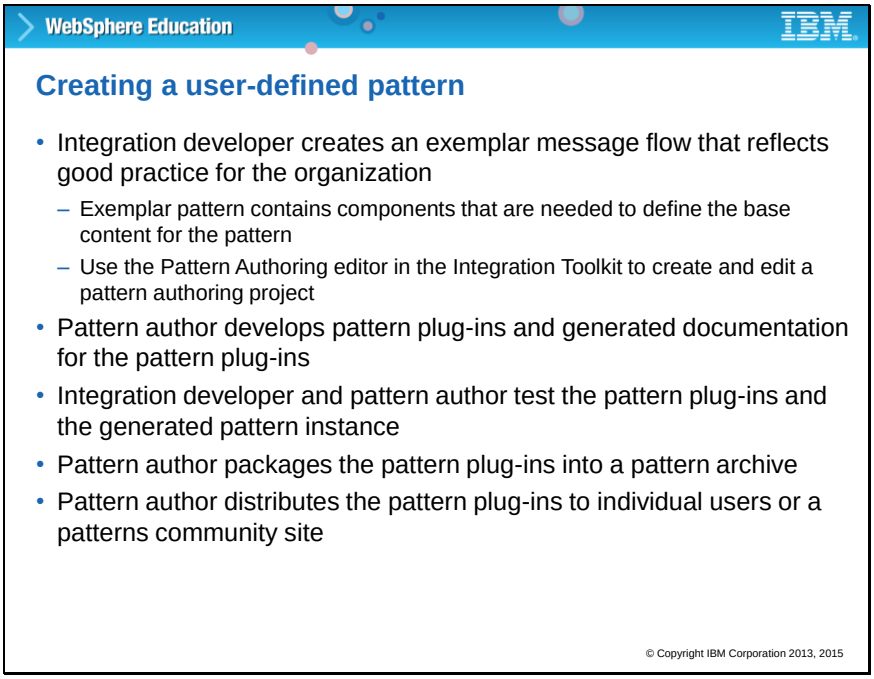
The Integration Bus developer develops the exemplar message flow. The exemplar is fundamental to the pattern authoring process as it defines the base message flow and is the starting point for a pattern.

The pattern author creates a pattern plug-in from the exemplar. Some of the pattern authoring tasks focus on the resources in Integration Bus. For example, one step in authoring a pattern is defining the target properties for the pattern and configuring the user interface that is presented to the pattern user.

The pattern author ensures that the pattern users can customize the pattern as needed. If the pattern user regenerates an instance of the pattern, the pattern author also ensures that these customizations are not overwritten. The pattern author then shares the user-defined pattern with the pattern user.

The pattern user is the Integration Bus developer that receives a user-defined pattern, generates a pattern instance, and uses it in accordance with the requirements of the organization.

Slide 7



Creating a user-defined pattern

The process of developing a user-defined pattern follows these basic steps:

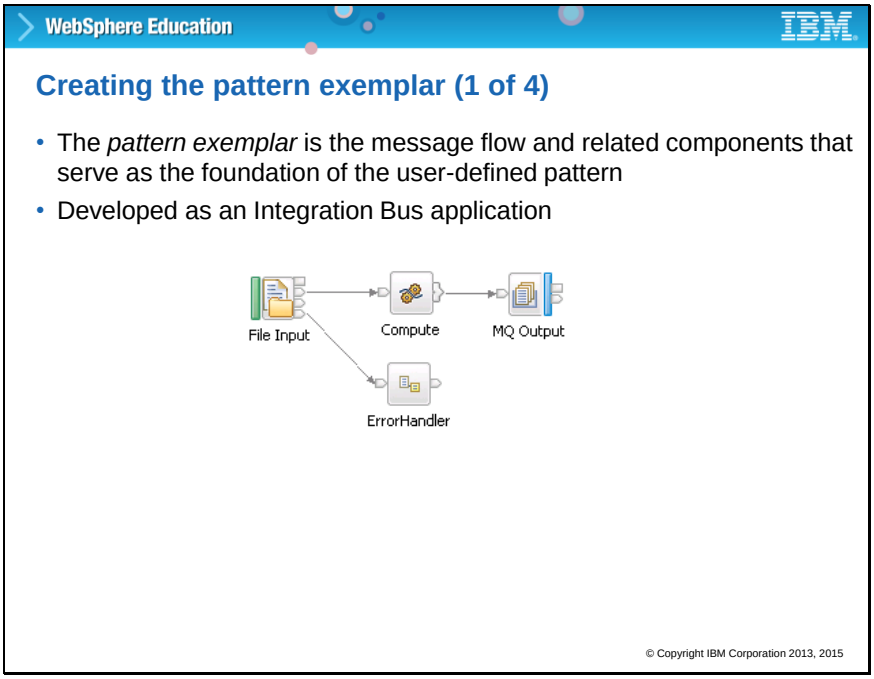
- The application developer works with the *pattern author* to develop an *exemplar* pattern. An exemplar pattern contains message flows, message maps, ESQL modules, Java classes for JavaCompute nodes, XML files, and any other components that are needed to define the base content for the pattern. The exemplar pattern should reflect good practices for the organization, since it serves as the basis for any instantiations of pattern instances.

- The pattern author reviews the exemplar pattern and determines which elements of the exemplar should be customizable by a user of the pattern. The developer and pattern author typically determine jointly which of the parameters should be customizable and how the pattern user should be able to configure them. The pattern author creates a pattern authoring project in the Integration Toolkit. After defining all the pattern plug-ins, the pattern author generates the pattern plug-ins. The Integration Toolkit creates a project file that contains the plug-in content.
- The integration developer and the pattern author test the pattern plug-in file. This testing allows the author to see the same image that a user sees after the pattern plug-in is installed on the user workstation.

- After the author is satisfied with the user-defined pattern, the author creates a *pattern* archive. The archive contains the pattern plug-ins and all the elements of the user-defined pattern.

- The pattern archive is made available to users. The archive can be sent to a user, or posted on a shared file system or on a website (such as a pattern community website) for downloading.

Slide 8



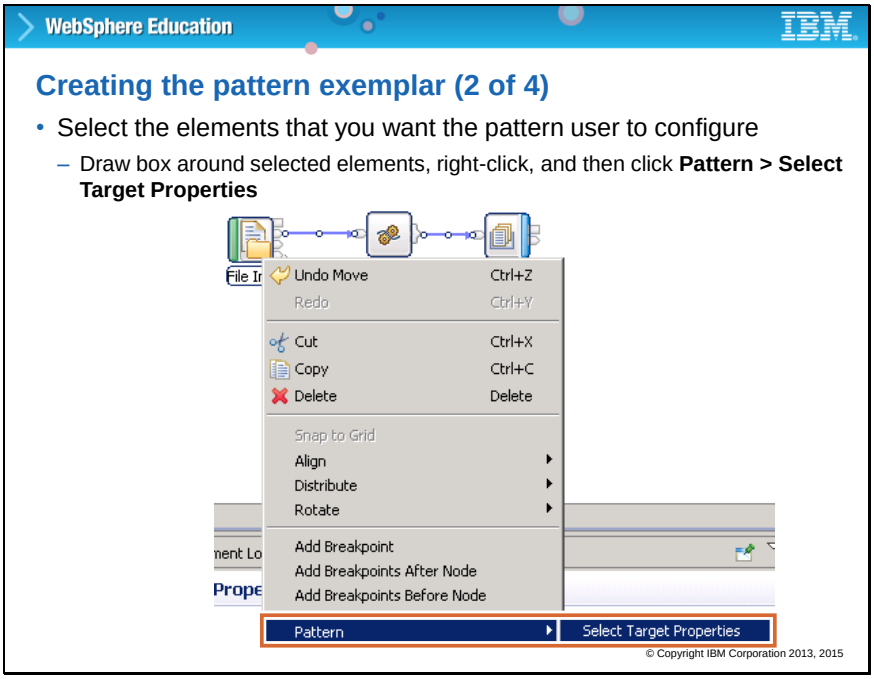
Creating the pattern exemplar (1 of 4)

The first step in developing a user-defined pattern is to create the message flow and related components to serve as a foundation for the pattern. The exemplar can be as simple or as complex as needed for your specific application but it should be a proven solution.

The Integration Bus developer develops the exemplar, usually with the pattern author, so that the expectations of the user-defined pattern are clearly understood.

In this example, the pattern exemplar consists of a File Input node, a Compute node, an MQ Output node, and an error handling subflow.

Slide 9



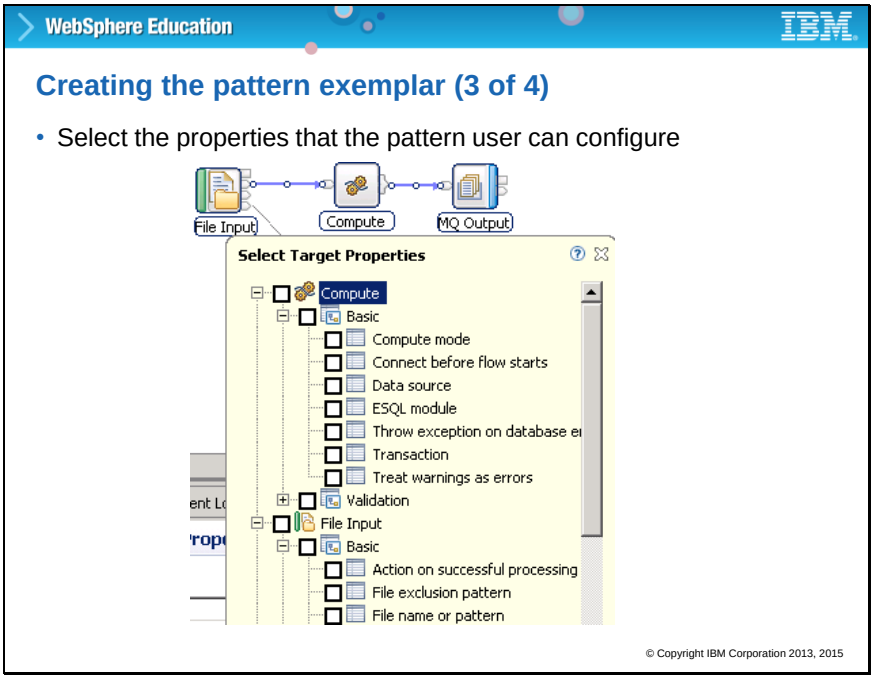
Creating the pattern exemplar (2 of 4)

The next step is to define the message flow properties that the pattern user can configure at pattern instantiation.

First, select the nodes that you want to make user-configurable by dragging a box around the nodes. The node outlines and wires turn blue.

Then, right-click a node or wire and click **Pattern > Select Target Properties**.

Slide 10

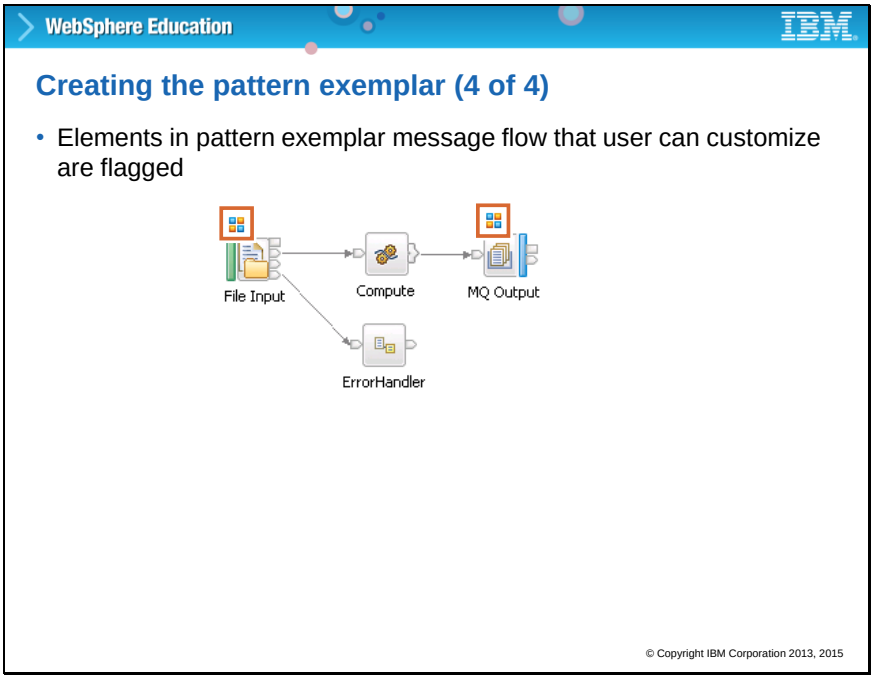


Creating the pattern exemplar (3 of 4)

When you click **Select Target Properties**, a dialog box is displayed. It contains a list of all the configurable properties for the selected objects.

Select the properties that you want the pattern user to be able to alter when the pattern is instantiated. When you are finished, close the dialog box by clicking the **X** in the upper right corner.

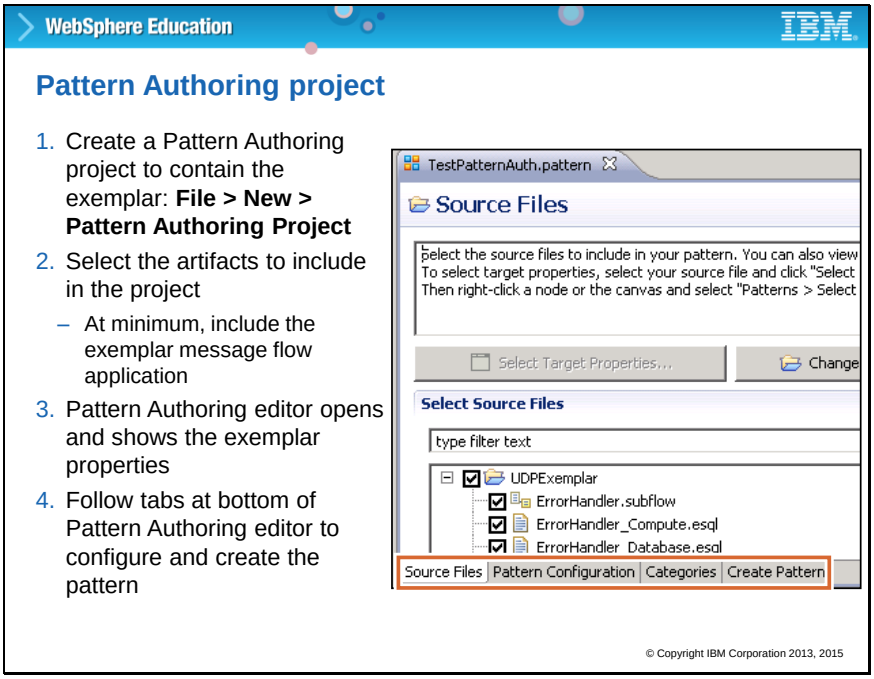
Slide 11



Creating the pattern exemplar (4 of 4)

After you finish defining the configurable properties, the exemplar is updated with a “decorator” above each node in the exemplar message flow that contains user-configurable properties.

Slide 12

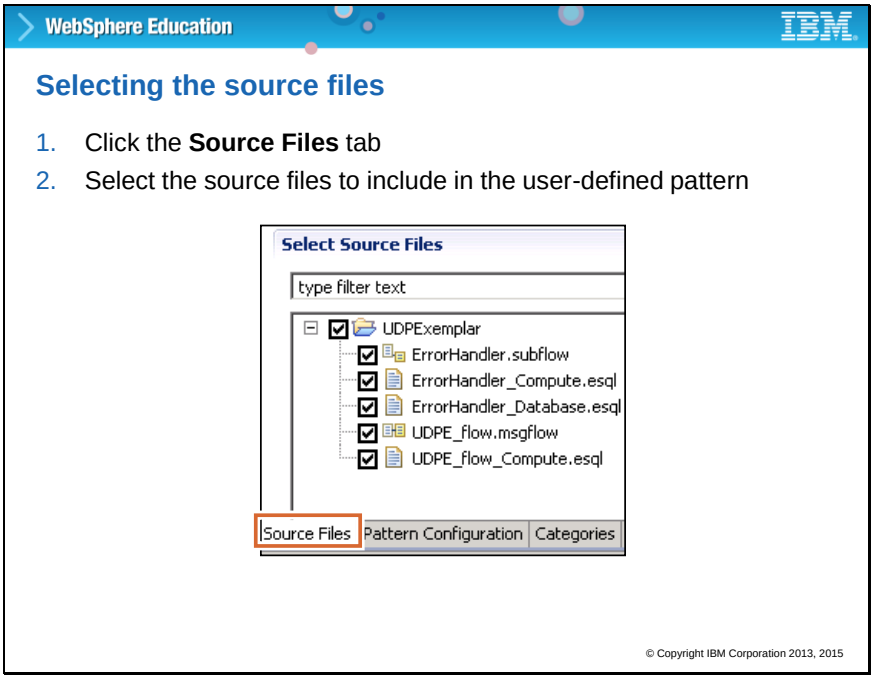


Pattern Authoring project

After the exemplar is created, the work of authoring the user-defined pattern begins. You start by creating a Pattern Authoring Project in the Integration Toolkit.

The tabs in the Pattern authoring editor guide you through pattern development.

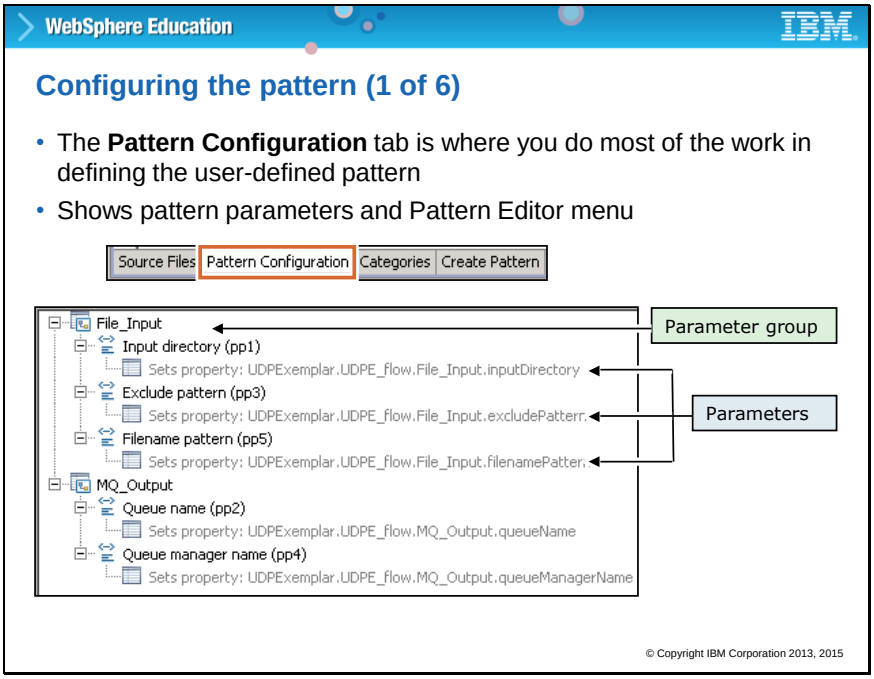
Slide 13



Selecting the source files

In the first tab section of the Pattern authoring editor, you select the source files from the exemplar that you want to include in the final user-defined pattern. By default, all components are included. Before you elect to omit one or more files, consider the effect on the final project (for example, whether you want to include ESQL code for the pattern user).

Slide 14



Configuring the pattern (1 of 6)

The next step of pattern authoring is typically where you spend most of your time when developing a user-defined pattern: pattern configuration.

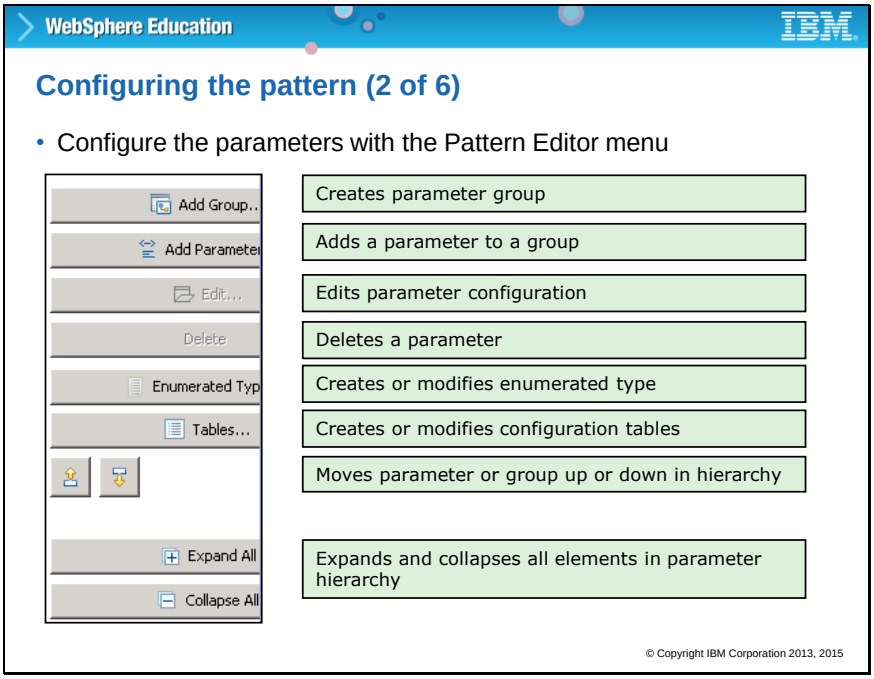
In the pattern authoring project, the pattern author builds the user interface with which the pattern user interacts. The user interface specifies the configurable pattern parameters, and documents the parameters so that the user knows how to set the appropriate values at pattern instantiation time. Parameters can be defined as mandatory, optional, or hidden, and can be presented to the user in a number of formats such as a list, radio button, or a check box. The author can create complex transforms on parameters, including checking the content of parameter fields and doing conditional processing that is based on those values.

The parameters can be grouped in to logical groupings so that they are presented together to the user. The pattern author defines the field labels and field prompts that are available to the user, with help text to guide the user. The pattern author also defines the pattern name, pattern category, and help text that is shown when the user installs the pattern into the Patterns Explorer. The author can also define the version number and the identifying information for the pattern. All of these artifacts become the pattern plug-in.

Patterns are associated with groups. By default, these groups are the nodes with which they are associated in the pattern exemplar. For example, the screen capture shows that **File_Input** is a parameter group that contains the **Input_directory**, **Exclude_pattern**, and **Filename_pattern**

parameters. These parameters correspond to the configurable parameters that were selected for the FileInput node in the pattern exemplar. You can create, delete, and modify parameters and parameter groups in the pattern authoring editor. You can also associate (and disassociate) parameters with groups. You can also move parameters and groups within the displayed hierarchy.

Slide 15



Configuring the pattern (2 of 6)

This slide shows the Pattern editor menu, which is displayed in the right side of the Pattern editor window.

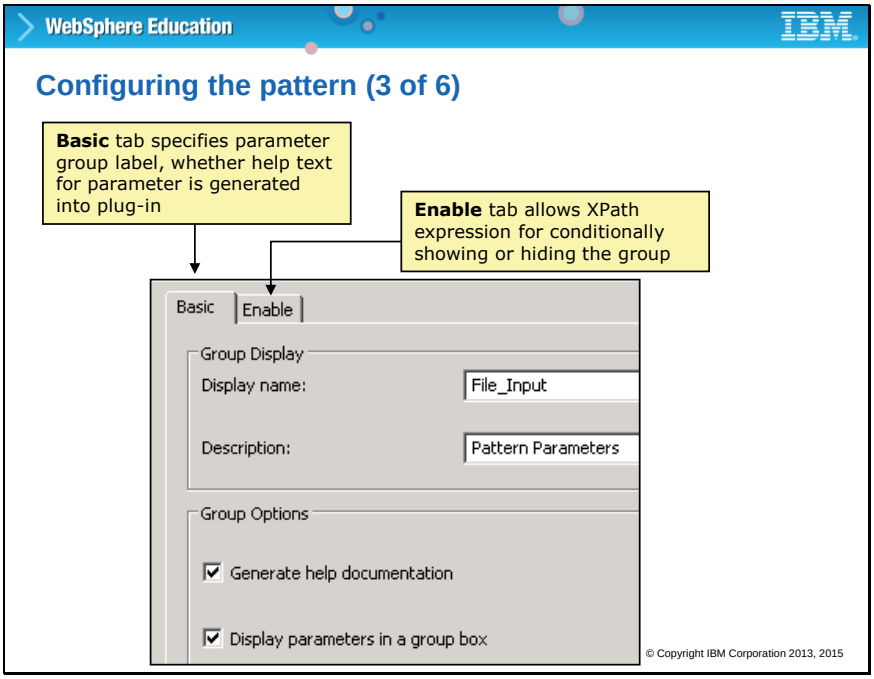
The pattern author uses the Pattern editor menu to configure the pattern parameters. As you see on this slide, you can create parameters and parameter groups, edit parameter configurations, and move parameters and groups within the hierarchy. You can also define enumerate types and tables to use in parameter configuration.

These functions (and others) are also available by right-clicking a parameter or group.

As an option, you can define a pattern parameter as an enumerated type or a table.

- An **enumerated type** is a set of values that a parameter accepts. It consists of a type name and a list of display names and values.
- A **table** contains a collection of rows that the pattern user can configure. Each row in the table contains one or more columns of data.

Slide 16



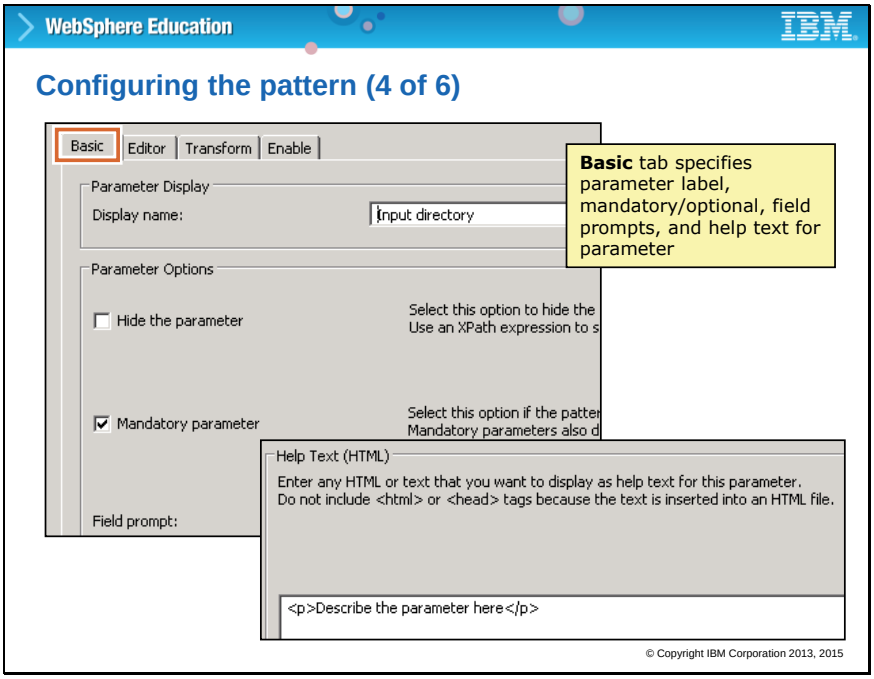
Configuring the pattern (3 of 6)

When you edit a parameter group, two tabs are displayed.

When you select the **Basic** tab, you can edit the characteristics of the parameter group, including the name and description. You can also control whether the help documentation for the parameter group is included with the pattern documentation.

On the **Enable** tab, you can conditionally display or hide the parameter group, depending on the result of an XPath expression. You specify an XPath expression in the expression editor. If the expression evaluates to true, the pattern group is enabled and is shown to the pattern user. If the expression evaluates to false, the pattern group is disabled and the pattern user cannot see it. You can select the value of any other parameters that are defined in the pattern project to use in the XPath expression. This conditional processing capability is a flexible way to control the appearance of the pattern when the pattern user instantiates it.

Slide 17

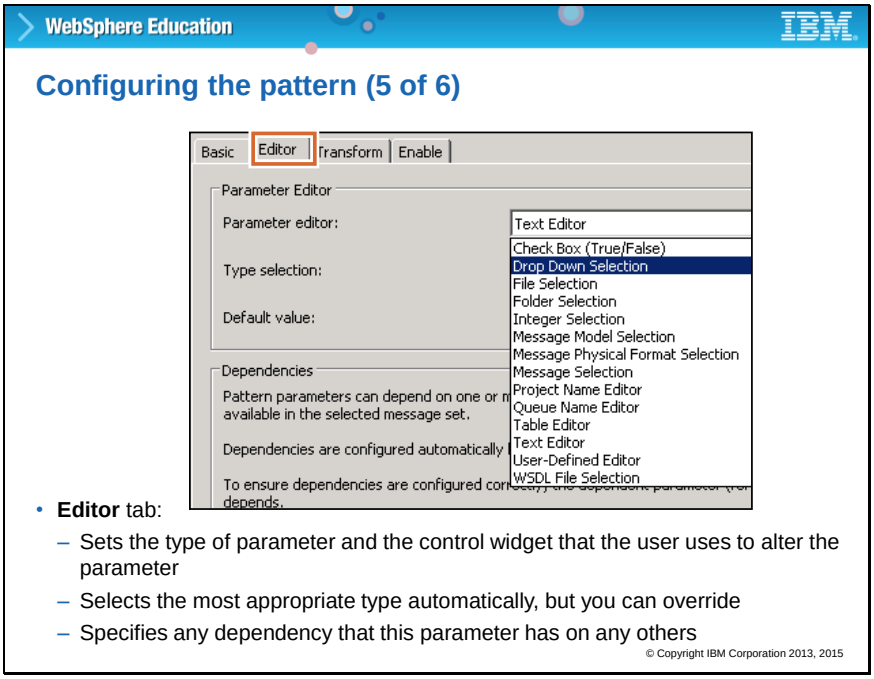


Configuring the pattern (4 of 6)

When you select a parameter to edit, four tabs are presented in the editor: **Basic**, **Editor**, **Transform**, and **Enable**.

On the **Basic** tab, you set parameter attributes such as the display name, whether the parameter is mandatory, and whether the parameter is hidden. You also set the field prompt, which is displayed in the parameter field itself at instantiation time. Also, you can create help text for the parameter by writing HTML. When you format the help text, a secondary window shows you how the help text is rendered to the pattern user.

Slide 18



Configuring the pattern (5 of 6)

The **Editor** tab is where you control the specifics of the user interface for the parameter as it is shown to the pattern user. If the parameter type allows it, these attributes include the type of editor that is used to modify the field and provide an optional default value for the parameter.

The Parameter editor is based on the parameter type. In most cases, the Pattern authoring editor automatically selects the most appropriate parameter editor, depending on the parameter type, but you can override this value.

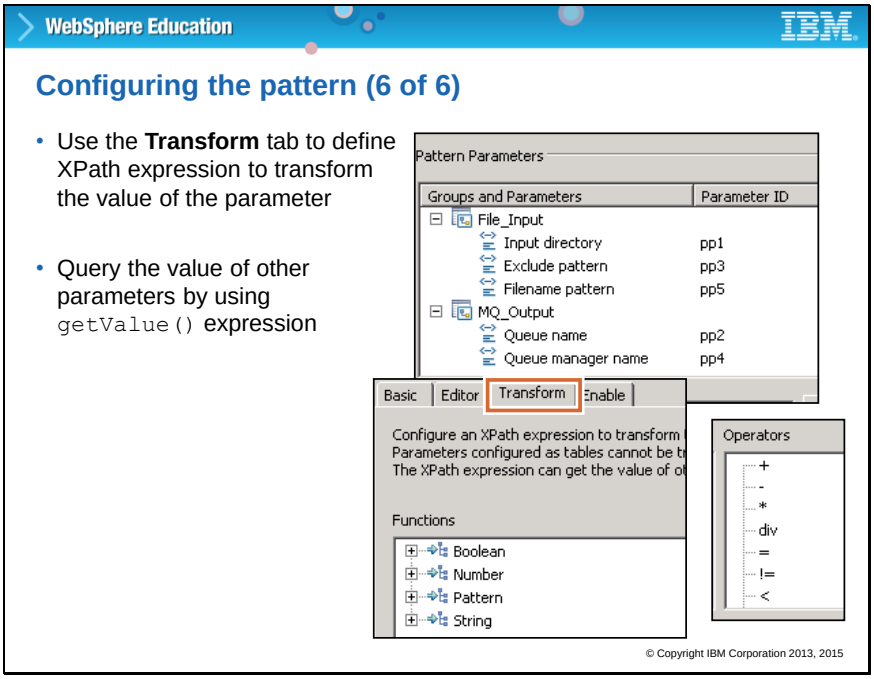
A number of parameter editors are available. For example, you can define a check box option, drop-down list, a file selection menu, or text editor.

Most of these editors contain configurable properties to assist the pattern user. For example, if you choose the **File Selection** editor, you are prompted to provide a list of file extensions that are displayed in the field prompt for the pattern user.

You can also create user-defined editors that are specific to the parameter type.

On the **Editor** tab, you can also view dependencies between parameters. For example, the editor that is used for a parameter can depend on the editor that is used for a second parameter, or on its values. The pattern authoring editor manages these dependencies for you, but the dependencies are listed here for your reference.

Slide 19



Configuring the pattern (6 of 6)

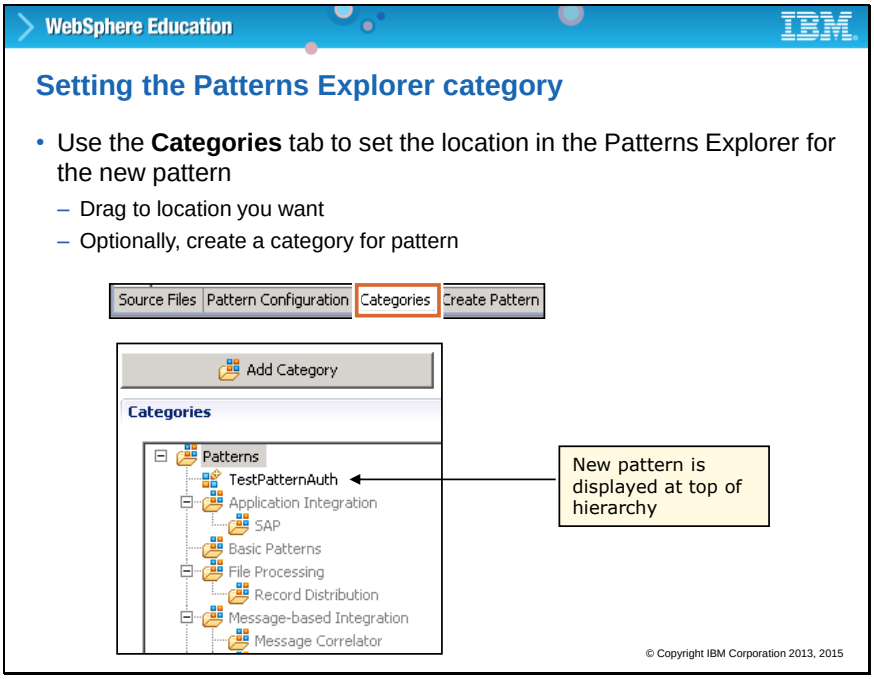
You use the **Transform** tab to define expressions to transform the value of a parameter. When the pattern user creates the pattern instance by clicking **Generate** in the Pattern Explorer, the XPath expression is evaluated and the transform occurs.

The transform can evaluate the value of other parameters by using the getValue() XPath expression. An XPath expression builder is included on this tab to help you construct the appropriate XPath expressions and to evaluate them for testing.

If you are using tables as part of the parameter configuration, you use Java code instead of XPath expressions to evaluate the table values.

You use the **Enable** tab to create an XPath expression that controls whether the parameter is visible to the pattern user in the Pattern Instance editor.

Slide 20



Setting the Patterns Explorer category

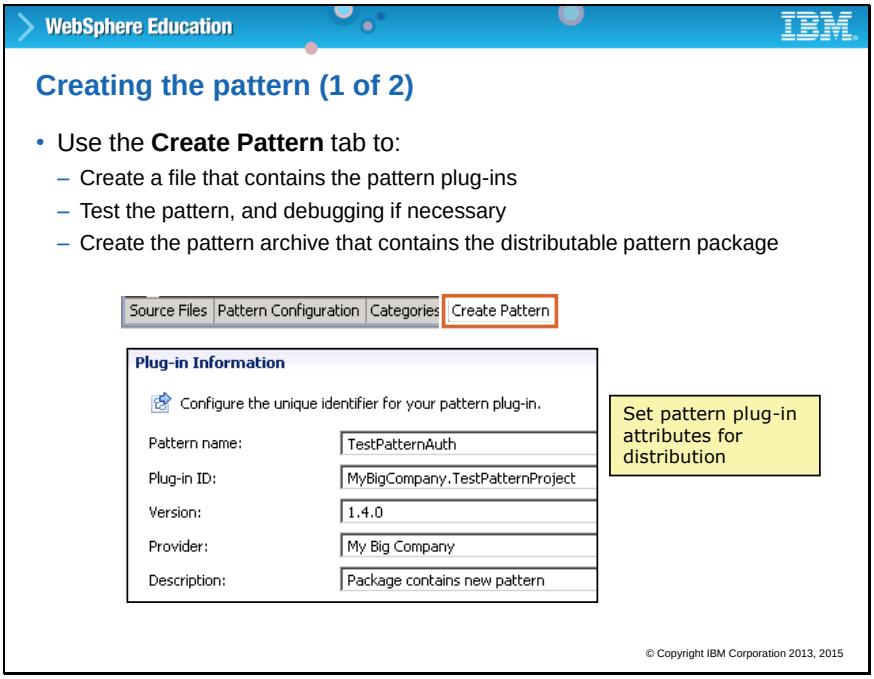
Before you generate the pattern plug-ins, you must define where the new pattern is displayed in the Patterns Explorer in the Integration Toolkit.

First, select the **Categories** tab, and then drag the new pattern name to the location in the pattern hierarchy that you want.

You can also create a category to contain the pattern by clicking **Add Category**. You can add the category at any location in the hierarchy.

Normally you set the pattern location after you customize all the parameters and parameter groups, but you can do it any time before you generate the pattern plug-ins.

Slide 21



Creating the pattern (1 of 2)

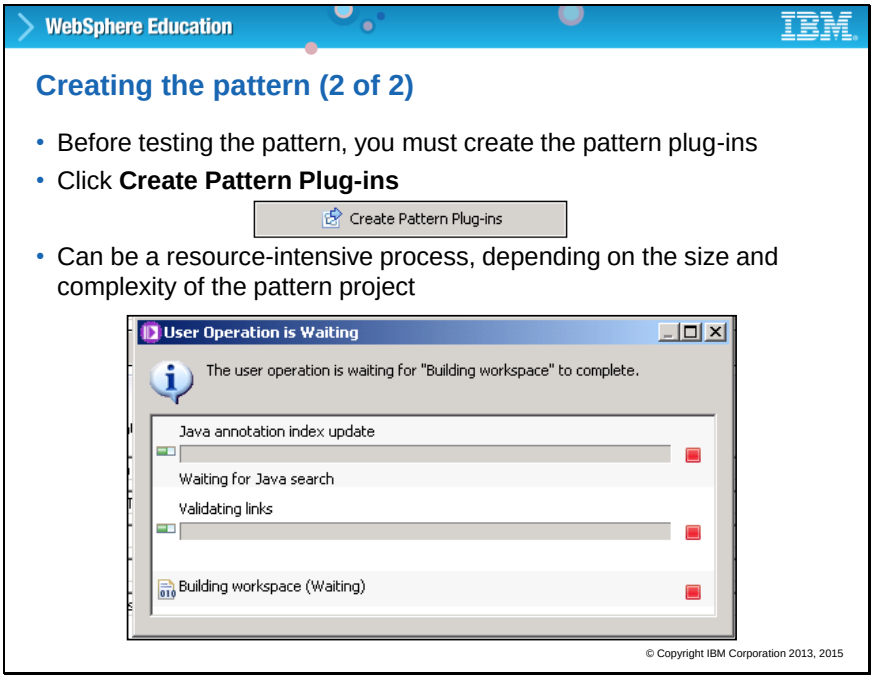
You do the final steps of creating a user-defined pattern in the **Create Pattern** tab. These steps include:

- Setting the attributes for the pattern plug-ins file
- Generating a file that contains the pattern plug-ins
- Testing and debugging the pattern
- Creating the pattern archive file. This file contains the user-defined pattern in distributable form.

As you saw with the other aspects of pattern development, the pattern authoring editor guides you through these last steps by presenting a window that contains controls for you to use.

On the **Create Pattern** tab, you set attributes for the plug-ins file. These attributes are packaged with the plug-ins themselves. The information that you provide on this page typically follows your development standards for nomenclature, standards, and version numbers.

Slide 22



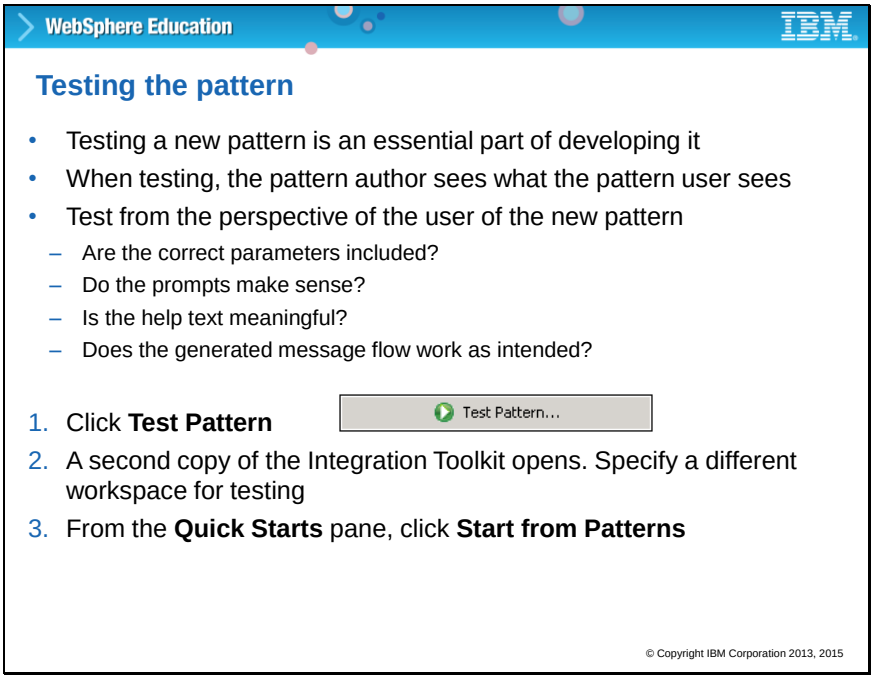
Creating the pattern (2 of 2)

Before you can test a pattern, you must generate a file that contains the pattern plug-ins by clicking **Create Pattern Plug-Ins** on the **Create Pattern** tab.

Depending on the size and the complexity of the pattern project, this operation can take several minutes and use much of the workstation processor power.

The file that contains the plug-ins is made available for testing.

Slide 23



Testing the pattern

A critical aspect of creating a user-defined pattern is testing it. When you test a pattern, a second copy of the Integration Toolkit opens. It includes the pattern plug-in that you created in the previous step.

During testing, the author should do the same as a user from selecting the pattern from the Patterns Explorer, through customizing the pattern parameters, to generating the pattern instance. This testing allows the author to determine whether any part of using the process must be modified.

Testing the pattern creates a workspace and shows the Patterns Explorer that includes the new pattern. You cannot use the same workspace that is open for the Pattern authoring editor in the original instance of the Toolkit.

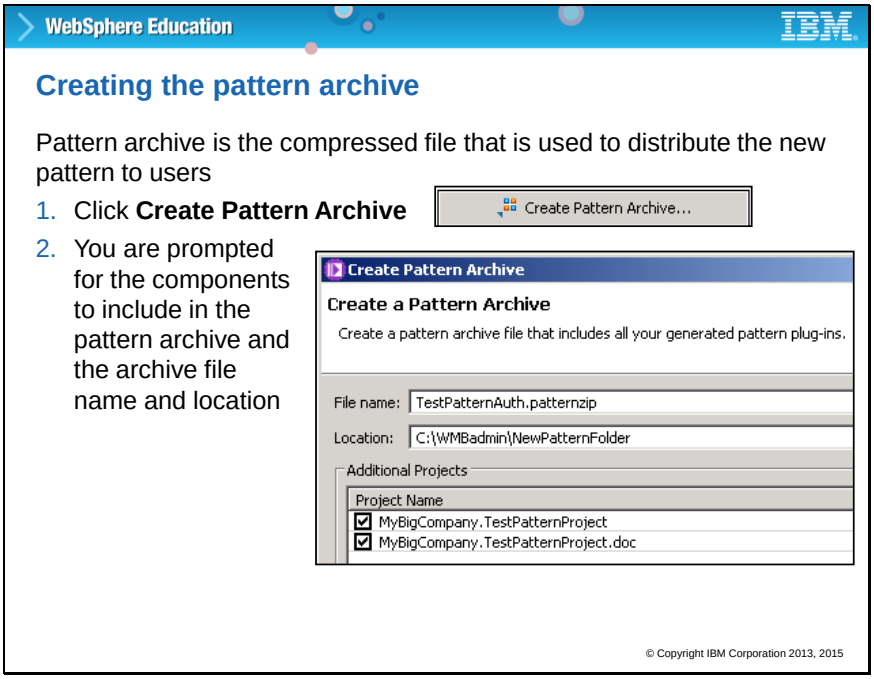
Open the Patterns Explorer from the **Quick Start** pane by clicking **Start from Patterns**. The new pattern is displayed in the Patterns Explorer in the location you set in the previous steps. Select the pattern and begin testing.

As you test, consider yourself to be the pattern user. Some considerations for testing include:
- Are the correct parameters available for the user to configure?
- Do the configuration steps make sense, and are they presented in a logical progression?
- Are the correct editors used?

- Are the default values appropriate?
- If parameters and parameter groups were set to appear conditionally, do they appear (or not appear) as expected?
- Is the documentation for the pattern and the parameters correct?
- Does the generated message flow work as intended?

When you are finished testing the pattern plug-in, close the second copy of the Integration Toolkit.

Slide 24



Creating the pattern archive

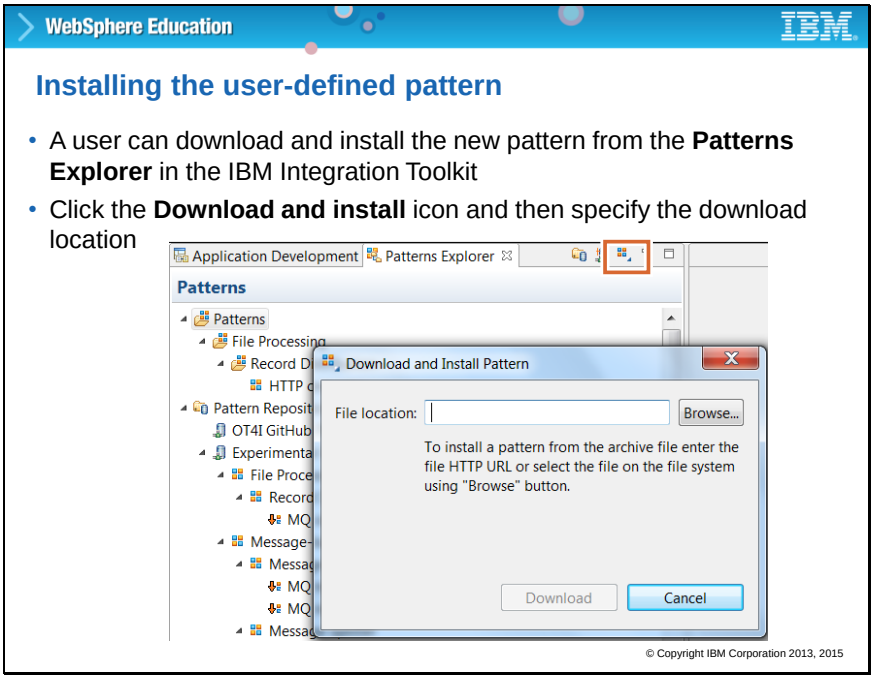
After you complete all development and testing for the new pattern, you can package it for distribution by creating a pattern archive. A pattern archive is a compressed file that contains all the artifacts that are needed to deploy the pattern to other Integration Toolkit installations.

To create the pattern archive, click **Create Pattern Archive** on the **Create Pattern** tab. You are prompted for the name and location for the archive file, and the components to include in it.

Like the generation process for the pattern plug-ins, creating the pattern archive can be resource-intensive and can take an extended amount of time to complete.

After the file is generated, you can distribute it according to the policies of your organization.

Slide 25

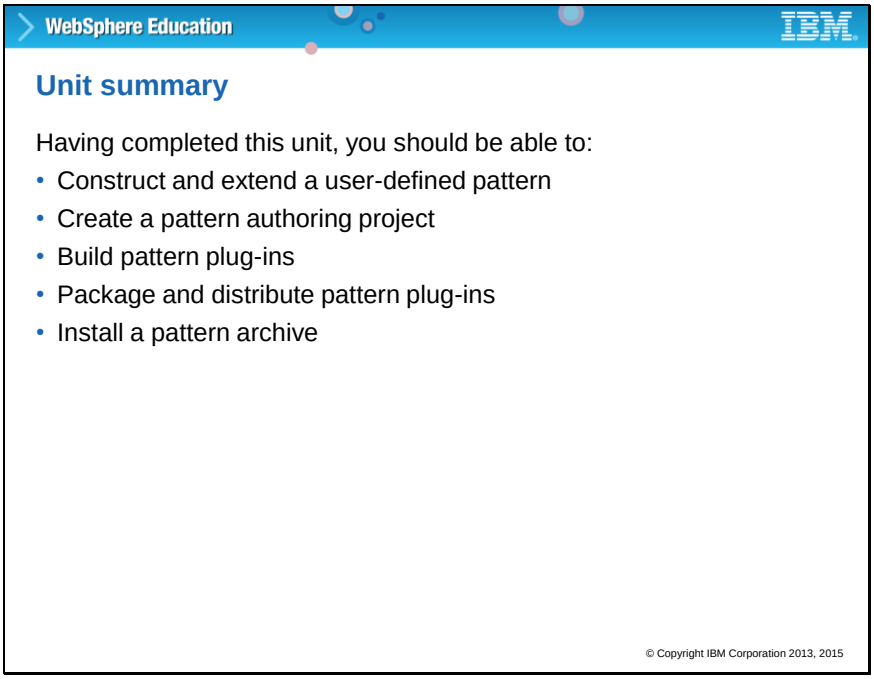


Installing the user-defined pattern

To install the user-defined pattern in the Integration Toolkit, you use the Patterns Explorer.

Click **Download**, and then select the location of the pattern archive (.patternzip) file. When the file is located, the Integration Toolkit automatically extracts the components and adds the pattern to the Patterns Explorer.

Slide 26



Unit summary

This unit described how to create and test user-defined message flow patterns in the IBM Integration Toolkit.

Having completed this unit, you should be able to:
- Construct and extend a user-defined pattern
- Create a pattern authoring project
- Build pattern plug-ins
- Package and distribute pattern plug-ins
- Install a pattern archive