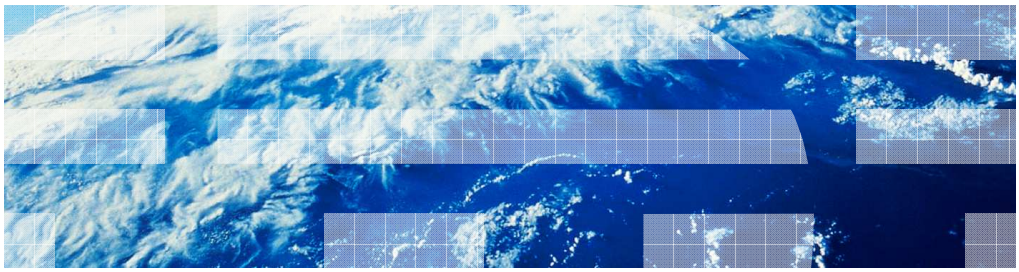
WebSphere Business Process Management

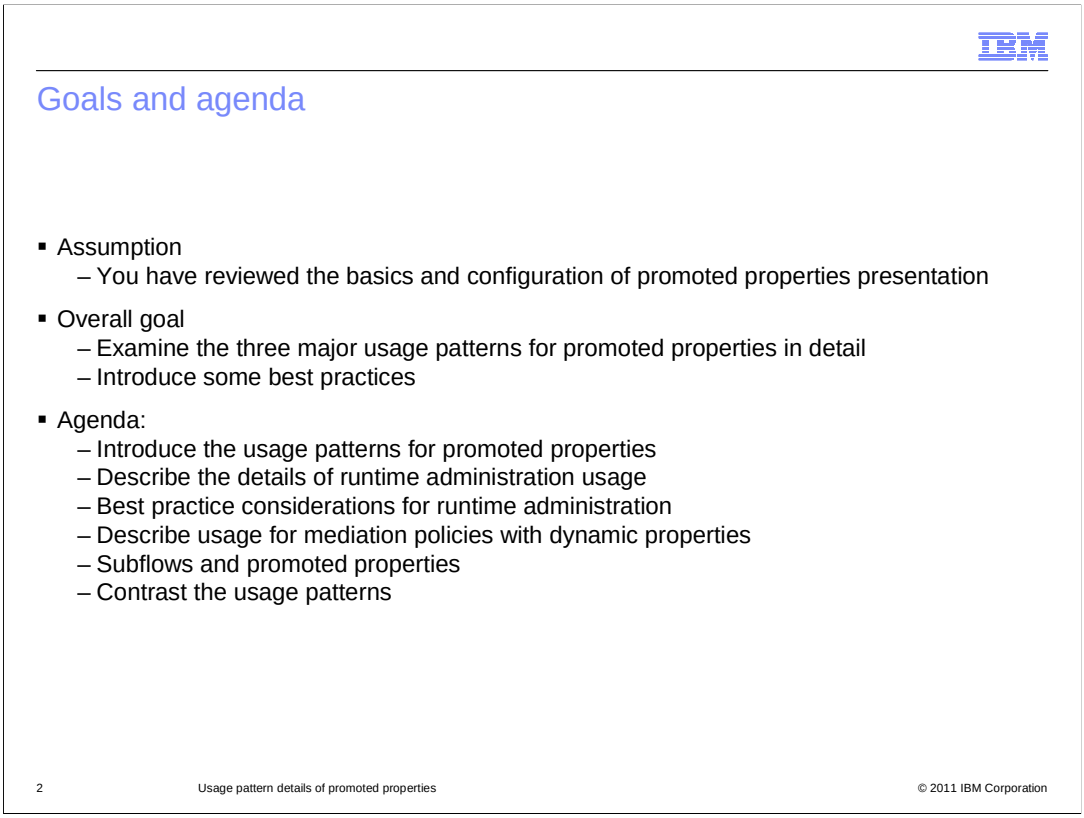
WebSphere Integration Developer
WebSphere Enterprise Service Bus
WebSphere Process Server

Usage pattern details of promoted properties



WebSphere. software

This presentation looks at the three major usage patterns for promoted properties. It is the second of two presentations. The first one dealt mostly with the definition and configuration of promoted properties, while this one looks at the details of the usage patterns themselves.



The content of this presentation assumes that you have reviewed the presentation on the basic concepts and configuration of promoted properties, as it provides a foundation for understanding the content presented here.

The overall goal is to view promoted properties in terms of the three major usage patterns, looking at the details of each. The runtime behavior is described and some best practices for runtime administration are introduced.

The presentation starts with an overview of the three usage patterns, contrasting the key factors of how each is different from the other. The runtime administration usage is then described in detail, followed by a set of best practices when using promoted properties for administration. The next section looks at the use of promoted properties for enabling the application of mediation policy. The third usage pattern looked at is how promoted properties are used to configure invocations of subflows.

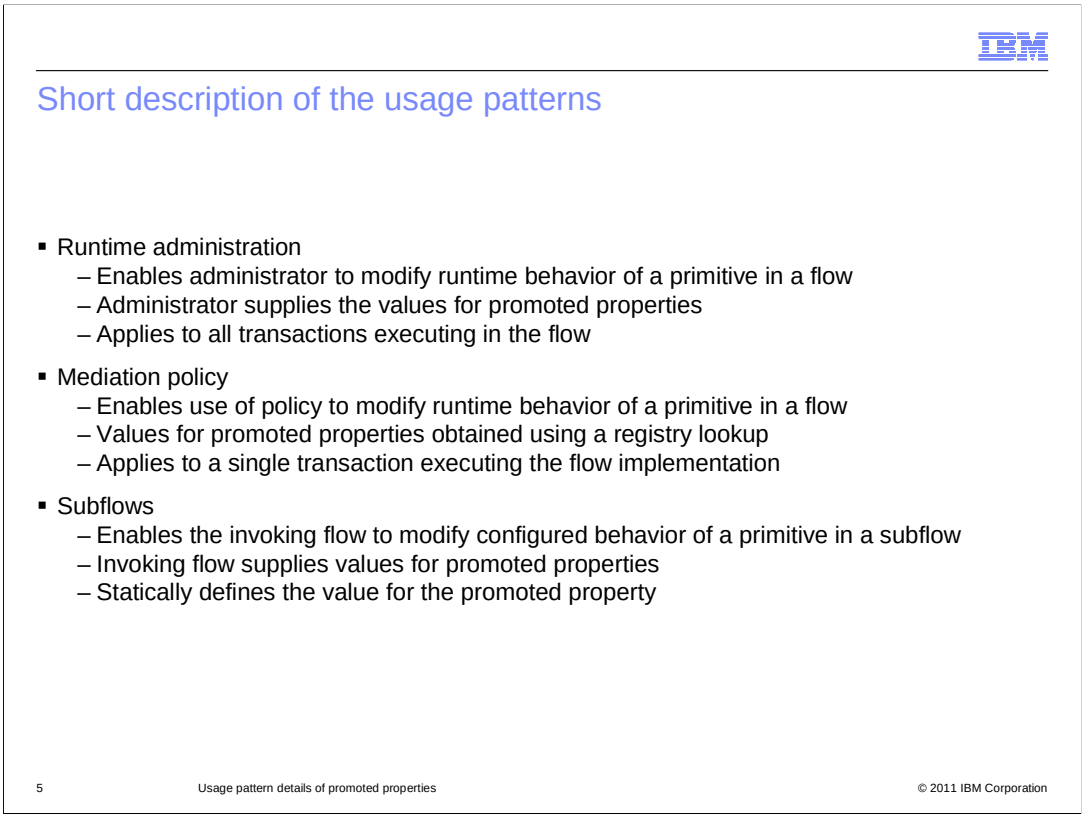
The presentation concludes with a chart that concisely details the contrasting aspects of each of these patterns.

This section provides a short introduction to the three usage patterns and highlights how they are similar and in what ways they are different.

Introduction to the usage patterns

- There are three major usage patterns for promoted properties
  - Runtime administration
  - Mediation policy
  - Subflows
- Concept of a promoted property is the same for a each pattern
  - Steps used to promote a property in WebSphere Integration Developer
  - A promoted property allows statically defined property value to be overridden
- Usage is different for each pattern
  - How the overriding value is provided
  - The scope over which the override is in effect

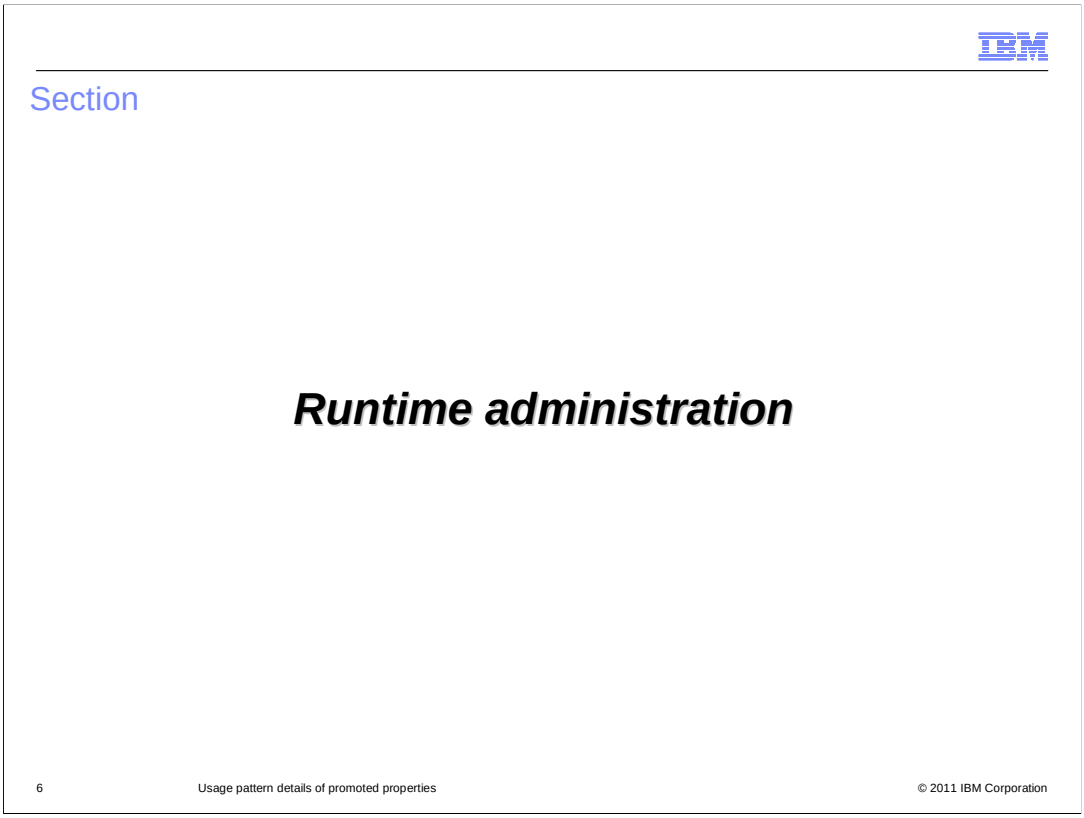
The three usage patterns for promoted properties are runtime administration, application of mediation policy and configuration of subflows. As was covered in the previous presentation, the mechanism provided by WebSphere Integration Developer for promoting a property is the same for all three patterns. The promotion of a property enables the statically defined value of a property to be overridden. That encompasses the similarity between the three patterns. They are different in how the overriding value is specified and the scope over which the overriding value has effect. These differences are covered in the next slide.



Runtime administration enables an administrator to modify the runtime behavior of a primitive that is part of a flow contained in an installed mediation application. The administrator provides the overriding value during application installation or as an administrative update to an installed application. The overriding value applies to all transactions executing in that flow.

Mediation policy enables the use of policy specifications to modify the runtime behavior of a primitive in a flow. The overriding value for the promoted property is obtained using a registry lookup. The lookup accesses policy information based on data values in a specific execution instance of the flow. The overriding values only apply to the one execution instance of the flow.

Subflows enable flow logic to be reused from many different flows. Promoted properties enable the invoking flow to configure the behavior of the subflow. The invoking flow provides the overriding value for the promoted properties. In essence, the subflow can be thought of as a primitive type, and the properties of the subflow are the promoted properties of the primitives that make up the subflow logic. Therefore, the statically defined value associated with a property of a primitive in a subflow is statically overridden for its use within a parent flow.

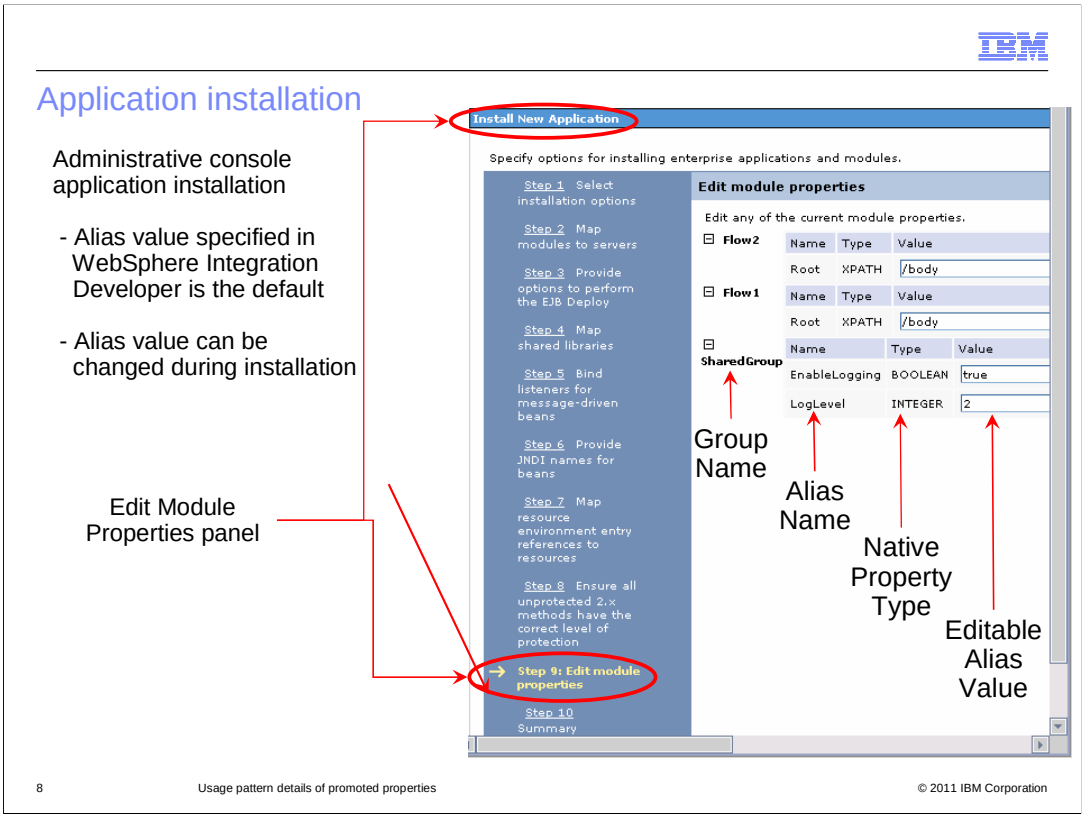


This section addresses the runtime administration usage pattern. It shows you how promoted properties are administered at runtime, looking at both installation updates and administrative runtime updates.

Runtime administration

- Promoted property values administered at runtime through:
  - Application installation
    - Administrative console
    - Command line
  - Updates to installed application
    - Administrative console
    - Command line

Promoted property administration at runtime provides the overriding property values either during the application installation process or as an administrative update to an installed application. Either of these can be done using the administrative console or the wsadmin command line. The next four slides looks at each of these variations.



This is a screen capture of the administrative console when doing a new application installation for an application containing a module with a mediation flow component. The left side contains a list of panels which are relevant to the installation of an application. Shown in the screen capture is the Edit module properties panel, which provides the ability to modify the alias values for the promoted properties. You can see in the screen capture that each alias is listed by group name and alias name, along with its native property type. The alias values are presented in an editable form. They are initialized to the values that were specified for them in WebSphere Integration Developer. The solution administrator has the opportunity to specify new alias values at this time.

Application installation

- Command line application installation
  - Normal use of: wsadmin $AdminApp install
  - Use SIBSCAClientInstall to specify alias override values

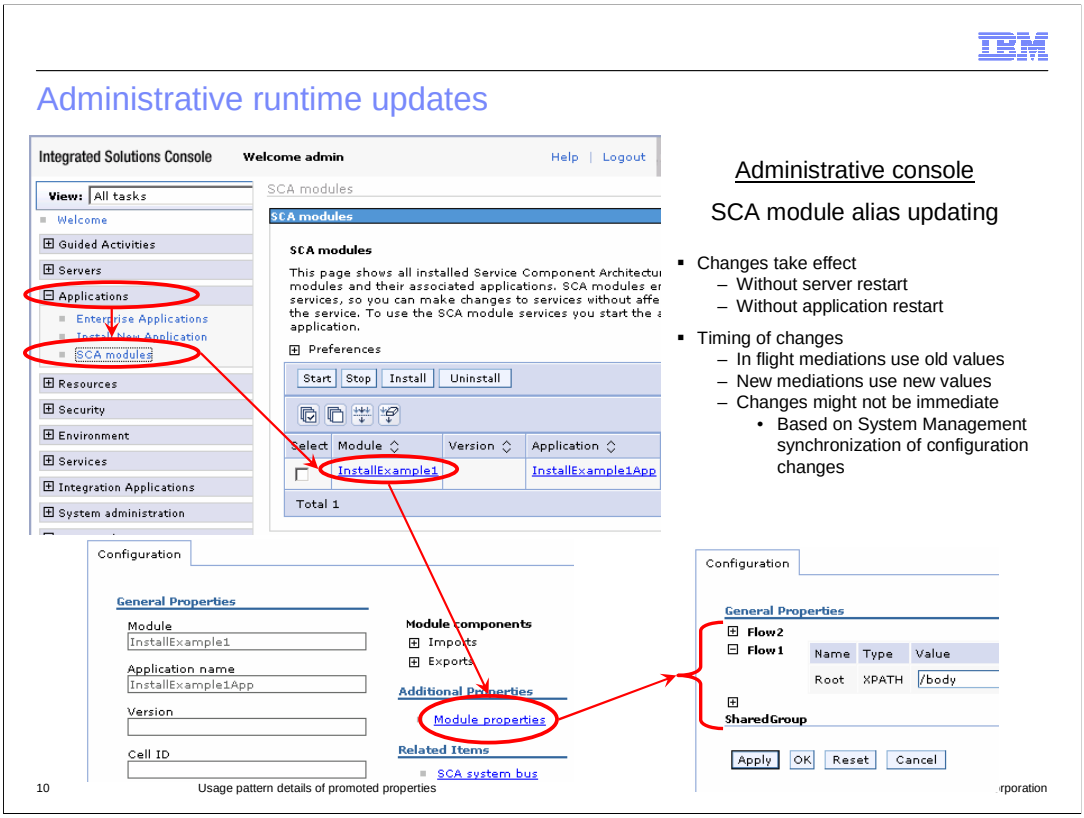
```
{ -SIBSCAClientInstall {{,[groupName]<aliasName>=<aliasValue>}}}
```

    - Aliases not specified default to value specified in WebSphere Integration Developer

  - Example:

```
wsadmin>$AdminApp install InstallExample1.ear
{-SIBSCAClientInstall
  {
     { [SharedGroup]LogLevel=3,
       [Flow1]Root=/context }
   }
}
```


This slide also describes application installation, but for command line usage rather than through the administrative console. It makes use of the wsadmin $AdminApp install command with the SIBSCAClientInstall option. The syntax for the command and an example are shown in the slide. Any aliases not specified using this option default to the value given to the alias in the WebSphere Integration Developer.



Screen captures from the administrative console are shown here. They illustrate how to display and update aliases for an SCA module in an installed application.

To navigate to the aliases, you open Applications and then select SCA Modules. This puts you in a panel listing all of the SCA modules within the scope of the administrative console. Selecting a particular module puts you into the configuration panel for that module, on which you then select Module Properties.

You are now shown the configuration panel listing all of the groups. Each group can be expanded to show the aliases, their native property types and editable values. If you edit a value, you need to apply the change and then save the configuration. This is the same as when you make any other change using the administrative console.

Once the configuration is saved, the new alias value applies to the mediation module. There is no server or application restart required. Any in flight instances of a mediation flow continue to use the old value, but new instances that are started make use of the new value. As with any server configuration change, if this is a Network Deployment environment, there might be delays between when the configuration is saved and when the changes take affect on individual servers within the cell.

Administrative runtime updates

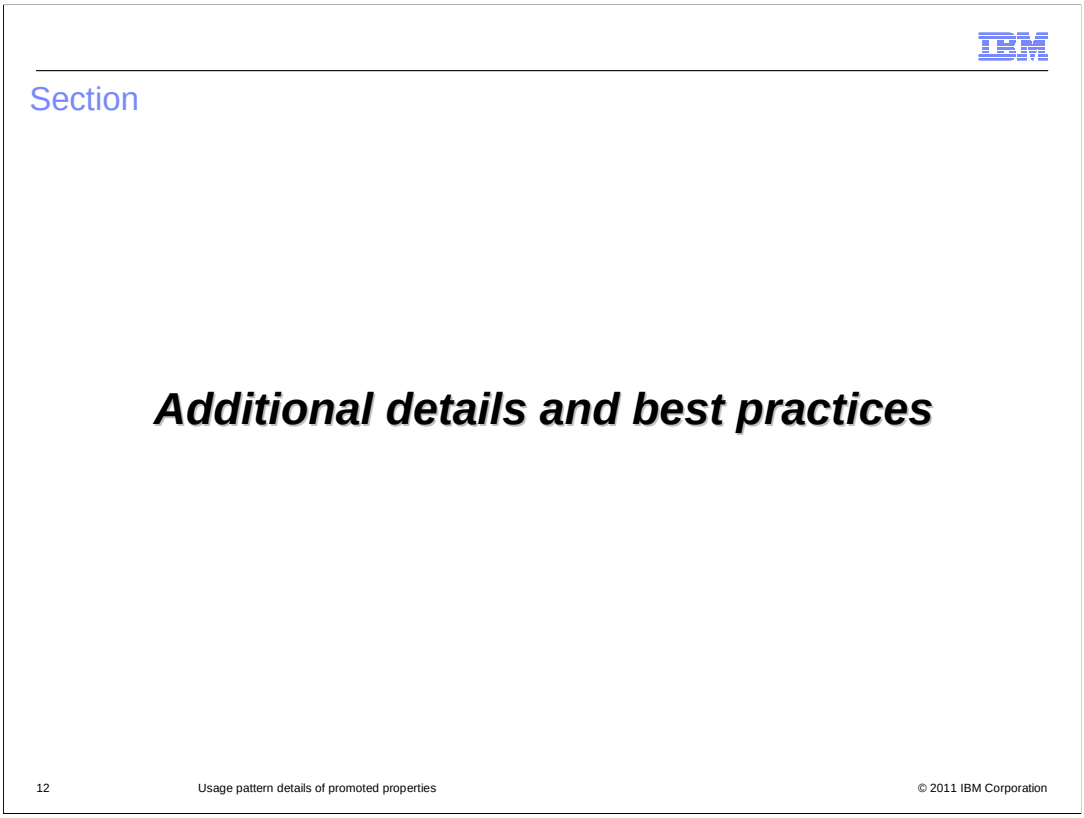
- Command line updates use the wsadmin $AdminTask modifySCAModuleProperty command

```
$AdminTask modifySCAModuleProperty {
      -moduleName myModule
      -applicationName myApplication
      -propertyName myPropertyName
      -newPropertyValue myNewPropertyValue
}
```

- Same behavior as described for updating using the administrative console
  - When configuration is saved, the changes take effect without a server restart or application restart
  - In flight mediations use old values
  - New mediations use new values
  - Changes might not be immediate due to timing of synchronization of configuration changes in a network deployment environment

This slide also describes administrative runtime updates to an installed application, but for command line usage rather than through the administrative console. It makes use of the wsadmin $AdminTask command modifySCAModuleProperty. An example of this command is shown in the slide.

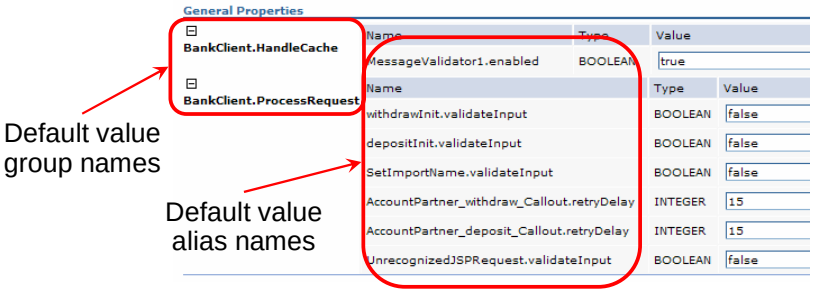
The behaviors described on the previous slide also apply. The configuration must be saved in order for the change to take affect, and then the change occurs without requiring an server or application restart. The in flight instances of a mediation flow continue to use the old value, whereas any new instances make use of the new value. As with any server configuration change, if this is a Network Deployment environment, there might be delays between when the configuration is saved and when the changes take affect on individual servers within the cell.



To fully understand promoted properties, there are some additional details that you need to be aware of. This section looks at some of those details and addresses some best practices for the use of promoted properties.

Default alias naming

- Alias names are qualified by group names
  - Default group name
    - Default group name = <module_name>.<mediation_flow_component_name>
  - Default alias name
    - <mediation_primitive_name>.<property_name>



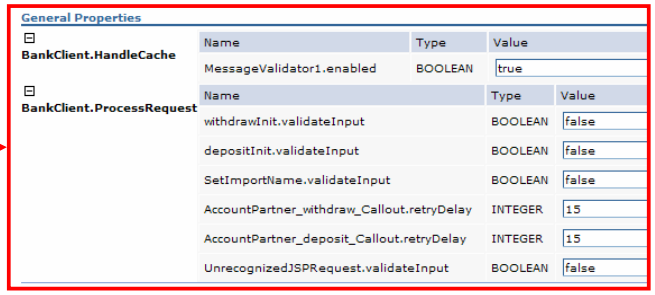
  - You can (should) change the default group and alias names to have more meaningful values

When you mark a property as being promoted, WebSphere Integration Developer assigns a default group name and default alias name. The default group name is the name of the mediation flow component, qualified by the name of the module. The default alias name is the property name, qualified by the mediation primitive name. You can, however, change the default names to provide your own names. This is generally a good idea. Although these default values might have meaning to you, the integration developer, they are not likely to be meaningful to an administrator.

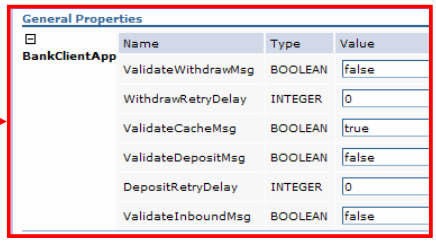
Example

- Example contrasting what administrator sees with default naming versus meaningful names

Default values used



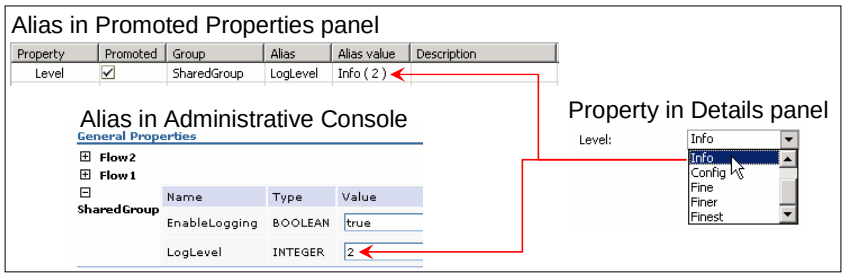
Meaningful values used



14 Usage pattern details of promoted properties

This slide provides an example of what the administrator sees when modifying promoted property values. In the upper screen capture, the default values have been used. In the lower screen capture meaningful group and alias names are used. You can see that, from the administrator's viewpoint, these names are clearer in their meaning.

Enumerated values

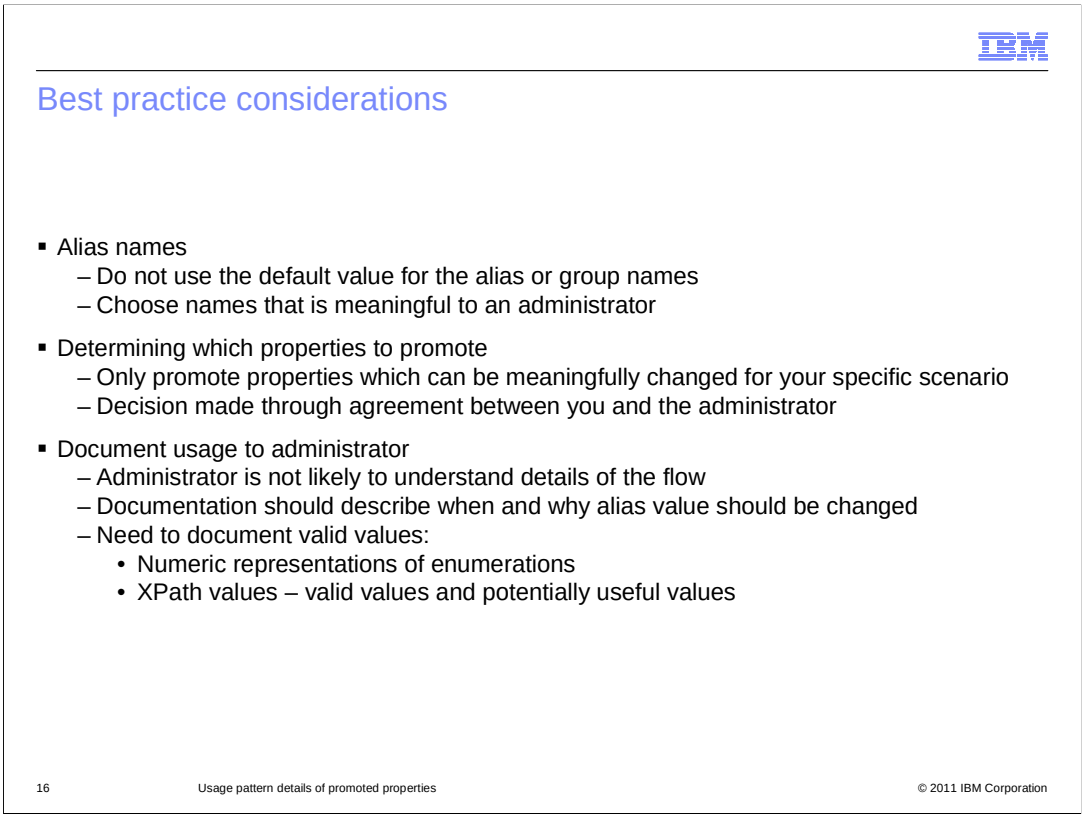


- Representation of enumerated values
  - WebSphere Integration Developer displays meaningful values for properties with enumerated values
  - However, at runtime alias values are String representations of integers
    - Promoted properties panel shows both the meaningful value and the integer value
  - WebSphere Integration Developer ensures value set is a valid choice
    - In Details panel
    - In Promoted Properties panel
- High potential for error by runtime administrator using administrative console
  - No way to check for correct value
  - Use of integer rather than meaningful value

15 Usage pattern details of promoted properties

One of the things to be aware of with promoted properties is the representation of enumerated values for properties. In WebSphere Integration Developer, if some property has a defined set of values, they are displayed to you with meaningful names. However, in the artifacts that are generated for the runtime to use, the meaningful name is normally replaced with some integer value expected by the runtime. When one of these integer values is used as an alias value, it is in fact treated as a String representation of the integer value. Because of this, the solution administrator is not presented with alias values that are meaningful to a human. Looking at the example, you can see on the right that the level property, from a message logger, can be set to one of several different values. For example, you can select info, config, fine, finer or finest. For the runtime, this choice is represented by an integer. In the example, the choice info is represented by the integer 2. When this property is promoted, WebSphere Integration Developer displays both the meaningful and integer value in the promotable properties panel so that you can know what integer is assigned to that choice. Finally, in the administrative console, the solution administrator is only presented with the string representation of the integer value.

This overall situation presents a few problems. First, the solution administrator must be made aware of what the integer values actually mean. A second problem has to do with specification of an incorrect value. In WebSphere Integration Developer, you cannot set a value for an enumerated property that is not valid. However, at runtime, the only checking done when the alias value is updated is to insure it is a valid string. If the solution administrator provides a value that is not an integer meaningful for the property, a runtime error occurs for all subsequent mediation flows using that alias.



Here are some best practices that can help you to make the best use of promoted properties when used for administration.

First, it is always best to use alias and group names that are meaningful to the solution administrator. They do not have access to the mediation module in WebSphere Integration Developer and are probably not aware of the names of the mediation primitives and properties which have been promoted. They need to have something at makes sense to them in the context of the module they are administering.

You should only promote properties which result in some meaningful usage for your mediation module. This decision most likely should be made through a dialog between yourself, as the integration developer, and the solution administrator. If the use of the promoted property does not make sense to the solution administrator, it is probably best not to promote it.

Finally, all promoted properties should be well documented for the solution administrator, as they are not likely to understand the details of the flow. The documentation should cover every alias name, describing when and why the alias value might be changed. It is very important to document what the valid values are. This is particularly true for enumerated types which do not have meaningful representations in the administrative console. It is also true of properties that contain XPath expressions. Which XPath expressions are potentially valid should be documented, along with the conditions under which each might be selected.

Aliases and problem determination

- Runtime problems can be caused by:
  - Specification of an incorrect alias value
  - An unintended shared alias
- Examine alias values in administrative console
  - Check for an alias with an incorrect value
    - Enumerated value set incorrectly
    - XPath expression incorrect
- Check alias names in WebSphere Integration Developer
  - Check for inadvertent use of a common group and alias name
    - Results in a shared alias when that was not your intention
      - Remember, scope of an alias name is the entire module, not just a flow
  - Check for unshared properties which should have been shared
    - Property was supposed to be promoted but was not
    - Property was given an incorrect group or alias name

As with any functionality that provides flexibility and dynamic capabilities, there is the potential for problems to occur. There can be runtime problems caused by aliases which have been given incorrect values. In addition, an unintended or incorrectly configured usage of a shared alias can also lead to unexpected results. This slide examines some of the things you can do when faced with a problem you suspect to be caused by use of an alias.

The first place to start with a runtime error is in the administrative console, checking the aliases listed in the module properties configuration panel. Make sure that all of the aliases have the right values. Those for enumerated types or XPath expressions are probably the ones most prone to error. If everything seems to be OK, the next place to look is at the mediation flow components in the WebSphere Integration Developer.

One problem might be the inadvertent use of a common group and alias name, thus establishing a sharing relationship that you did not intend. Remember that the scope of sharing is the entire module. Use the promoted Properties panel associated with the Overview panel of the mediation flow editor to check for this possibility, and cross check them if you have multiple mediation flow components.

Another problem to check for is when a property is supposed to be shared but does not appear to be using the shared alias value. It can be caused if the property was not promoted when it should have been and can also occur if the alias name was not specified correctly.

Section

Mediation policies and dynamic properties

This section looks at mediation policies. The use of promoted properties for mediation policies is enabled by the runtime capability of dynamic properties. How dynamic properties work in the runtime is explained.

What is mediation policy?

- A mediation policy is used to dynamically control a mediation flow
- Can be used to control any aspect of a mediation
  - Message transformation
  - Message content
  - Message routing
- A mediation policy can be associated with:
  - A mediation flow implementation
  - A target endpoint
- The mediation policy definition is external to the mediation flow implementation
- A mediation policy is applied on a per message basis
  - Message content is used to select the policy applied to each message flow

A mediation policy is used to dynamically control a mediation flow. They can be used to affect any aspect of the mediation flow, such as message transformation, message content or message routing. Mediation policies can be associated with a particular mediation flow implementation, or can be associated with a target endpoint. The mediation policy definitions are stored in a registry external from the mediation flow implementation and are then looked up as part of the mediation flow. The application of a mediation policy is done on a per message basis, with the selection of policy based on message content.

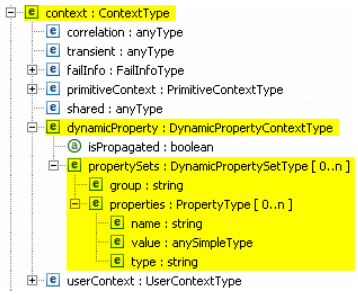
Basic steps for application of mediation policy

- Mediation policies are stored in WebSphere Service Registry and Repository
- Mediation flows lookup policies using the policy resolution primitive
- Policies returned are based on evaluation of conditional expressions using values taken from the message
- Returned policies are stored in the dynamicProperty context in the SMO
- Subsequent primitives in the flow get the value for their promoted property from the dynamicProperty context of the SMO

The essential steps associated with application of mediation policy are explained here. First of all, the policies are loaded into and stored in the WebSphere Service Registry and Repository. Within a mediation flow, the policy resolution primitive is used to perform a lookup of policies from the registry. The policies that are returned from the registry are selected based on gate conditions, which are essentially conditional expressions evaluated using values taken from the message. The policies that are returned contain property values that are written to the dynamicProperty section of the SMO context. Subsequent mediation primitives in the flow resolve their promoted property aliases by taking the values from the dynamicProperty section of the SMO context. These values override the alias values associated with the installed application. Because the property values are in the SMO, the scope of dynamic properties affects individual instances of a flow.

Dynamic properties context in the SMO

- SMO dynamic property context contains:
  - dynamicProperty: DynamicPropertyContextType
    - Contains sequence of propertySets
  - propertySet: DynamicPropertySetType [0..n]
    - Has a group name
    - Contains a sequence of properties
  - properties: PropertyType [0..n]
    - Has a name, value, type triplet
- Equivalent to module properties
  - Groups of properties
  - Each property has a name, value and type
- Resolution of promoted property values at runtime
  - If group/property found in dynamic properties context, use the value
  - Obtain value from the module properties



The screen capture on the right shows the context section of the SMO with the dynamic property section expanded. The section is defined by the DynamicPropertyContextType, which essentially contains a sequence of property sets. Each property set, defined by the DynamicPropertySetType, has a group name and a sequence of properties that are part of that group. Each property is defined by PropertyType and contains a property name, its type and its value. If you recall the module properties previously shown in the administrative console, they were organized into groups of properties with each property having a name, type and value. Therefore, the dynamic property context in the SMO has equivalent information to the module properties maintained by the runtime application configuration.

So how do dynamic properties work at runtime? When a primitive's promoted property in a mediation flow needs to be resolved to a value, the dynamic properties context in the SMO is checked first. If the corresponding group and property name is found in the SMO, the value contained in the SMO is used. If it is not present, than the value for the corresponding group and property name is taken from the module properties.

Contrasting module and dynamic properties

Module Properties	Dynamic Properties
Contained in the server's application configuration	Contained in the SMO for a flow instance
Property value set in this order (last one wins) 1) Default (in application EAR) 2) Specified at application install 3) Set in installed application by administrator	-Value set by Policy Resolution mediation primitive (intended use) -Value set by mediation flow logic using any primitive type (works, but is not the intended use)
Are always present	Are only present if explicitly set
Applies to all instances of the flow	Applies to a single instance of the flow
Enables administrative control of the flow	Enables contextual control of the flow

One of the best ways to understand the overall behavior of promoted properties is to compare and contrast module properties with dynamic properties.

First of all is the location where they are stored. The module properties are part of the server configuration being kept as part of the installed application's configuration. However, the dynamic properties are part of the SMO associated with an individual flow running in the server.

Secondly is how the property values are set. Values for module properties can come from one of three sources. The application EAR file contains default values for the promoted properties. These default values can be overridden when the application is installed. After the application is installed, the administrator can modify the property values in the installed application. The dynamic properties are set during a mediation flow. The intended and documented usage is for the policy resolution mediation primitive to lookup policies from the registry and set the values into the SMO. However, it is also possible to explicitly set the dynamic properties section of the SMO using any other mediation primitive types in the flow. This works, although it is not the advertised usage.

The next thing to note is that the module properties are always there, containing values for all of the promoted properties associated with the module. The dynamic properties only contain groups and properties that have been explicitly set.

One key differentiation is that the module properties apply to all instances of a flow. Even when an administrator updates a property value, the old value applies to all instances of the flow started before the change, and the new value applies to all instances started after the change. Dynamic properties, however, apply to individual instances of a flow. Which properties have values depend upon the policies applied, and the policy lookups depend upon conditions which are based on values from the individual messages.

So, essentially the module properties are there to enable administrative control over mediation flows, whereas the dynamic properties enable contextual control of the flow.

Yet another use of promoted properties is to allow for the configuration of subflows. This section takes a brief look at how this is done.

Subflows overview

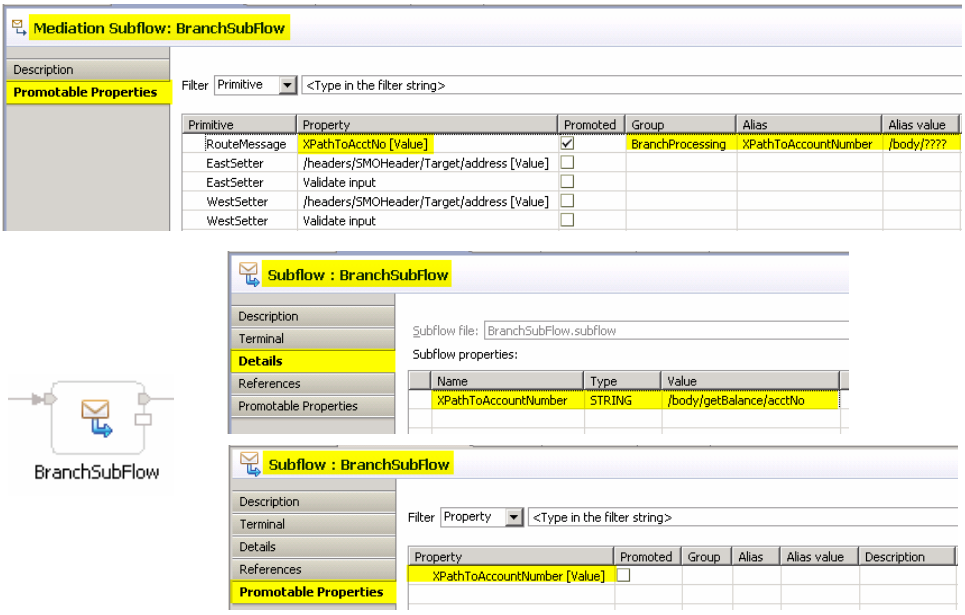
- Mediation subflows enable reuse of mediation logic
- A subflow is very similar to a mediation primitive
  - Performs some defined function
  - Has properties that can be set to configure its specific behavior
  - Is used by wiring it into a mediation flow
- A subflow is composed of:
  - Mediation primitives wired together similar to a flow
  - In and Out - represent the input and output terminals of the subflow
- Properties for primitives in a subflow
  - Subflow designer promotes selected properties
  - Promoted properties become the subflow's properties, similar to a primitive's properties
  - The promoted property values are set by the invoker of the subflow

Mediation subflows are a way to enable reuse of mediation logic across multiple flows and applications. From the outside, a subflow is very much like a mediation primitive. It has some defined function that it performs and has properties that can be used to configure its behavior. The subflow can be dropped onto the canvas of the mediation flow editor and wired into the flow, the same as any other mediation primitive can be.

From the inside, the subflow is very much like a flow. You drop mediation primitives onto the canvas of the editor, set their property values and wire them together. The subflow has an In and an Out, which are like the input node and callout node of a request flow. These represent the input and output terminals of the subflow when looking at it from the outside.

When designing a subflow, you determine which of the promotable properties should be configurable by an invoker of the subflow. You then promote those properties. When you use the subflow in a parent flow, these promoted properties are exposed to you as the properties for which you need to provide values. This is very similar to configuring any other mediation primitive.

Subflow promoted property usage



Usage pattern details of promoted properties

The screen captures on this slide are intended to illustrate the use of promoted properties with subflows. The screen capture at the top of the slide is the Promotable Properties panel for a mediation subflow named BranchSubFlow. As you can see, there are several promotable properties associated with the subflow, but only one, XPathToAcctNo is promoted. It is given a slightly more meaningful alias name, XPathToAccountNumber. Apparently, this subflow does some kind of processing based on an account number, but when invoked from different flows, the account number is located in different places in the SMO. By promoting this property, the invoker can tell the subflow where in the SMO the account number is found.

Moving down on the slide, the lower left shows that an invocation of the subflow looks just like any other mediation primitive wired into a flow. To its right are the Details and Promotable Properties panel associated with the invocation of the subflow. You can see in the Details panel that the property XPathToAccountNumber is displayed and that a value has been configured for it. All the properties promoted by the subflow appear in this table, allowing the invoker to set the property values.

On the Promotable Properties panel, notice also that the XPathToAccountNumber property in the invoker is promotable, but in this case it is not promoted.

From this example, you can see that a promoted property in a subflow is used to allow an invoker to configure a use of the subflow. The promoted property is not available for administrative control nor is it available for policy control. It is whether the invoker promotes it that determines if the property is made available for administrative and policy control.

Some final observations and a summary of this presentation follow.

Observations on promoted property usage

- Single mechanism for promoting properties
- Three distinct usages of promoted properties

Administrative control	Policy control	Subflow
Configurable application	Dynamic application	Static application
Installation and runtime administration of application	Control of individual invocations of the application	Development time configuration of application
Affects individual installations of the application	Affects individual usage of the application	Affects all installations of the application

- Beware of overlapping/conflicting usage

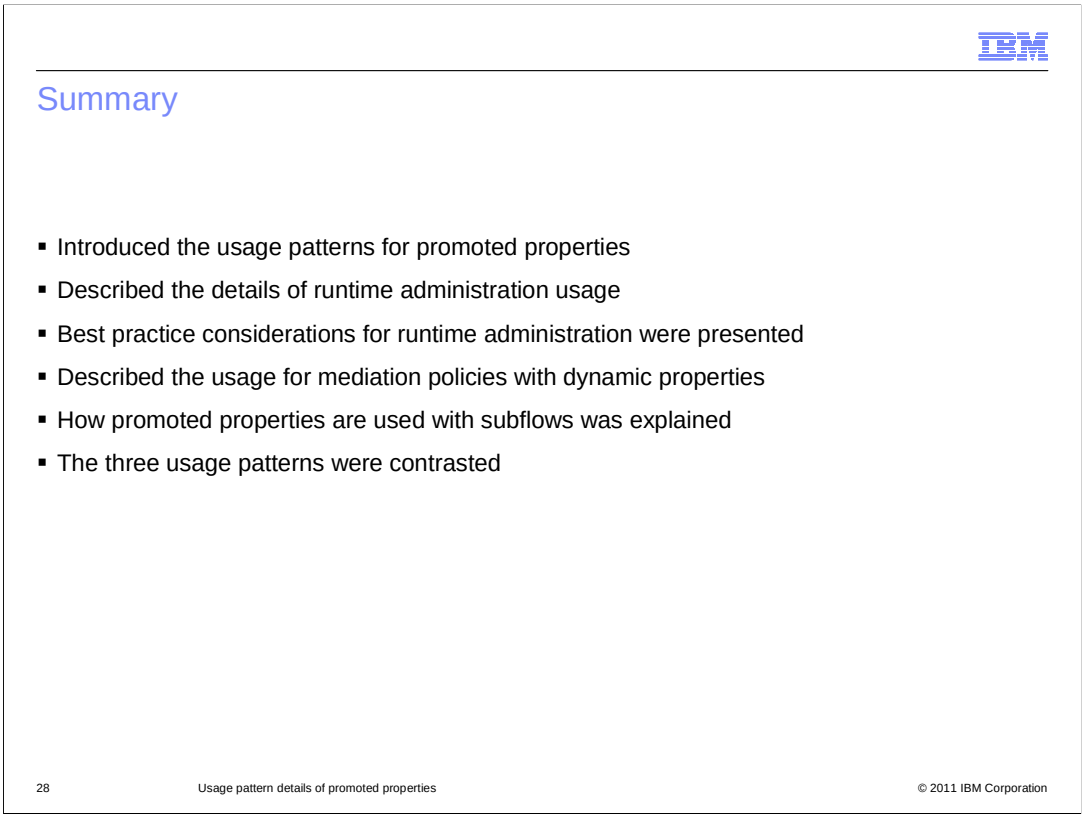
The overall use of promoted properties is somewhat overloaded. Although there is a single mechanism provided for promoting properties, there are three distinct uses for them. The different uses of promoted properties have varying characteristics. Understanding these helps to put the use of promoted properties into perspective.

First of all, the original purpose for promoted properties was to enable administrative control of mediation applications. This made the application configurable, allowing an administrator to configure its behavior at application installation time, and additionally to modify its behavior after installation. This allowed an application to be configured differently when installed into different server environments.

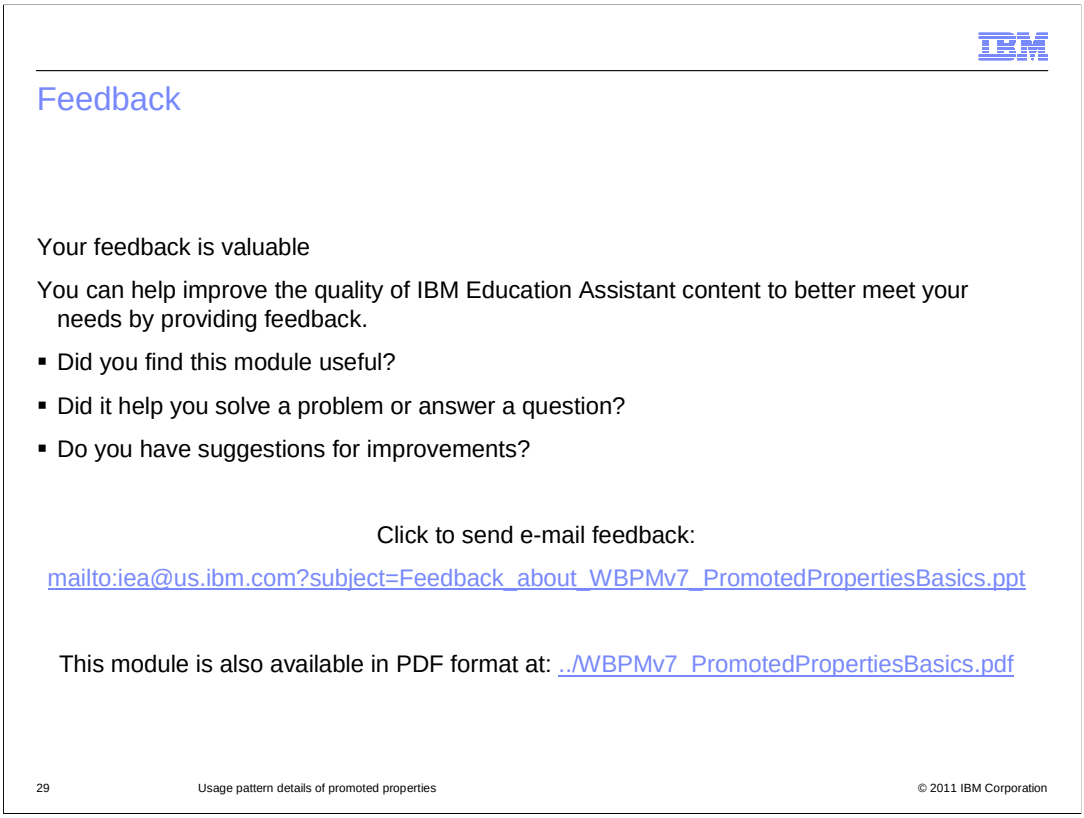
Policy control, enabled through the use of dynamic properties, allows an application to be dynamic. Modifications in behavior are controlled for every invocation of a flow based on contextual information in the message. This affects every single instance of using the application.

Subflows fall completely at the other end of the spectrum. They enable flexible reuse of mediation logic within an application and even across multiple applications. However, once the application is constructed, the use of the subflow and its promoted properties is static, being the same everywhere the application is installed.

This overloaded use of promoted properties should not be a problem if you are clear about your requirements and why you are promoting properties. Just as an example, suppose you have a property that is promoted expressly for policy based dynamic control. There are legitimately times when the property is not set based on policy, and you then want the development time default value to be used. Unfortunately, that property is exposed as a module property that can be modified by an administrator. Therefore, you need to make it clear to the administrators not to modify its value.



In summary, the presentation started with an overview of the three usage patterns, contrasting the key factors in how each is different from the other. The runtime administration usage is was described in detail, followed by a set of best practices when using promoted properties for administration. The next section looked at the use of promoted properties for enabling the application of mediation policy. How promoted properties are used to configure invocations of subflows was examined next. Finally, the presentation concluded with a chart that concisely detailed the contrasting aspects of each of the three usage patterns.



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information