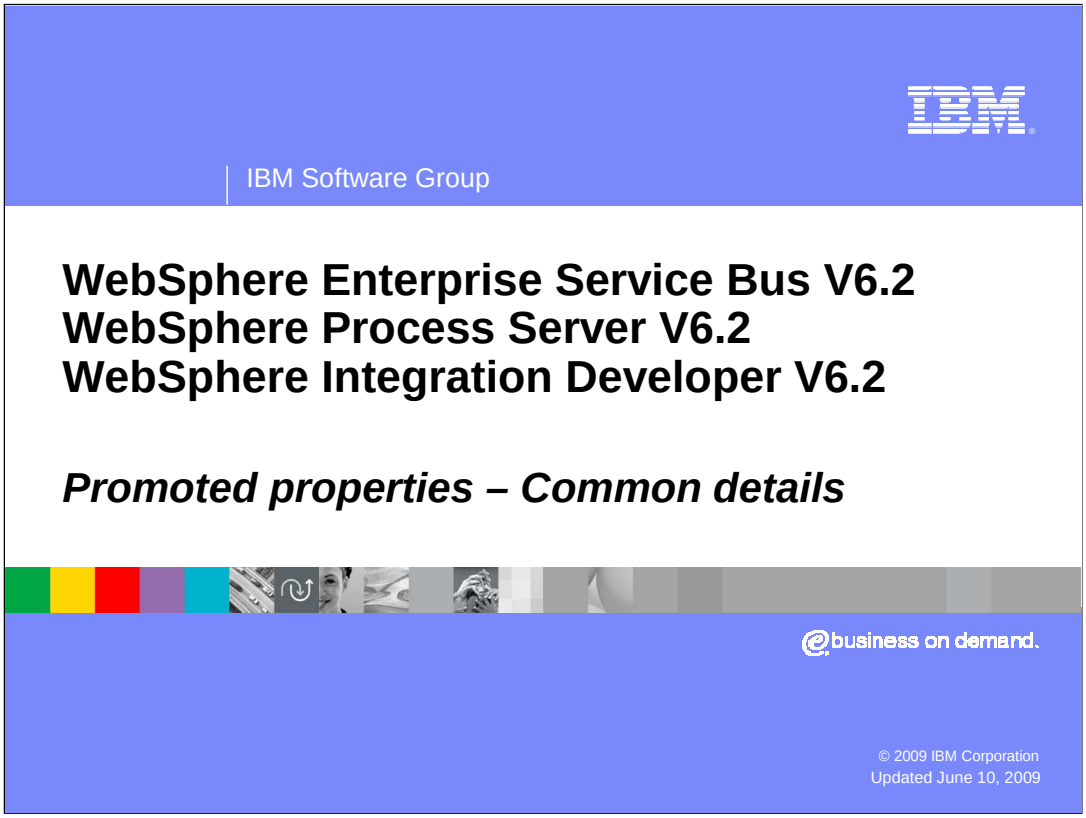
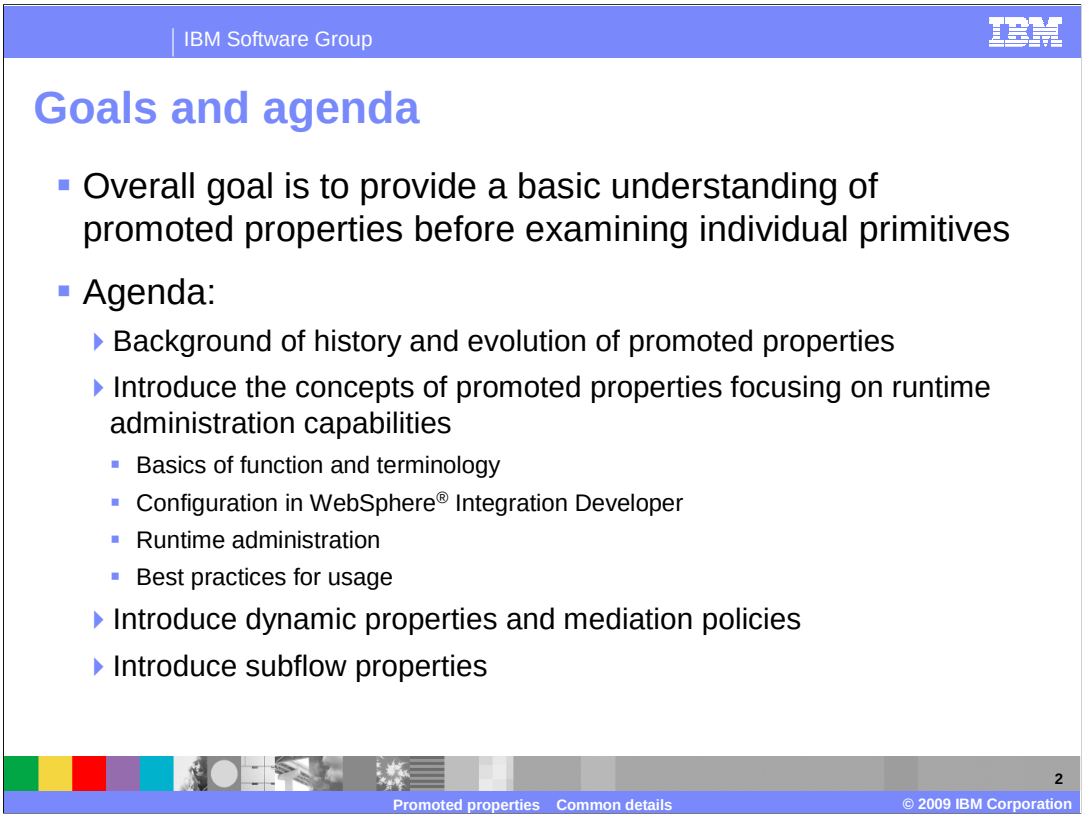


This presentation looks at the common aspects of promoted properties from both a development and a runtime perspective. There are multiple different uses of promoted properties that are introduced in this presentation.



The overall goal of this presentation is to introduce you to the concepts of promoted properties. With the information you learn from this presentation, you are ready to understand the specifics of promoted properties for each of the individual mediation primitive types.

The presentation starts out by providing some background information on the history and evolution of promoted properties, which helps to put in perspective the organization of the rest of the presentation.

The concepts of promoted properties are presented within the context of how they are used to enable runtime administration of mediation flows. This starts by introducing the terminology that is used with promoted properties. You are then shown how promoted properties are configured using WebSphere Integration Developer. The next topic examined is the various runtime administrative tasks and capabilities that enable administrators to have some control over aspects of a mediation flow. Once you have the full end to end picture of concepts, development, and runtime administration, additional details are presented with an emphasis on best practices for the use of promoted properties.

The final two topics explore uses for promoted properties other than runtime administration. The first is the concept of dynamic properties that are set through mediation policies. The other is the use of promoted properties to configure a subflow from a parent flow.

Section

Basics of function and terminology

The first section of the presentation introduces the basic functions and characteristics associated with promoted properties. In addition, the key terms that are used when discussing promoted properties are introduced and explained. As already mentioned, the emphasis is going to be on promoted properties used for runtime administration, with dynamic properties and subflows discussed at the end of the presentation.

Why promoted properties

- Characteristics without promoted properties:
  - Property values are statically defined during development
  - Property values cannot be modified at runtime
    - Changes can only be made using WebSphere Integration Developer
    - A modified mediation module must be redeployed to the server.
- Promoted properties enable administrative runtime changes, for example:
  - How much of the message to log
  - Toggle input validation for mediation primitives
  - Change values used to control flow paths

This slide addresses the rationale for having promoted properties.

In the absence of promoted properties, mediation property values are statically defined at development time using WebSphere Integration Developer. The values cannot be modified for a mediation application that is already installed in a WebSphere Process Server or WebSphere Enterprise Service Bus. If there is a need to modify a property value, the mediation must be modified in WebSphere Integration Developer and reinstalled into the server.

Promoted properties enable property values to be updated for a running application. This gives the solution administrator the flexibility to make changes to the mediation flow by administratively setting different values for the configuration properties. Here are some examples of how this might be used.

The first example is a case where there is a Message Logger primitive that is only logging a small portion of the message, such as part of the body. When trying to debug a runtime issue, it might be useful to see the entire message written to the message log, allowing headers and context information to be examined. This can be done by modifying the root property of the Message Logger.

Another example also involves problem resolution. In this case, there is a problem with a malformed message and an attempt is being made to determine where the message gets corrupted. Several of the mediation primitives have a validate input option, which is normally not used in a production system because of performance implications. However, in order to debug the malformed message problem, promoted properties can be used to turn on the validate input options. This allows you to catch the message corruption issue closer to where it originally occurred.

The third example is related more to an application change. Assume you have a mediation that uses a Message Filter to take different flow paths based on some criteria, such as a rating value. Promoted properties can be used to adjust the rating value threshold used to select a particular path.

Overview of function

- Solution administrators can modify property values
  - Can be properties of mediation primitives or of callouts
  - Changes can be applied during application installation
  - Changes can be applied while the application is running
    - The changes take effect dynamically
    - Server does not need to be restarted
    - Mediation module does not need to be reinstalled or restarted

An overview of the functionality of promoted properties is provided on the next few slides. As previously mentioned, promoted properties allow mediation property values to be modified for a running mediation application in a WebSphere Process Server or WebSphere Enterprise Service Bus. This is normally done by someone assuming the role of solution administrator.

The properties that can be modified include those associated with mediation primitives in the flow and properties associated with a flow's callout nodes.

At the time the application is installed, the property values assigned during development are used unless they are explicitly changed during the application installation process. Property values can also be changed for an application that is already installed and running. When this is done, the changes take effect dynamically. There is no need to restart the server, reinstall the application, or restart the application.

Overview of function

- Not all properties can be administered
  - Only certain properties are allowed to be administered
    - Restricted to properties whose value can change without:
      - requiring a redeployment
      - affecting some related artifacts created from the development time definition
    - This is determined by the mediation primitive or callout implementation
  - As the integration developer, you
    - Configure the mediation primitives and callouts for a specific mediation module
    - Identify which of the allowable properties can actually be administered

This slide examines which properties are available to have their values changed by a solution administrator.

The first thing to consider is that only certain mediation properties are eligible to be administered. For any given mediation primitive type, only selected properties can be made available to be administered. Those that cannot be administered generally have some characteristic that prevents them from being used in this way. One reason is that changing the value for a particular property can sometimes force an application redeployment, in order for the change to take affect in the running application. Another reason is that some properties define relationships to other development time artifacts, which requires the change to be made using WebSphere Integration Developer. Each mediation primitive type defines which of its properties are potentially available for administration and which are not.

Of those properties which are eligible to be administered, you, as the integration developer, choose which will actually be made available for a particular mediation flow and mediation primitive instance. It is only those properties which you designate that are made available for the solution administrator to modify.

The next slide introduces some terminology which makes these concepts easier to understand.

Concepts and related terminology

- Promotable
  - Identifies a property which can be available for administrative configuration
  - Applies to all instances of that mediation primitive type
- Promoted
  - Identifies a property which is available for administrative configuration
  - Determined by you when configuring the mediation primitive
  - Applies only to that one configured instance of the mediation primitive
- Alias
  - A name you give to the promoted property
  - Provides a meaningful way for a solution administrator to identify the property
  - An alias is associated with a value for the property
- Shared alias
  - Same alias name can be given to multiple properties in the same group
  - A common name causes the alias to be shared
  - Properties with a shared alias share a common value

The previous slide introduced some concepts about promoted properties and how they work. The terminology introduced here provides specific terms that make those concepts easier to understand and to discuss.

The first term in Promotable. This term is applied to a specific property associated with a specific mediation primitive type, if the property has the ability to be made available for modification by a solution administrator. Whether a property is promotable applies to all instances of the mediation primitive type.

The second term is Promoted. This term is applied to a property which is not only promotable, but has also been explicitly selected to be made available for modification by a solution administrator. This selection is made by you, the integration developer, when developing a mediation flow. Whether a property is promoted applies to only an individual instance of the mediation primitive to which that property belongs.

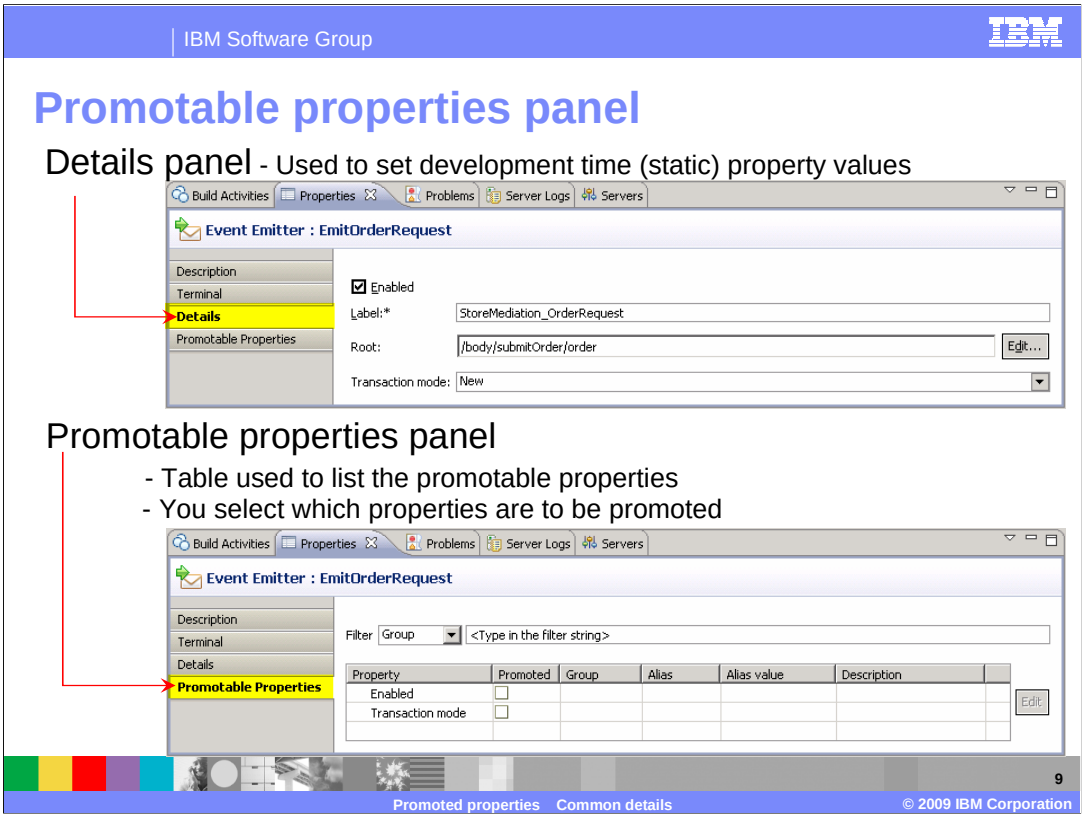
The next term is Alias. When a property is promoted, it is associated with an alias, which you can think of as a name value pair. You provide an alias name that has some meaning to the solution administrator. The alias name is associated with a value that is applied to the property. So the solution administrator, in essence, does not provide a modified value for the property. Rather, the value is associated with the alias and the modified alias value is applied to the property.

The final term is Shared Alias. Within a mediation module promoted properties are organized into groups. Within a group, promoted properties which have the same alias name share the same alias value. This allows multiple related properties within a mediation module to be administered together. An example might help to illustrate this. On a previous slide there was a scenario described where multiple validate input properties were to be toggled between off to on. By having a shared alias for these multiple validate input properties, the solution administrator can modify them all by only having to change a single alias value.

Section

Configuring promotable properties

Now that the basic concepts of promoted properties have been covered, this section looks at how to use WebSphere Integration Developer to configure promoted properties during development time.



In the properties view for a mediation primitive, two of the panels provided are the details panel and the promotable properties panel.

In the details panel are the various properties that are used to configure the mediation primitive, as shown in the upper screen capture. In the example shown, the panels are for an event emitter primitive which as four properties, enabled, label, root and transaction mode.

Looking at the promoted properties panel in the lower portion of the slide, you can see that only two of the four properties are promotable, enabled and transaction mode. The label and root properties are not promotable because these affect the contents of the emitted event. There might be external applications which have a dependency on the event format and therefore these are not good candidates for administrative control.

The promoted properties panel is composed of a table with a column for the promotable properties and a column to indicate if the property is promoted. When the property is promoted, the group, alias and alias value columns come into play. The description column is only provided for documentation purposes.

In these screen captures, neither of the promotable properties have been promoted.

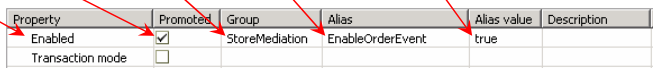
Promoting a property

Promoting a property

– Enabled property

– Select as promoted on Promotable Properties panel

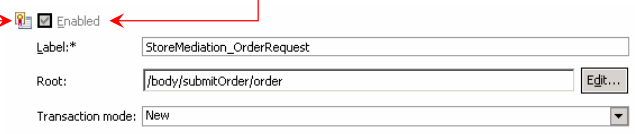
- Provide group, alias name and value



– Icon marks property as promoted on the details panel

– Clicking Promoted icon gets you to Promotable Properties panel

– Alias value is displayed but grayed out – cannot be changed here

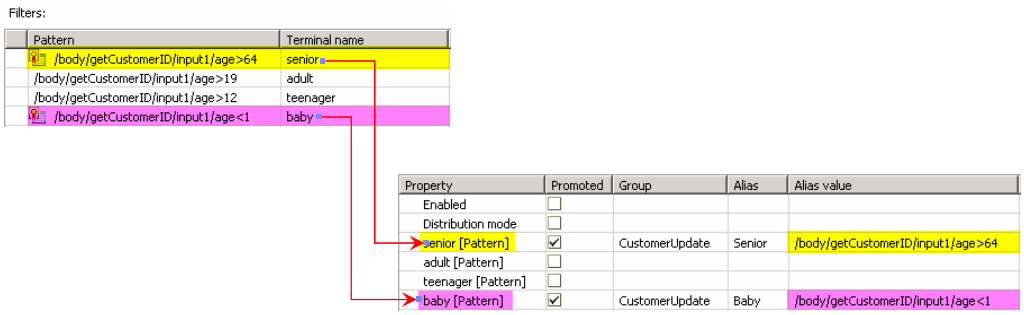


This slide illustrates the promoting of a property. The Promotable Properties panel is shown on the top. As you can see, the Enabled property has been marked as promoted and values for Group and Alias names and Alias value have been filled in.

Looking at the Details panel on the bottom, you can see that the Enabled property looks somewhat different than it did in the previous slide. First of all, an icon has been placed next to the property that indicates the property has been promoted. Also, the value for the property is shown but is no longer editable from this panel. If you click the icon, it switches you to the Promotable Properties panel which allows you to update the default alias value that is applied to the property.

Table based properties

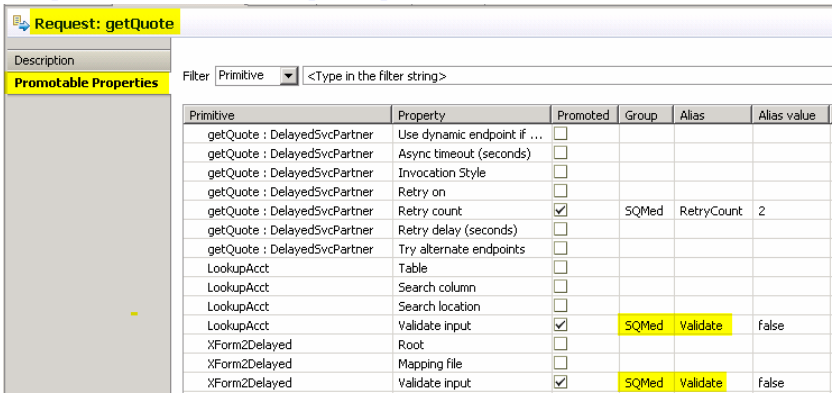
- Rows of a table can be prompted
  - Rows selected individually to be promoted
  - Only one column of the table is identified as promotable
    - For a promoted row, only the cell in the promotable column is administered
  - When promoted, group and alias names and a value are assigned
  - Table row must be created before it shows up as available to promote



One of the more interesting uses of promoted properties is the ability to promote a row in a table. It is actually only a single column in the table which is promotable. Individual rows in the table can be identified as promoted. For a row that is promoted, the alias value applies to the single table cell identified by the promotable column and the promoted row. When configuring a mediation primitive containing a table, the table rows must first be created in the Details panel. The individual rows then show up in the Promotable Properties panel so that then can be selectively promoted.

This slide illustrates this capability using the Filters property of a message filter primitive. The Filters property from the Details panel is shown on the left side. Basically, it is controlling the flow through the mediation based on the age of a customer identified in the message. Looking at the Pattern column, you can see that customers greater than 64 are considered senior. Customers greater than 19 through 64 are considered adult. Customers greater than 12 through 19 are considered teenager and customers less than one are considered baby. This leaves the default flow to be for customers who are one through 12, which are considered children. Looking at the filters table you can see by the icons that the greater than 64 and less than one patterns are shown as being promoted. Now look at the Promotable Properties panel on the lower right side. You can see the Terminal name is used to identify the row but, as indicated, it is actually the Pattern column that is promotable. For those rows that are promoted, the pattern becomes the default alias value. In this particular example, the solution administrator is given the capability to modify the age dividing adult from senior and the age dividing baby from child.

All promotable properties in a flow

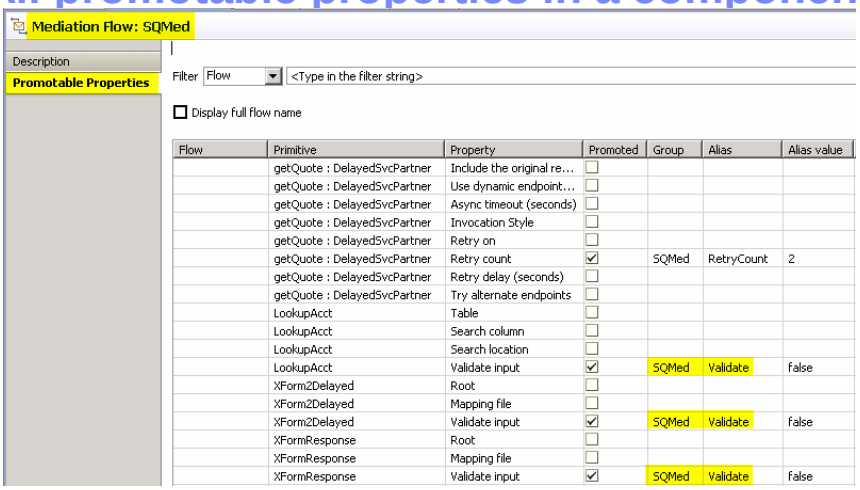


- Properties view for a mediation flow
  - Shows all properties for the flow
  - Properties can be edited from this panel
- Review overall usage of promoted properties in the flow
  - Identify shared aliases in the flow

This slide shows the table from the Promotable Properties panel that is associated with a mediation flow. It shows the promotable properties associated with an individual request flow or an individual response flow. It is slightly different than the table used for a mediation primitive's promotable properties. It includes a Primitive column to identify the specific primitive or callout node the property is associated with. This table can be edited to specify whether a property is promoted and for providing group and alias names and values. It provides a good way to review the overall use of promoted properties in a flow and to easily identify shared aliases in the flow. In the screen capture, the highlighted group and alias names for two of the properties show that they have the same values, indicating that they are using a shared alias.

Looking at the screen capture, notice in the Primitive column the entries that are labeled getQuote colon StockPartner. This identifies a promotable property associated with a callout node, rather than a property for a mediation primitive.

All promotable properties in a component



- Properties for an entire mediation flow component
  - Properties view of the Operations Connection panel
  - Adds Flow column identifying the flow containing the property
- Use this to check and verify appropriate shared alias usage

This slide shows the Promotable Properties panel that is associated with an entire mediation flow component, encompassing all the request and response flows defined within the component. It is part of the Properties view of the Operation Connections panel of the Mediation Flow Editor. It is similar to the table for an individual request or response flow, with the addition of the Flow column and a property to indicate if full or abbreviated flow names should be used. The addition of the flow name helps to distinguish between primitives in different flows that have the same name.

You will notice that in the screen capture the Flow column is blank. This is due to a problem that existed in WebSphere Integration Developer at the time this presentation was developed. Hopefully, the version of WebSphere Integration Developer you are using has this problem fixed and the flow names are displayed.

The same editing capabilities described for the table on the previous slides are available for this table as well. The key use for this particular table is to ensure that you have configured your shared aliases correctly, which is discussed in more detail later in this presentation.

All promotable properties in a module

- Promoted properties at the module level
  - Modules and mediation modules can have more than one mediation flow component
  - Shared aliases are scoped at the module level
    - Promoted properties from two different mediation flow components with the same group/alias name are shared
- Pitfalls you need to be aware of:
  - There is no mechanism provided to review all the promotable/promoted properties at the module level
    - Makes it difficult to review for shared alias usage
  - The default group name is the name of the module
    - Primitives in different components that have the same name by default have shared aliases for their promoted properties
  - There is no management of shared alias values across mediation flow components

The table on the previous slide addressed all the promotable properties in a mediation flow component. In most cases, there is only one mediation flow component in a module or mediation module. When that is the case, the previous table is sufficient to review promotable and promoted properties for an entire module. However, it is possible to have more than one mediation flow component in a module. The considerations for that are addressed on this slide.

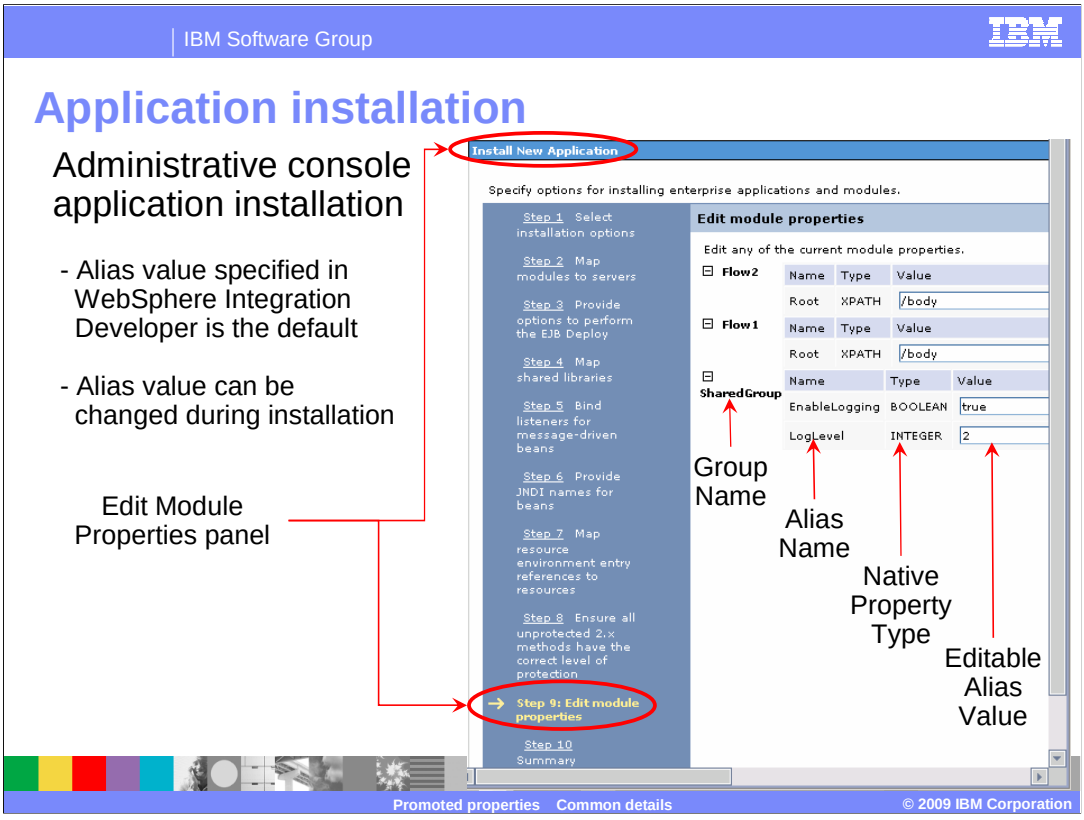
Shared aliases are scoped at the module level. Therefore, when promoted properties from two different mediation flow components have the same group and alias names, they are shared. Unfortunately, this introduces some pitfalls that you should be aware. First of all, there is no mechanism provided to review all of the promotable and promoted properties at the module level. This makes it difficult to review the promoted properties to verify that the shared aliases defined are in fact those that you intended. The next problem making this more difficult to manage is that the default group name given to a promoted property is the name of the module name rather than the name of the mediation flow component. As a result of this, primitives from different mediation flow components that have the same name will by default have shared aliases for their promoted properties. Finally, there is no management of shared alias values across mediation flow components. For example, within a mediation flow component with a shared alias, when the value of one of the promoted properties is updated, the value is also updated for all other properties that share the same alias. This does not occur across shared aliases in different components.

Considering all of this, you need to be very careful if you are using promoted properties in a module with multiple mediation flow components.

Section

Runtime administration

This section shows you about how promoted properties are administered at runtime, looking at both installation and administrative runtime updates.



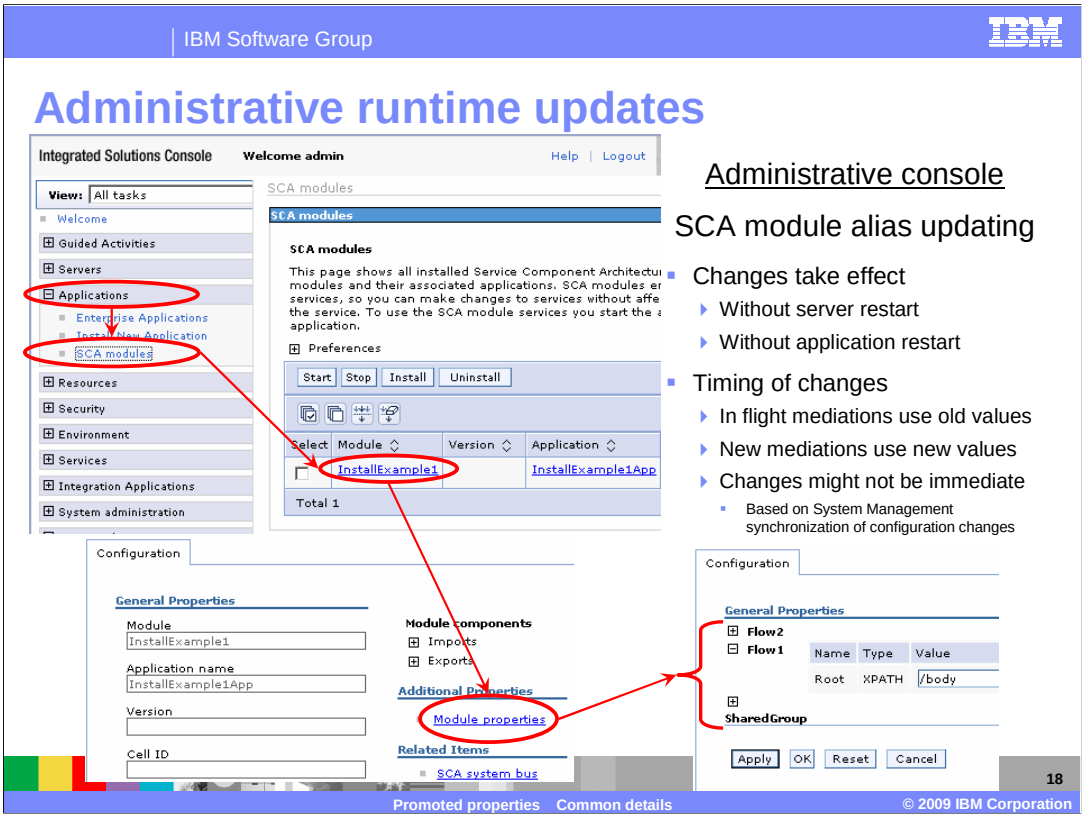
This is a screen capture of the administrative console when doing a new application installation for an application containing a module with a mediation flow component. The left side contains a list of panels which are relevant to the installation of an application. Shown in the screen capture is the 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 to 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, same as when you make any other change made using the administrative console.

Once the configuration is saved, the new alias value applies to the mediation module. 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.

Section

Additional details and best practices

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.

Alias names and shared aliases

- Alias names
  - Are qualified by a group name which defaults to the module name
  - Default alias name = <mediation_primitive_name>.<property_name>
  - You can (should) change the default group and alias names to have more meaningful values
- Shared alias
  - Multiple properties can share a common alias
    - Scope of sharing is the module
      - Encompasses all the flows in all the mediation flow components in the module
    - Having the same group name and alias name defines the sharing association
  - Alias value updating
    - Can update on Promotable Properties panel for any property sharing the alias
    - Shared value is automatically updated for all properties sharing the alias
      - **Note**: This is not true for shared aliases across multiple mediation flow components
  - Normally used to control a common property
    - Example – toggle the Enabled property on all message loggers

More details on alias names and shared aliases are provided here, expanding upon what was previously covered.

First, when you mark a property as being promoted, WebSphere Integration Developer assigns a default group name and alias name. The default group name is the name of the module. The default alias name is composed from the mediation primitive name and property name, with an intervening dot, as shown in the slide. In general, you should change the generated alias name to something that is more meaningful to both you and the solution administrator. In some cases, you might also what to change the group name.

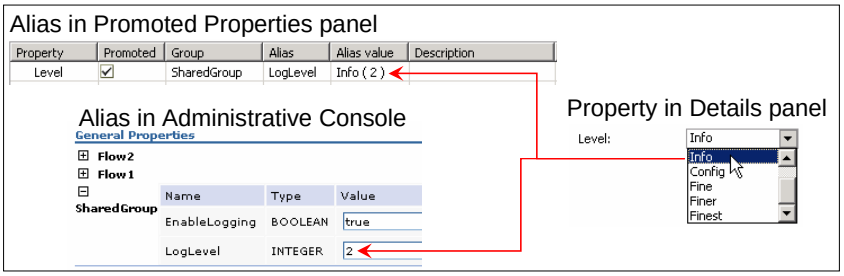
As has been discussed, multiple properties can have a shared alias by using the same group name and alias name, which results in them having the same value assigned. The scope of this sharing is the entire module or mediation module. This encompasses all of the flows within all of the mediation flow components contained in the module. Because of this, it is important to make use of the Promotable Properties panel associated with the Operation Connections panel of the Mediation Flow Editor, which was shown on a previous slide. This allows you to check all aliases assigned within a component to determine if all the sharing relationships are correct. In most cases, where you have only one mediation flow component in the module, this check is sufficient. However, if you have more than one mediation flow component in the module, there is no equivalent panel to check everything. In this case, you need to reconcile the shared aliases across multiple Promotable Properties panels.

The next point to mention is related to the updating of the value for a shared alias. The value for the shared alias can be updated on any Promoted Properties panel on which it is found. When the value is updated, the update is reflected in all other Promoted Properties panels which contain that alias. This is true within all flows of a mediation flow component, but is not true across multiple mediation flow components.

An example of the usage of a shared alias is to toggle the Enabled property, which is contained on some mediation primitives, such as the Message Logger. By having a shared alias for all of the message logger Enabled properties, logging can easily be turned on and off.

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

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.

Best practice considerations

- Alias names
  - Do not use the default value for the alias
  - Choose a name that is meaningful to an administrator
  - Do not use the default group name if you have more than one mediation flow component in the module
- Determining which properties to promote
  - Only promote properties which can be meaningfully changed for your specific scenario
  - Decision made through agreement between you and the administrator
- Document usage to administrator
  - Administrator is not likely to understand details of the flow
  - Documentation should describe when and why alias value should be changed
  - Need to document valid values:
    - Numeric representations of enumerations
    - XPath values – valid values and potentially useful values

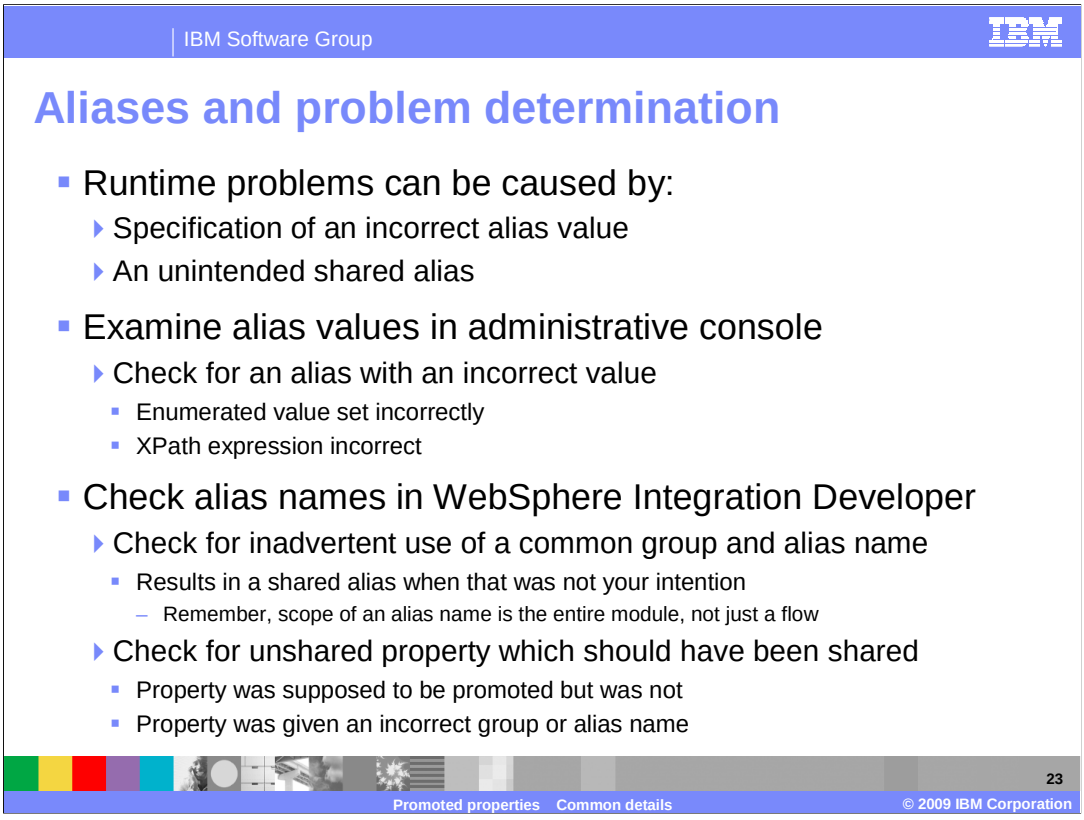
Here are some best practices that can help you to make the best use of promoted properties.

First, it is always best to use alias 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.

Also, if you have more than one mediation flow component in your module, do not use the default group name. This will help avoid unintended shared aliases across components.

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 and why each might be used should be documented.



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.

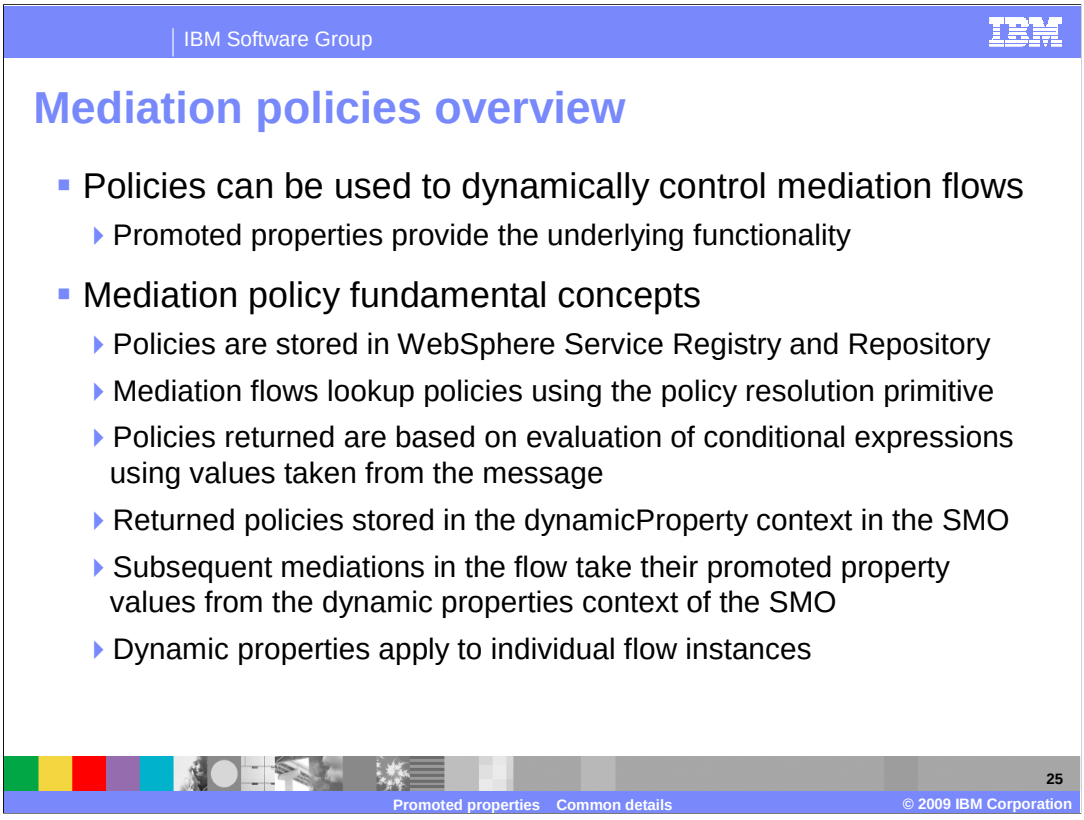
One problem might be the inadvertent use of a common group and alias name, thus establishing a sharing relationship that you did not intend. Remember that the scope of sharing is the entire module. Use the Promoted Properties panel associated with the Operation Connections of the Mediation Flow Editor 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 just was not promoted when it should have been and can also occur if the alias name was not specified correctly.

Section

Dynamic properties and mediation policies

Up until this point in the presentation, the concepts of promotable and promoted properties have been examined from the point of view of allowing administrative control over mediation flows. This section briefly examines how promoted properties are used with dynamic property values to enable mediation policy support for the contextual control of flows.



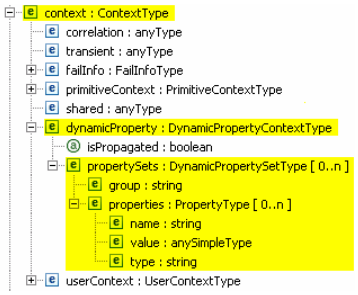
The concept of dynamic properties was introduced to enable support for mediation policies. This is a brief summary of mediation policies and how they work. It will help to provide background for the short discussion of dynamic properties that follows.

Mediation policies are used to provide dynamic control over mediation flows. They make use of the promoted property capabilities as the underlying support for controlling a flow.

The fundamental concepts of mediation policies and the essential steps associated with them are provided 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 dynamic properties section of the SMO context. Subsequent mediation primitives in the flow resolve their promoted properties by taking the values from the dynamic properties section of the SMO context. 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: DynamicPropertySetContext
    - 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



The screen capture on the right shows the context section of the SMO with the dynamic properties 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 content in the dynamic property context in the SMO has equivalent information to the module properties maintained by the runtime application configuration.

How promoted properties work

- Resolution of promoted property values at runtime
  - If group/property found in dynamic properties context, use the value
  - Obtain value from the module properties
- Contrasting module properties 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

So how do these 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.

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 as 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 to 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.

Section

Subflows and promoted properties

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.

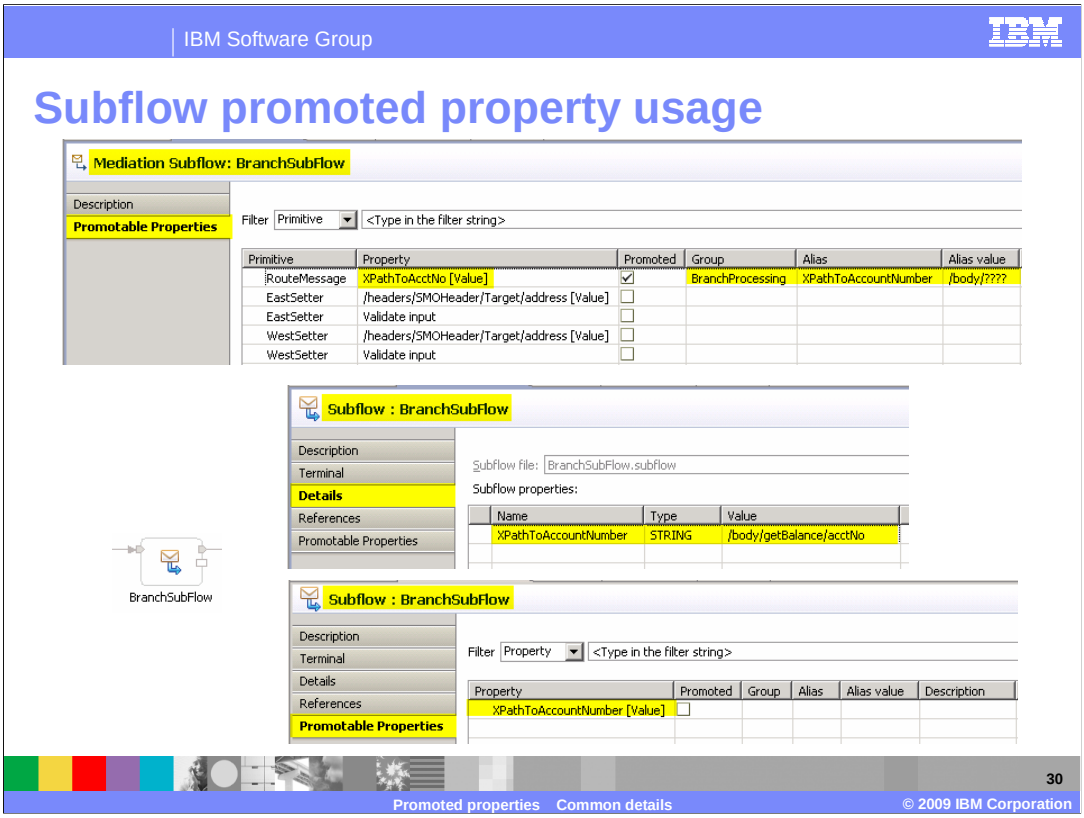
Subflows overview

- 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, 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.



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.

Section

Summary

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

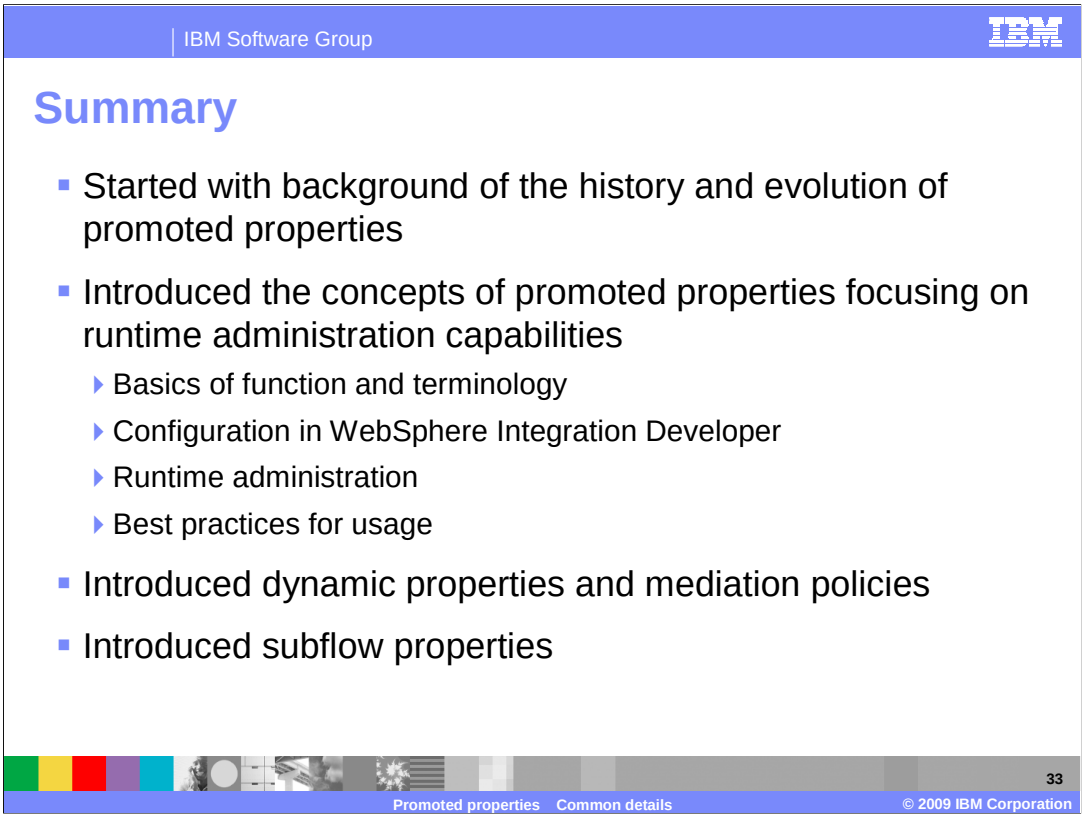
The overall use of promoted properties is somewhat overloaded. Although there is a single mechanism provided for promoting properties, there are three distinct uses for them. The different uses of promoted properties have varying characteristics. Understanding these helps to put the use of promoted properties 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 out by providing some background information on the history and evolution of promoted properties. Hopefully, this helped to put into perspective the organization of the rest of the presentation.

The concepts of promoted properties were presented within the context of how they are used to enable runtime administration of mediation flows. This started by introducing the terminology that is used with promoted properties. You were then shown how promoted properties are configured using WebSphere Integration Developer. The next topic examined was the various runtime administrative tasks and capabilities that enable administrators to have some control over aspects of a mediation flow. Once you had the full end to end picture of concepts, development, and runtime administration, additional details were presented with an emphasis on best practices for the use of promoted properties.

The final two topics explored uses for promoted properties other than runtime administration. The first was the concept of dynamic properties that are set through mediation policies. The other was the use of promoted properties to configure a subflow from a parent flow.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WBPMv62_CommonDetailsPromotedProperties.ppt

This module is also available in PDF format at: ../WBPMv62_CommonDetailsPromotedProperties.pdf

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.