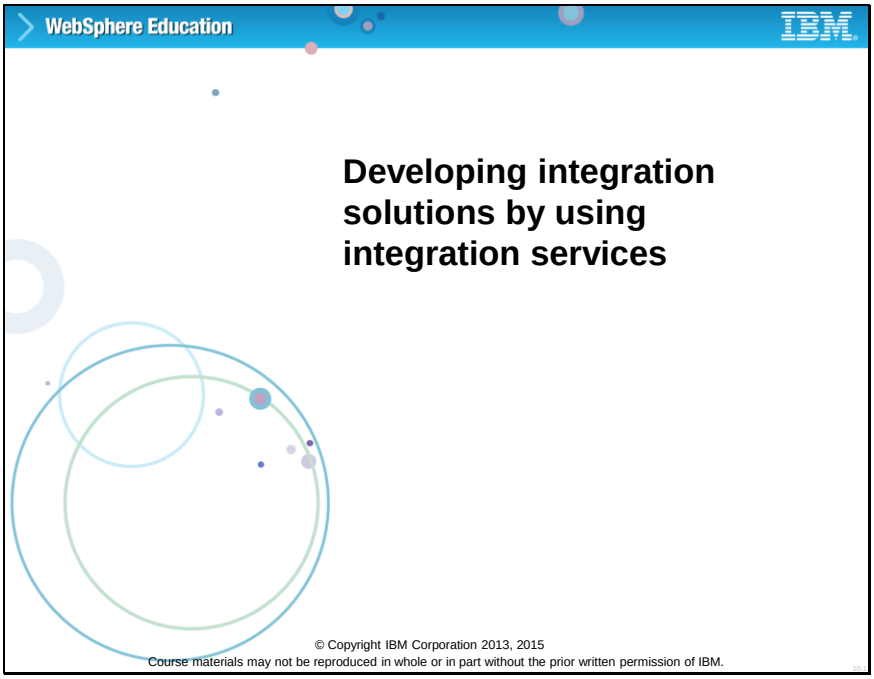
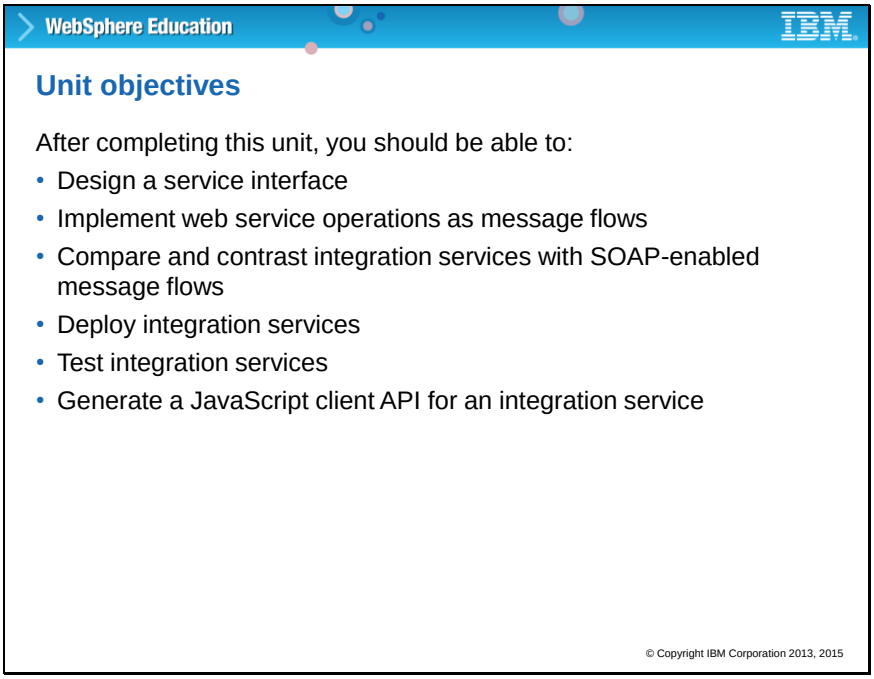
Slide 1


WebSphere Education
IBM
Developing integration solutions by using integration services
© Copyright IBM Corporation 2013, 2015
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Slide 2



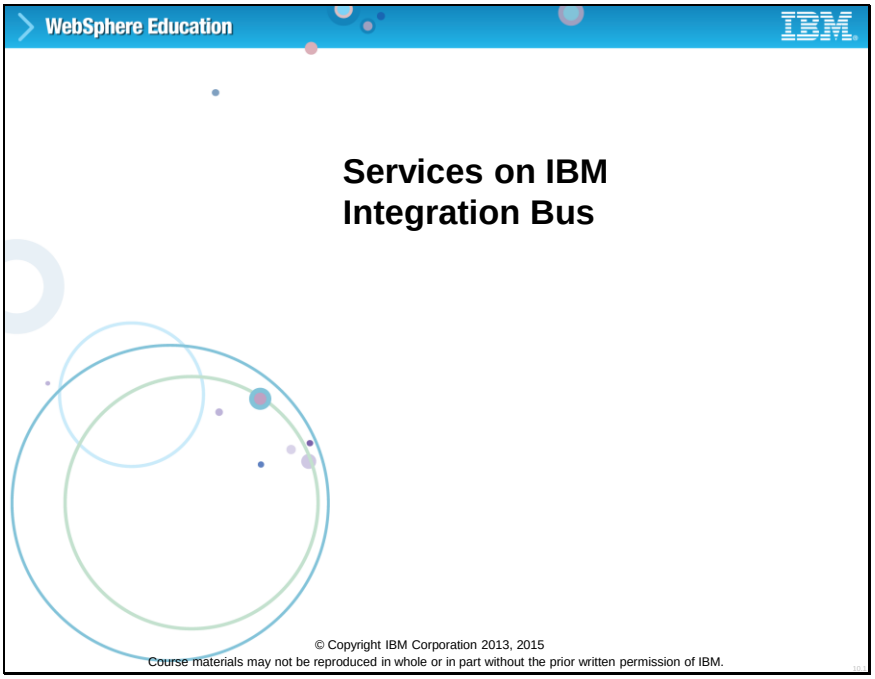
Unit objectives

An integration service is a specialized application with a defined interface and structure that acts as a container for a web services solution. In this unit, you learn how to create and implement IBM Integration Bus integration services.

After completing this unit, you should be able to:

- Design a service interface
- Implement web service operations as message flows
- Compare and contrast integration services with SOAP-enabled message flows
- Deploy integration services
- Test integration services
- Generate a JavaScript client API for an integration service

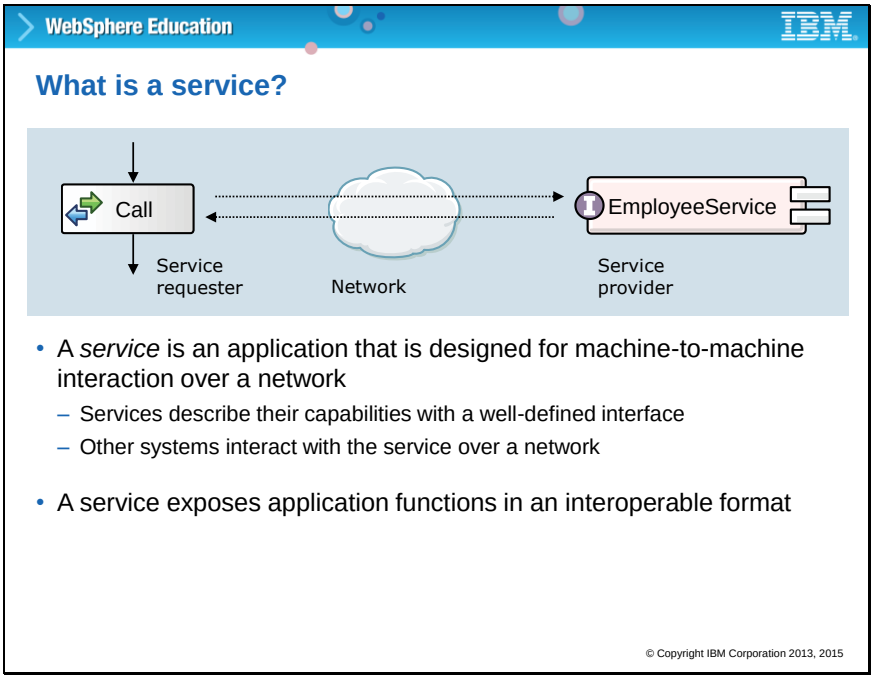
Slide 3



Topic 1: Services on IBM Integration Bus

This topic introduces services and services support in Integration Bus.

Slide 4



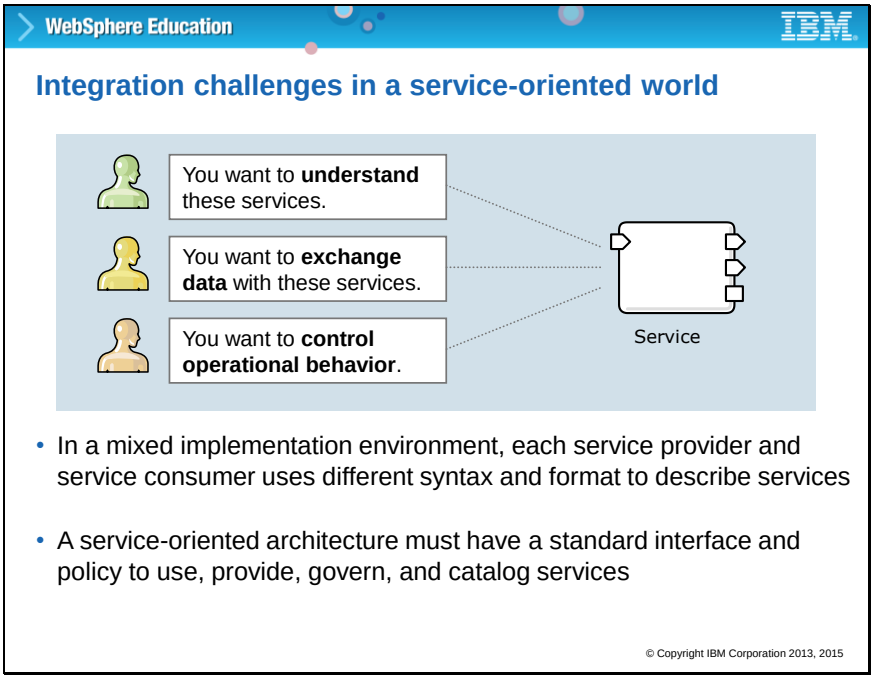
What is a service?

A service is an application that is designed for machine-to-machine interaction over a network. Services describe their capabilities with a well-defined interface. The service hides its implementation details from the service requester. Other systems interact with the service over a network.

The purpose of a service is to expose application functions in an interoperable format. Its definition does not force you to use a specific interface format, such as a WSDL, or specify a messaging format, such as SOAP.

A service provides application functions that are exposed through a standard interface.

Slide 5



Integration challenges in a service-oriented world

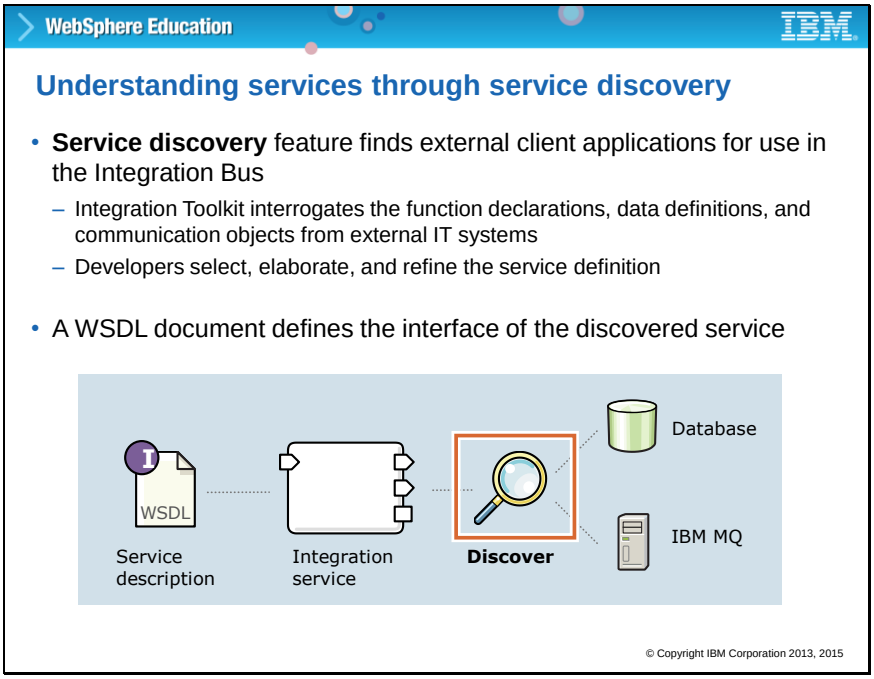
What are the challenges of a service-oriented architecture?

In a mixed implementation environment, each service provider and service consumer uses different syntax and format to describe services. For example, an MQ service describes the data structure for its messages as a COBOL copybook. The service consumer defines a request message with XML schema.

A mixed implementation environment is also known as a heterogeneous environment. Many businesses have a mix of implementation types: MQ applications, database applications, Enterprise Java applications, and enterprise information systems. These systems define services in different ways.

The main goal of a service-oriented architecture is interoperability. However, if services do not use a standard syntax or format, service consumers and providers cannot communicate.

Slide 6



Understanding services through service discovery

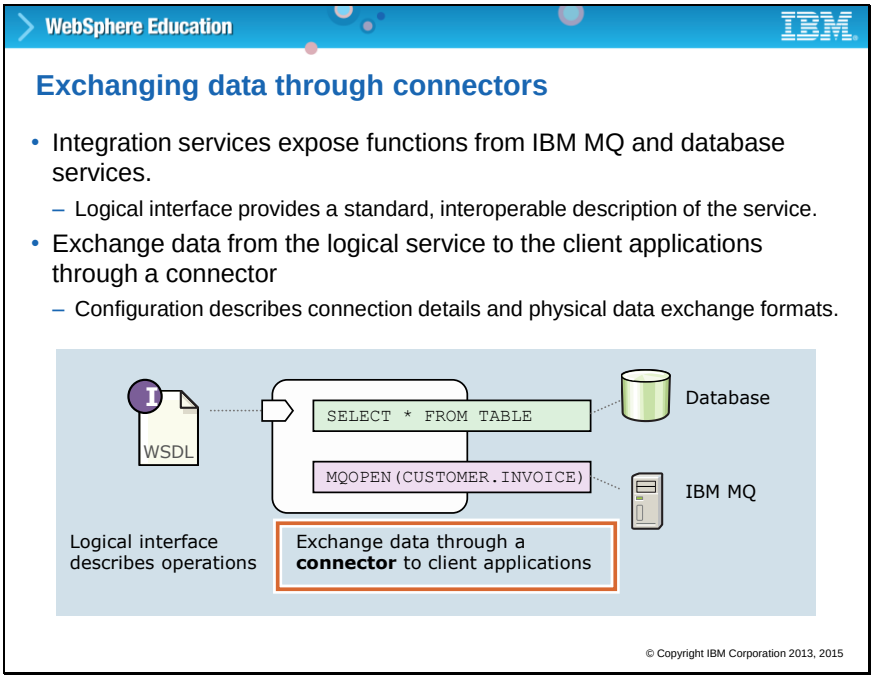
How do you gain an understanding of the services in your environment?

You can use the Integration Bus service discovery feature to find external client applications. Two examples of external applications are database applications and MQ applications.

During service discovery, the Integration Toolkit interrogates the function declarations, data definitions, and communication objects from external IT systems. Developers select, elaborate, and refine the service operation.

The purpose of the service discovery feature is to survey and standardize server applications as SOAP web services. A WSDL document defines the interface of the discovered service. The WSDL document is the standard interface for these enterprise services.

Slide 7

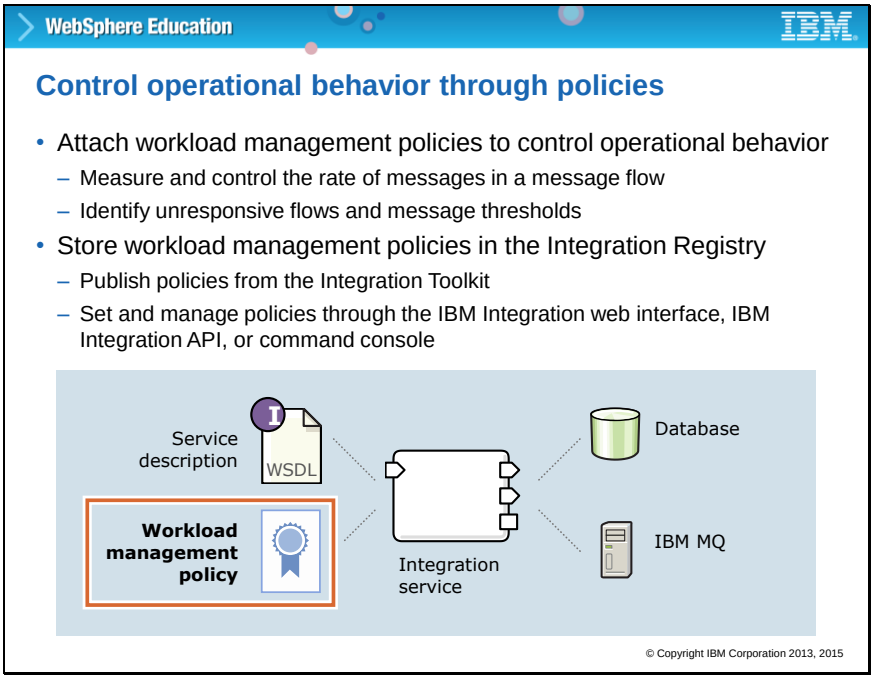


Exchanging data through connectors

The integration service is a container for enterprise applications, in the form of a SOAP web service. The web service interface is a logical view to the MQ or database applications. This interface provides a standard, interoperable description of the service.

A connector is used to exchange data between the integration service and the enterprise applications. The configuration for the connector describes connection details and physical data exchange formats.

Slide 8



Control operational behavior through policies

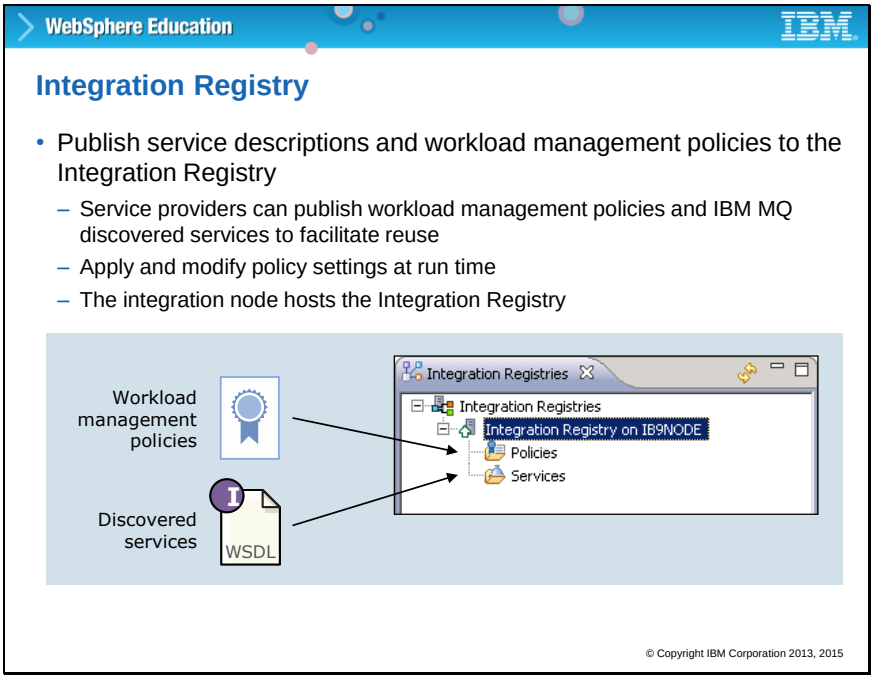
You can control operational behavior by using policies.

For example, you can apply workload management policies to control the operational behavior of deployed integration services. These policies can measure and control the rate of messages in a message flow. Workload management policies can also identify unresponsive flows and message thresholds. You also define policies for MQ and MQTT endpoints.

Policies are stored in the integration node's Integration Registry.

You can set and manage policies through the Integration web user interface, Integration API, or command console. You can also publish some policies, such as MQ and MQTT endpoint policies, from the Integration Toolkit.

Slide 9



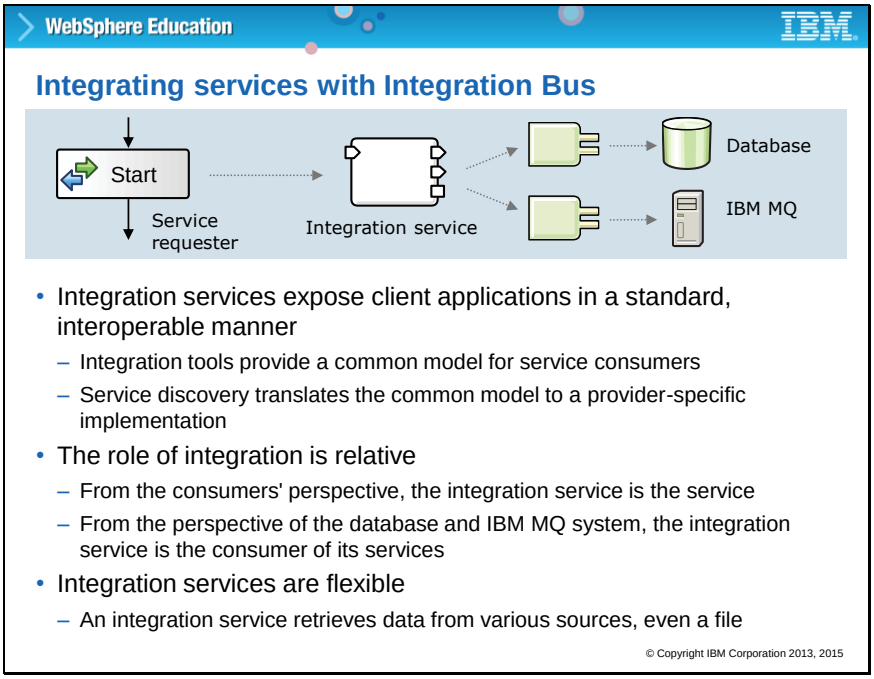
Integration Registry

The integration node's Integration Registry stores service descriptions for discovered services and workload management policies.

Developers creating integration services can retrieve and incorporate discovered services in message flows. At run time, the administrator can apply policies to services.

The developer can access the policies by using the **Integration Registries** view in the Integration Toolkit and selecting the integration node.

Slide 10



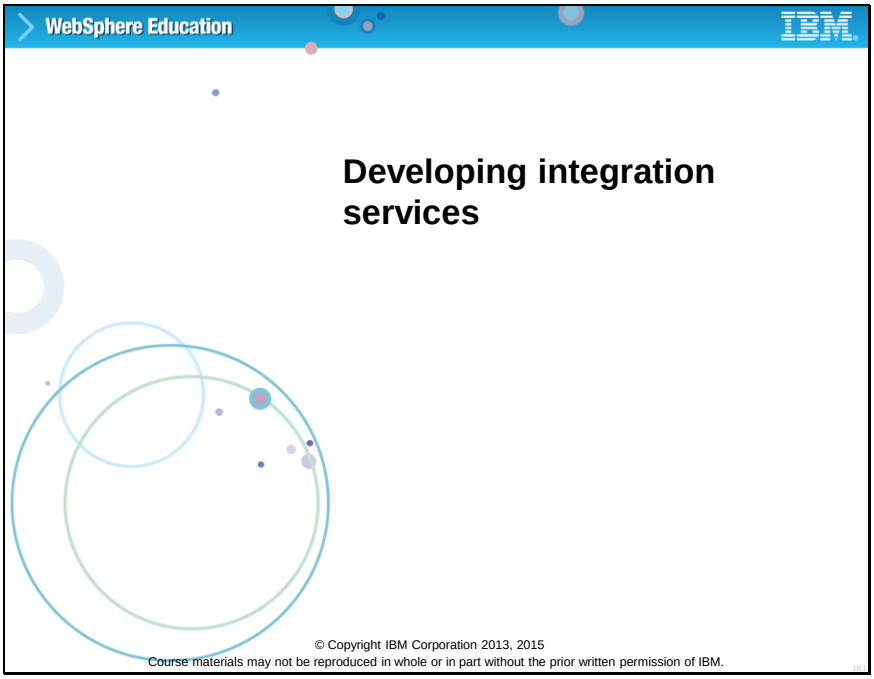
Integrating services with Integration Bus

Integration services expose enterprise applications as standard, interoperable web services. Integration tools provide a common model for service consumers. The service discovery feature in the Integration Toolkit converts the common model to a provider-specific implementation.

The role of integration is a matter of perspective. From the point of view of the consumer, the integration service is the service. From the point of view of the enterprise application, the integration service is the consumer of its services.

Integration services are flexible. For example, an integration service retrieves data from various sources, even a file.

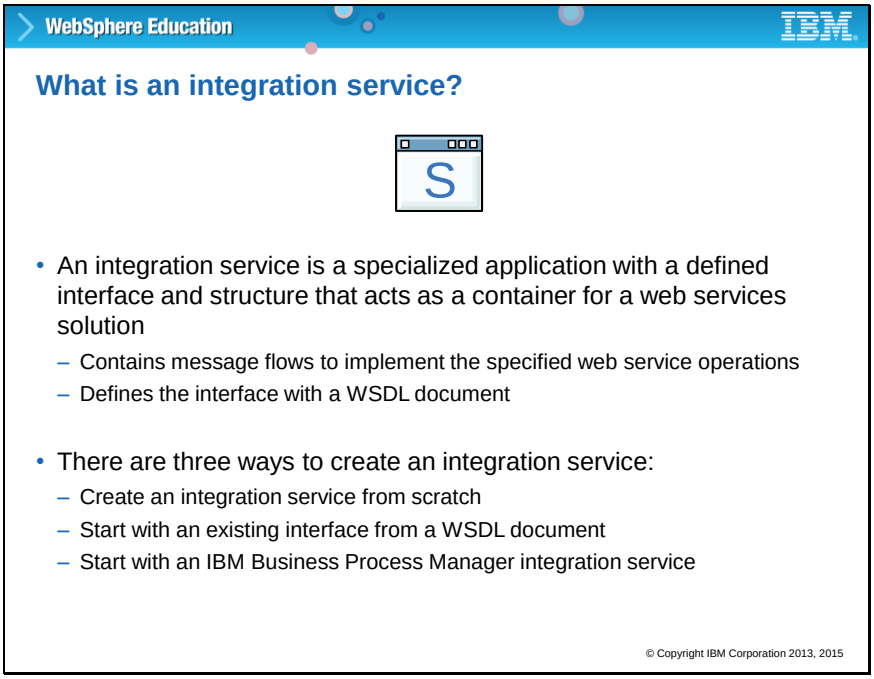
Slide 11



Topic 2: Developing integration services

In this topic, you learn how to design a service interface by using the Integration Toolkit. This topic also compares and contrasts integration services with SOAP-enabled message flows.

Slide 12



What is an integration service?

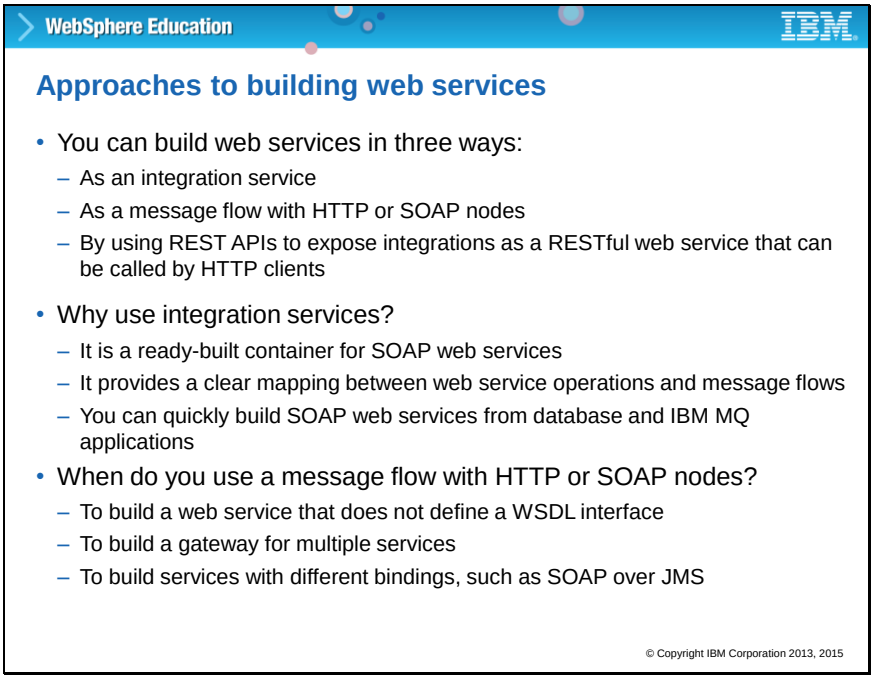
An Integration Bus integration service is a specialized application with a defined interface and structure that acts as a container for a web services solution. It contains message flows to implement the specified web service operations. It defines the interface with a WSDL document.

Integration Bus supports three ways to create an integration service in the Toolkit:

- Create an integration service from a blank canvas.
- Start with an existing interface from a WSDL document.
- Start with an IBM Business Process Manager integration service.

This unit describes how to create an integration service from a blank canvas and start from a WSDL document.

Slide 13



Approaches to building web services

As you learned in the previous unit, you can build web services in three ways: as an integration service, as a message flow with HTTP or SOAP nodes, or by using REST APIs to expose integrations as a RESTful web service that HTTP clients can call.

Why should you use integration services?

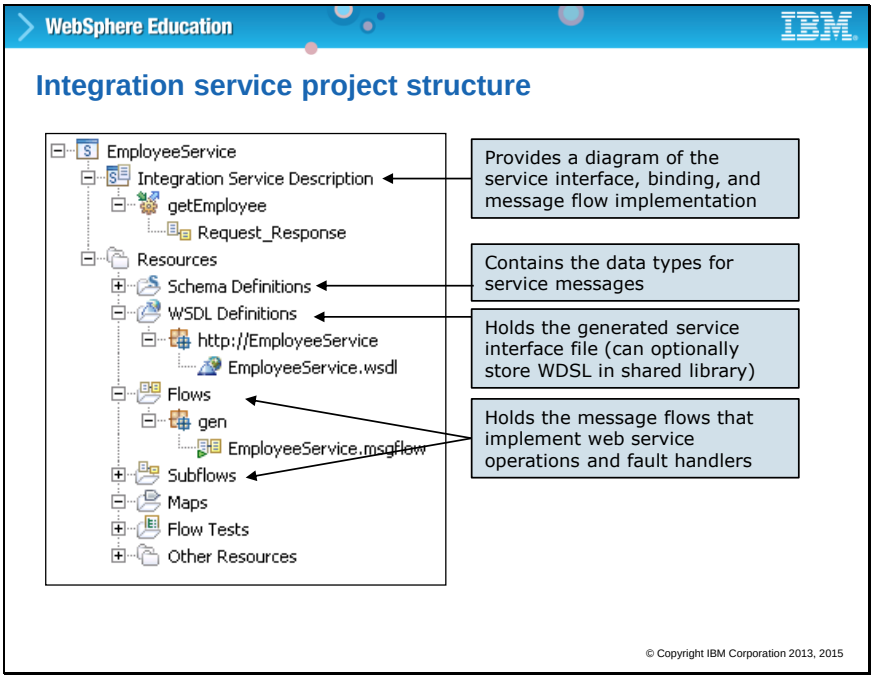
An integration service is a ready-built container for SOAP web services. It provides a clear mapping between web service operations and message flows. You can quickly build SOAP web services from database and MQ applications.

When should you use a message flow with HTTP or SOAP nodes instead of an integration service?

Use HTTP or SOAP nodes to build web services that do not define a WSDL interface. For example, RESTful services do not use the WSDL specification to define the interface or implementation details. In this case, use HTTP nodes to map REST resources to message flow features.

The previous unit described the message flow approach to web services. This unit focuses on using integration services.

Slide 14



Integration service project structure

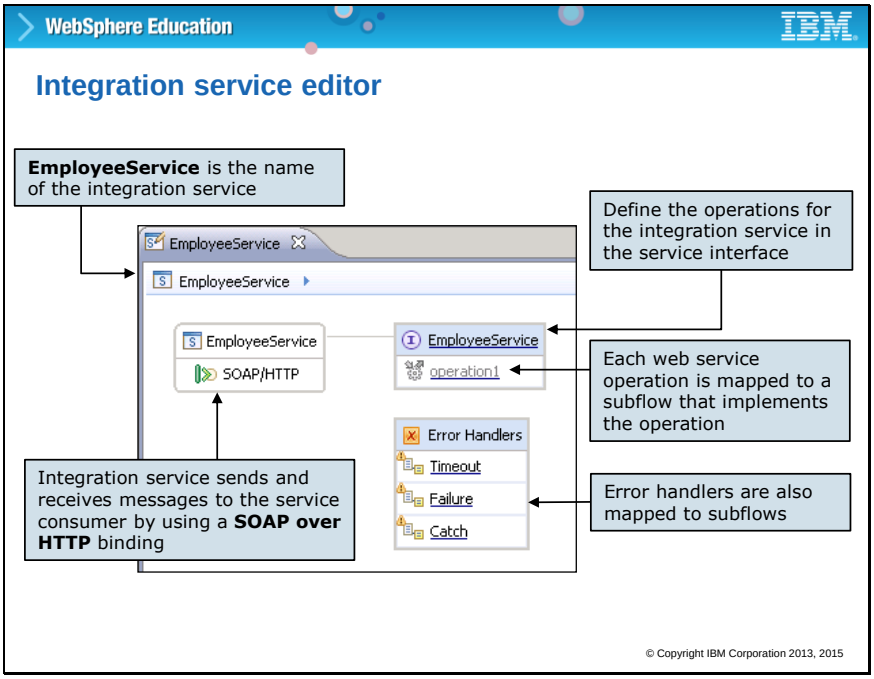
In the Integration Toolkit, you contain integration services in an integration service project.

The main components of an integration service project are the **Integration Service Description** and the **Resources**.

The **Integration Service Description** folder contains a diagram of the service interface, binding, and message flow implementation.

The **Resources** folder contains the components of the integration service and includes schemas, flows, subflows, and WSDLs.

Slide 15



Integration service editor

When you double-click the **Integration Service Description** folder in the **Application Development** view, the Integration Service editor opens.

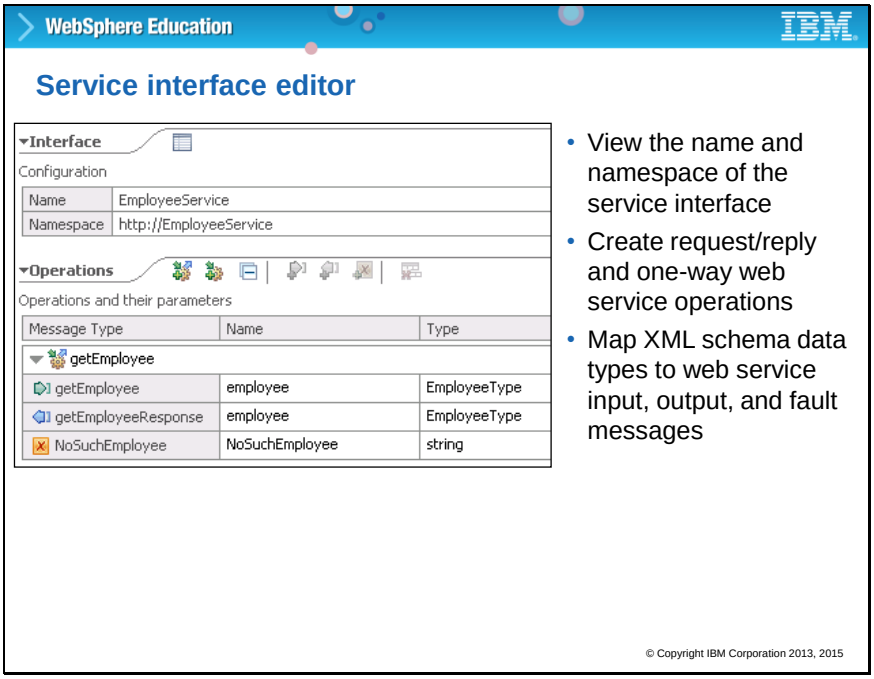
The Integration Service editor has two tabs: **Service** and **Interface**.

This slide shows an example of the **Service** tab contents for an integration service that is named EmployeeService. The editor also indicates that this service provides a SOAP over HTTP binding for its web service endpoint.

In this example, the EmployeeService service interface defines one operation: operation1. A subflow defines the implementation for each web service operation.

Separate from the web service interfaces are three error handlers: Timeout, Failure, and Catch. You can customize the behavior for each of the three error handlers by editing the subflows for each handler.

Slide 16



Service interface editor

On the **Interfaces** tab of the Service interface editor, you can create, modify, and view web service operations for the integration service. You can create a one-way operation or a two-way request/reply operation with an input and output message.

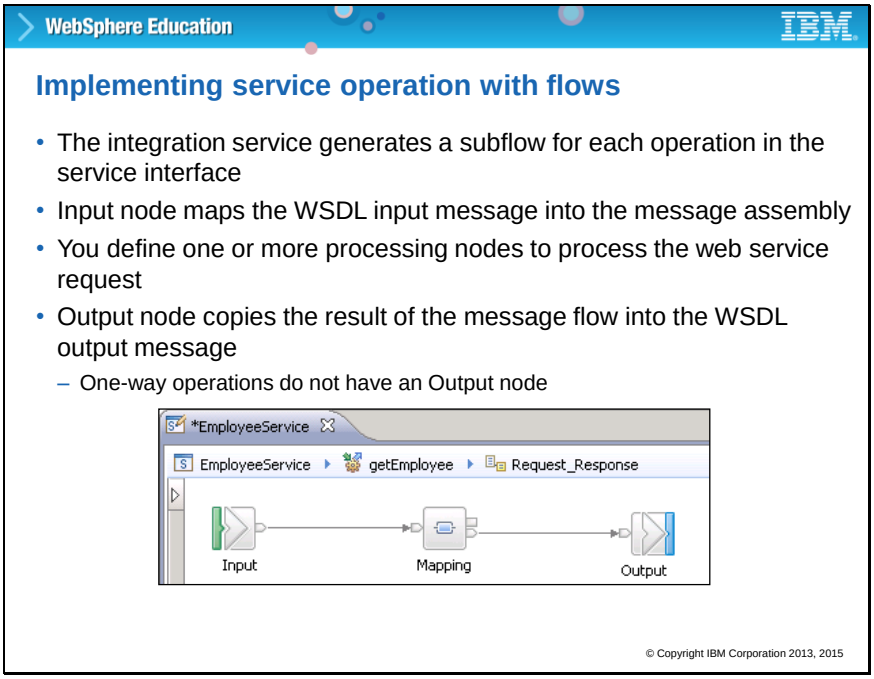
This slide shows an example of the **Interface** tab of the Integration Services editor for the EmployeeService. In this example, the getEmployee operation accepts an input message that is named getEmployee. The web service operation returns the result in a getEmployeeResponse message.

When you create one-way web service operation, it has an input message only.

For both types of operations, you can specify one or more fault messages. Web service fault messages represent exception conditions that occurred in the web service execution.

You can also review the name and namespace for the integration service. However, you cannot edit these properties in this editor.

Slide 17



Implementing service operation with flows

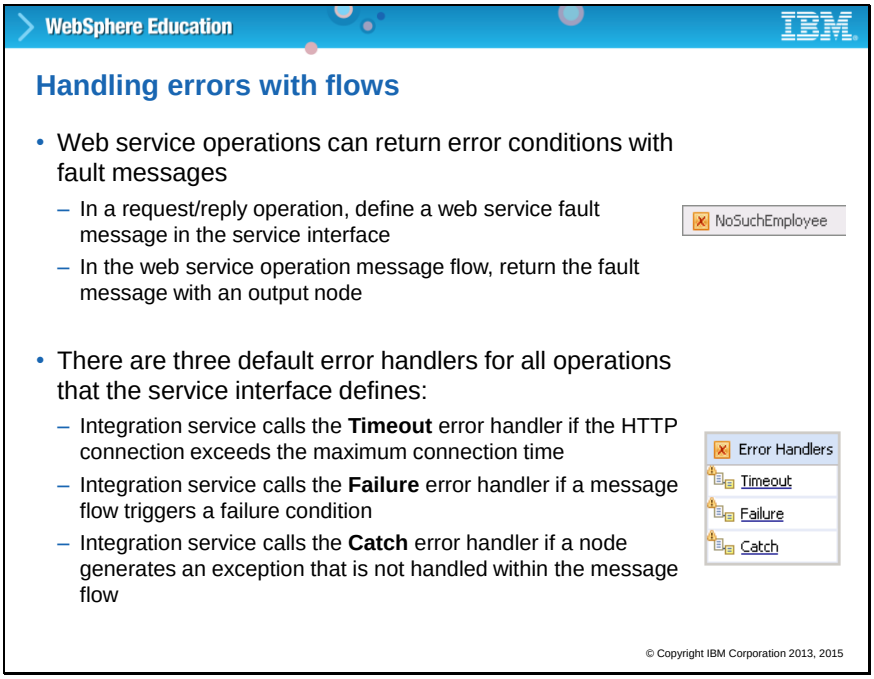
The integration service generates a subflow for each operation in the service interface.

If the operation is two way, the subflow contains an Input node and an Output node, corresponding to the input and output messages for the service operation. The Input node in the subflow maps the WSDL input message into the message assembly. The Output node in the subflow copies the result of the message flow into the WSDL output message.

If the operation is one way, the subflow contains an Input node only because a one-way operation does not return an output message to the client.

You complete the implementation by adding one or more processing nodes in the operation subflow.

Slide 18



Handling errors with flows

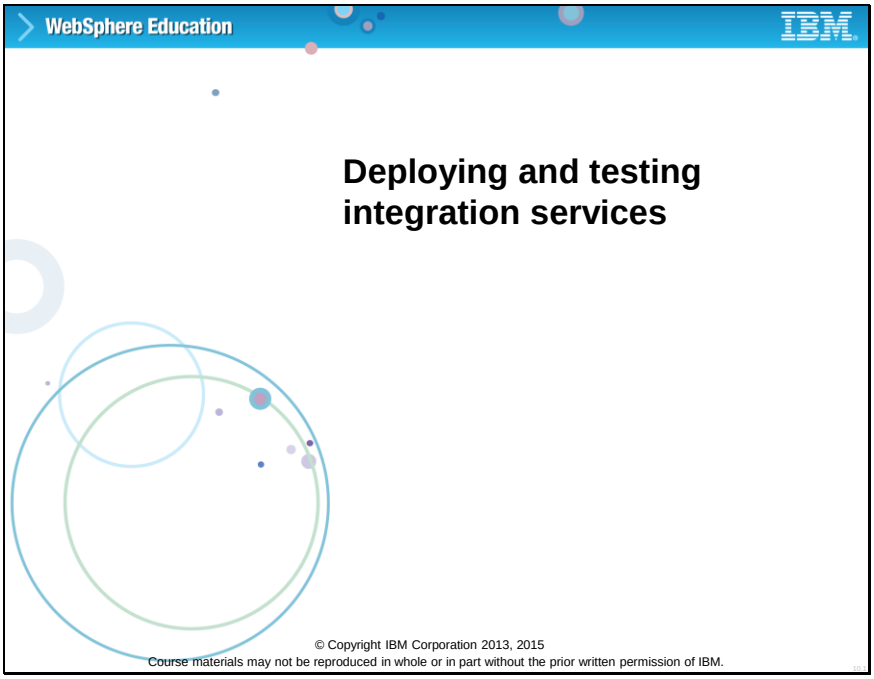
Web service operations can return error conditions with fault messages.

An integration service provides three default error handlers for all operations:

- If the HTTP connection exceeds the maximum connection time, the **Timeout** error handler is called.
- If a message flow triggers a failure condition, the **Failure** error handler is called.
- Iif a node throws an exception that is not handled within the message flow, the **Catch** error handler is called.

To handle the error message, define a web service fault message in the service interface. Then, in the operation subflow, return the fault message with an Output node.

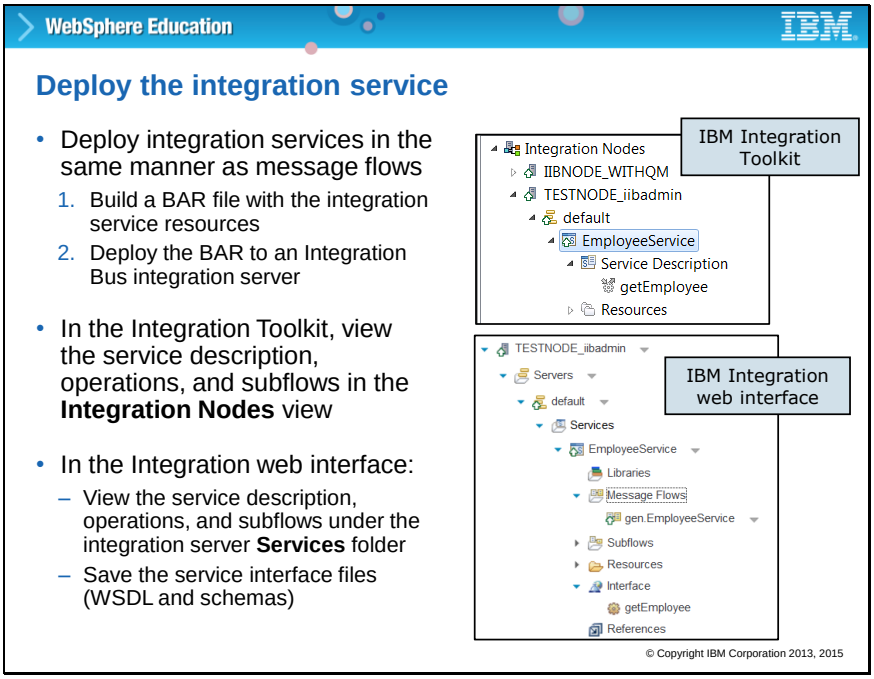
Slide 19



Topic 3: Deploying and testing integration services

In this topic, you learn how to deploy integration services and test them by using the Integration Toolkit Flow exerciser.

Slide 20



Deploy the integration service

An integration service is a first class application on the Integration Bus. To deploy an integration service, follow the same steps as deploying a message flow application.

Start by building a BAR file, with the integration service resources. Then, deploy the archive to an Integration Bus integration server. In the Integration Toolkit, you can deploy the integration service by dragging the integration service from the **Application Development** view to an integration server in the **Integration Nodes** view.

In the **Integration Nodes** view, you can view the service description, operation, and their corresponding subflows.

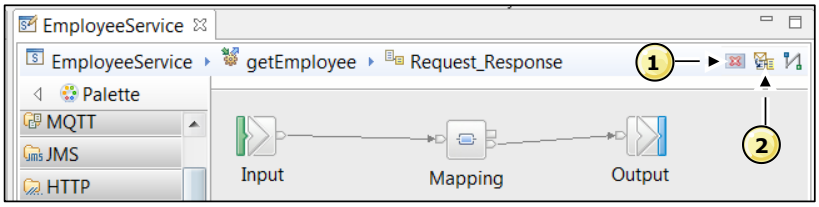
You can use the Integration web user interface to view the service description, operations, and subflows under the integration server **Services** folder. By using the Integration web user interface, you can also save the service interface WSDL and schemas.

Slide 21

WebSphere Education

Testing the integration service (1 of 2)

- To test an integration service, send a test message with the Integration Toolkit Flow exerciser
- Edit the web service input message in the message section.

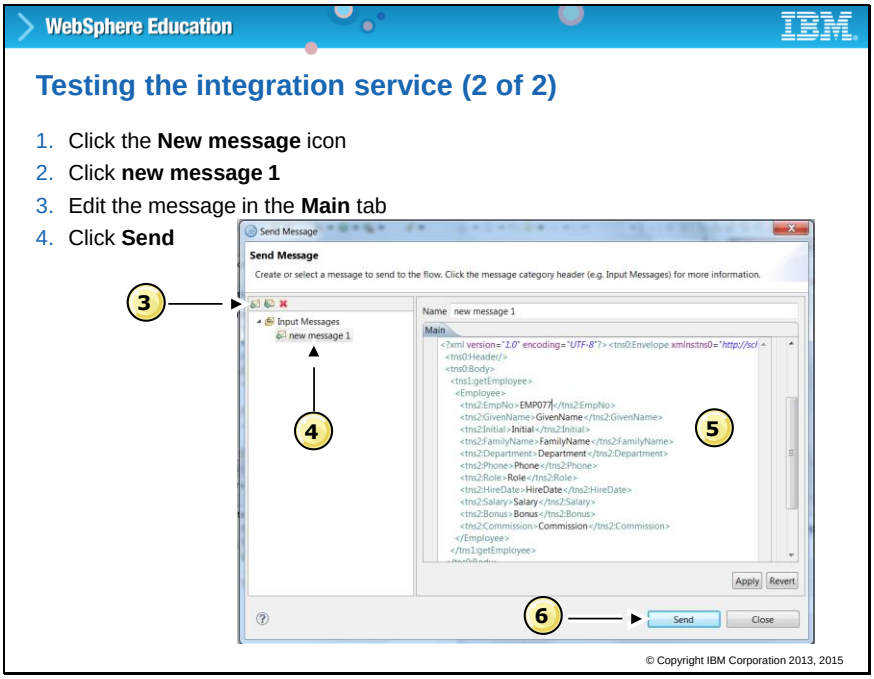


1. Click the **Start the Flow exerciser** icon to deploy the service
2. Click the **Send a message to the flow** icon

Testing the integration service (1 of 2)

You can use the Integration Toolkit Flow Exerciser to test the subflows in an integration service by clicking the **Start the Flow exerciser** icon in the editor.

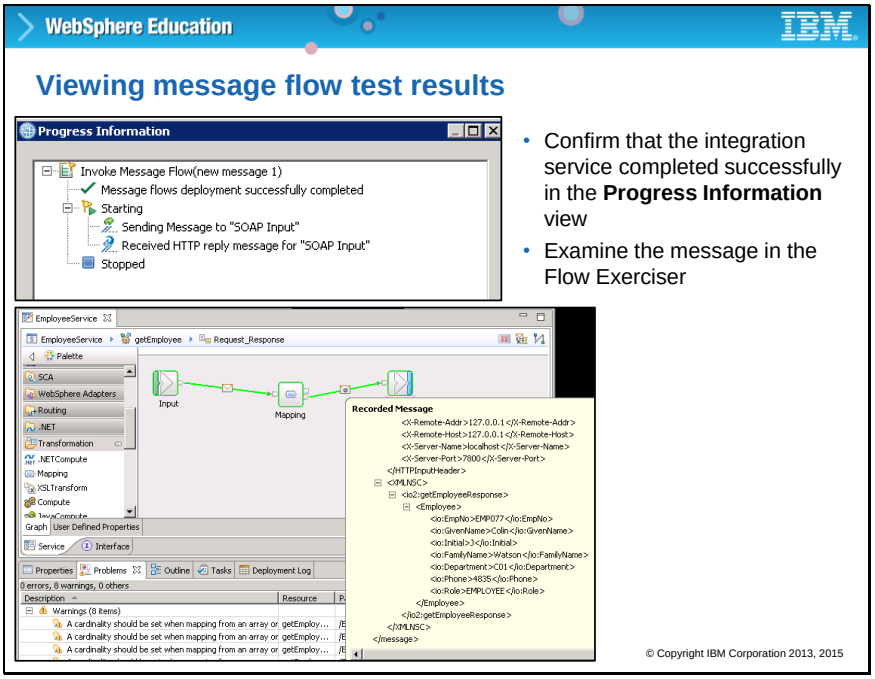
Slide 22



Testing the integration service (2 of 2)

In the Flow exerciser, you can edit the input message as an XML structure or read an input message from a file. Click **Send** to send the message and start the test.

Slide 23



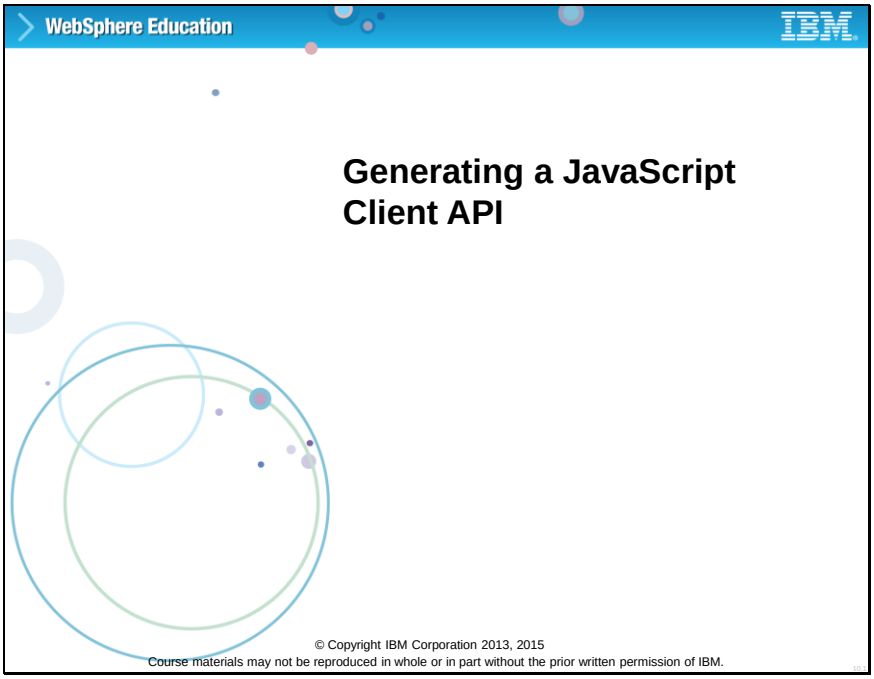
Viewing message flow test results

The Flow exerciser shows the message path. You can also view the logical message at different points in the message flow.

You should already be familiar with using the Flow exerciser.

In the exercise for this unit, you create an integration service and test it by using the Flow exerciser.

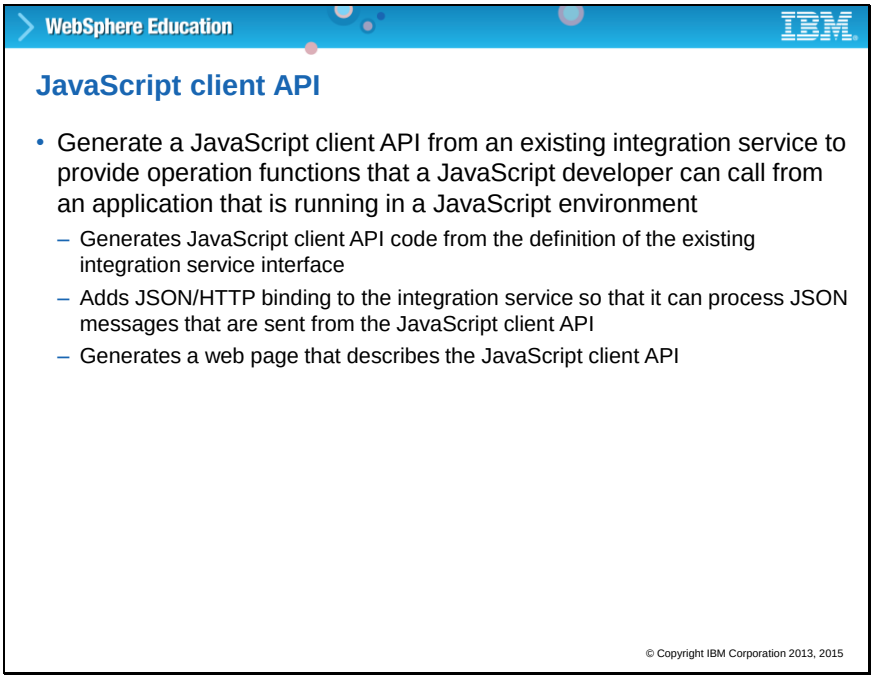
Slide 24



Topic 4: Generating a JavaScript Client API

A new feature in Integration Bus V10 is the ability to generate a JavaScript client API for an integration service. This topic describes this new function.

Slide 25



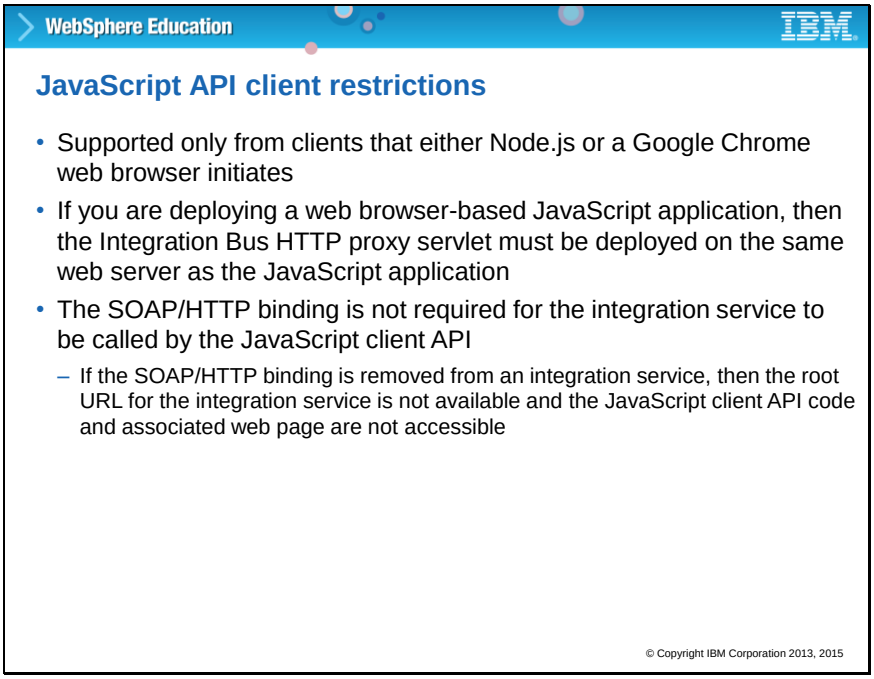
JavaScript client API

In the Integration Toolkit, you can generate JavaScript client API code from the definition of an existing integration service interface. You can then use this code in JavaScript applications to call the integration service operations without configuring the underlying communication mechanism.

A JSON/HTTP binding is added to the integration service so that the integration service can process JSON messages that are sent from the JavaScript client API. The JSON/HTTP binding is intended for use only by the JavaScript client API.

The Integration Toolkit also generates a web page that describes the JavaScript client API. From the web page, the JavaScript developer can download or reference the JavaScript client API code and copy sample code directly into JavaScript client applications.

Slide 26



JavaScript API client restrictions

The JavaScript API client implementation in Integration Bus does have some restrictions.

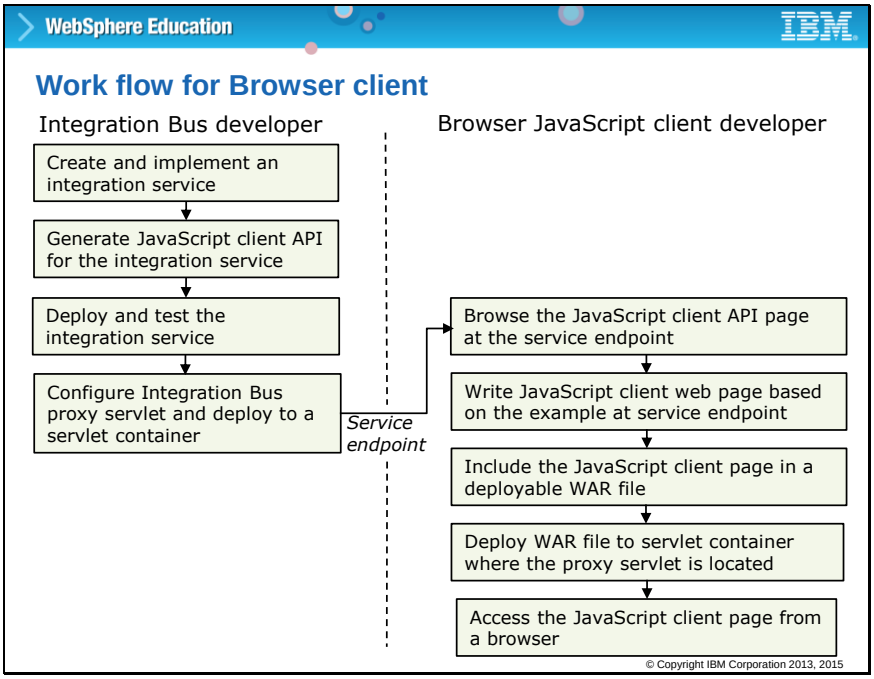
Calling integration services by using the JavaScript client API is supported only from clients that either a Node.js or a Google Chrome web browser initiates.

Node.js is an open source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js run time on OS X, Microsoft Windows, Linux, NonStop, IBM AIX, IBM System z, and IBM i.

If you are deploying a web browser-based JavaScript application, then the Integration Bus HTTP proxy servlet must be deployed on the same web server as the JavaScript application. This implementation also requires an integration node listener and that the integration node is associated with an MQ queue manager.

All integration services in Integration Bus have a SOAP/HTTP binding by default. This binding is not required if the integration service is called by the JavaScript client API but it is required to access the root URL for the integration service, the JavaScript client API code, and the associated web page.

Slide 27



Work flow for Browser client

The work flow for implementing a JavaScript client API is divided between the integration service developer and the JavaScript client developer. This slide summarizes the work flow when the JavaScript client API is generated for use by a web browser-based JavaScript application.

The Integration Bus developer completes the following steps to generate a JavaScript client API from an existing integration service:

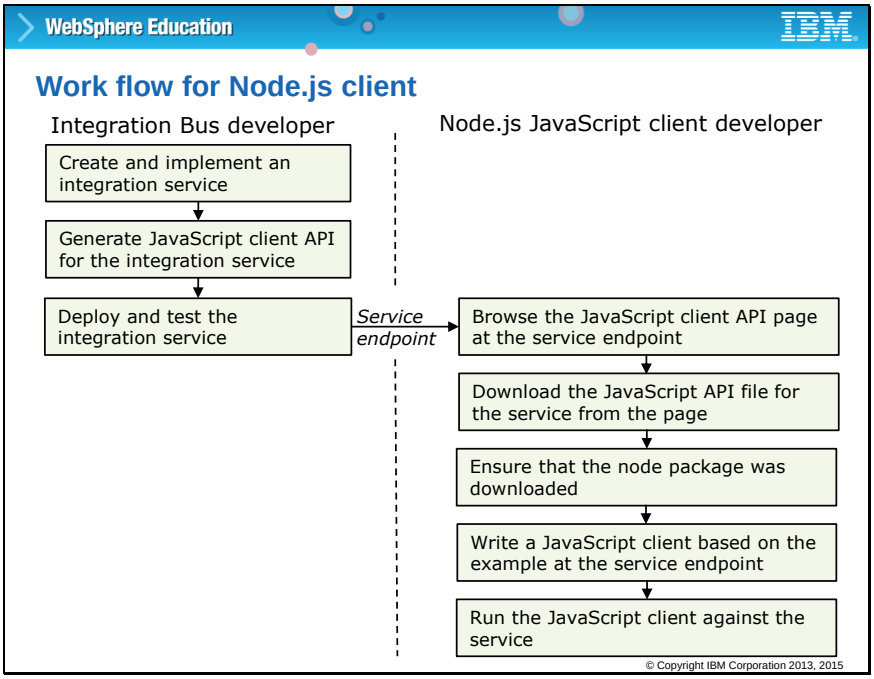
1. Open the integration service in Integration Toolkit.
2. On the **Service** tab of the Integration service editor, right-click the integration service name and then click **Generate > JavaScript Client API**.
3. Deploy and test the integration service to an integration server.
4. To ensure that a web browser-based JavaScript application can receive messages from the integration service, you must deploy the Integration Bus HTTP proxy servlet onto the web application server that hosts the JavaScript application. The IBM Knowledge Center for Integration Bus contains step-by-step instructions for configuring the environment for a web browser-based application that uses the JavaScript client API.

At this point, the developer can provide the JavaScript developer with the integration service endpoint URL for the running service so that the JavaScript developer can download or reference the JavaScript client API files and sample code.

The JavaScript developer can then:

- Browse the JavaScript client API page at the service endpoint to get the sample code.
- Create the JavaScript client web page based on the sample code.
- Include the JavaScript client page in a deployable WAR file.
- Deploy the WAR file to servlet container where the proxy servlet is located.
- Access the JavaScript client page from a browser.

Slide 28



Work flow for Node.js client

This slide summarizes the work flow when the JavaScript client API is generated for use by a node.js application.

The integration service developer completes the following steps to generate a JavaScript client API from an existing integration service:

1. Open the integration service in the Integration Toolkit.
2. On the **Service** tab of the integration service editor, right-click the integration service name and then click **Generate > JavaScript Client API**.
3. Deploy and test the integration service to an integration server.

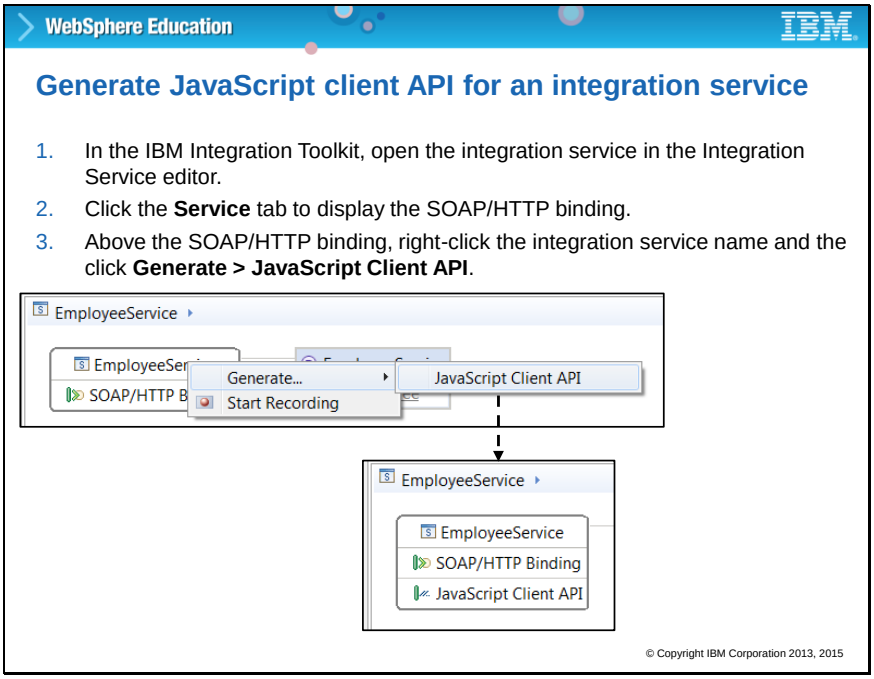
Now, the Integration service developer can provide the JavaScript developer with the integration service endpoint URL for the running service so that the JavaScript developer can download or reference the JavaScript client API files and sample code.

The JavaScript developer can then:

- Browse the JavaScript client API page at the service endpoint URL.
- Download the JavaScript API file for the service from the page.
- Ensure that the node package was downloaded.

- Write a JavaScript client based on the example at the service endpoint.
- Run the JavaScript client against the service.

Slide 29



Generate JavaScript client API for an integration service

This slide reviews the steps for generating a JavaScript client API for integration service:

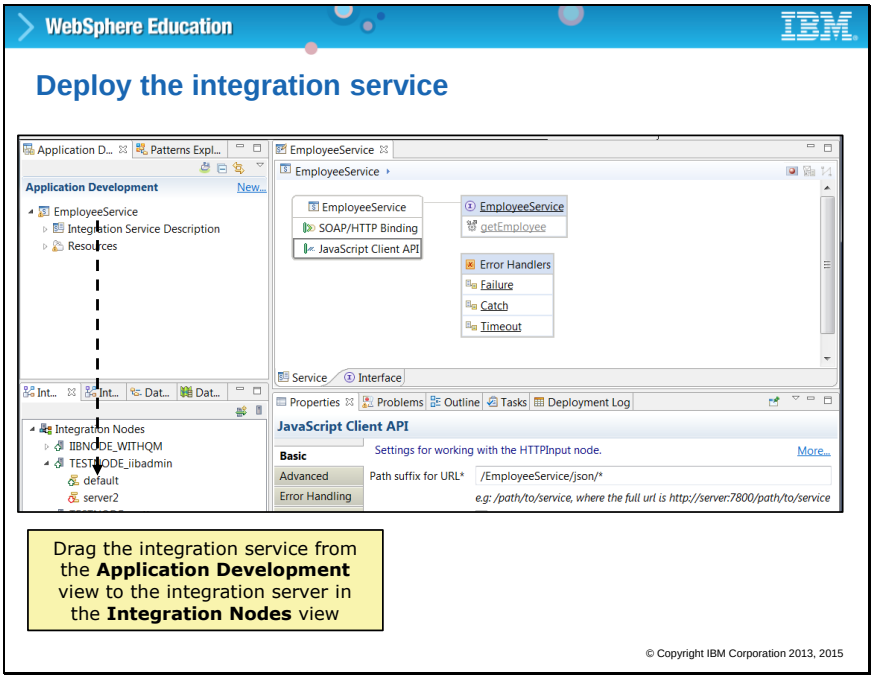
1. In the Integration Toolkit, open your integration service in the Integration service editor by double-clicking the **Integration Service Description** folder in the **Application Development** view.

2. Click the **Service** tab. The integration service description is displayed, which includes the SOAP/HTTP binding.

3. Above the SOAP/HTTP binding, right-click the integration service name and click **Generate > JavaScript Client API**. The JavaScript client API for the integration service is generated and a reference to the JavaScript client API is displayed below the SOAP/HTTP binding.

In normal conditions, you should not need to regenerate the JavaScript client API. When you add new operations to the integration service, or update or delete existing operations, the API is automatically updated when you save the integration service. However, if you want to manually regenerate the JavaScript client API, you must remove the existing API first by right-clicking the JavaScript Client API entry in the Integration Toolkit integration service editor and then clicking **Remove**.

Slide 30

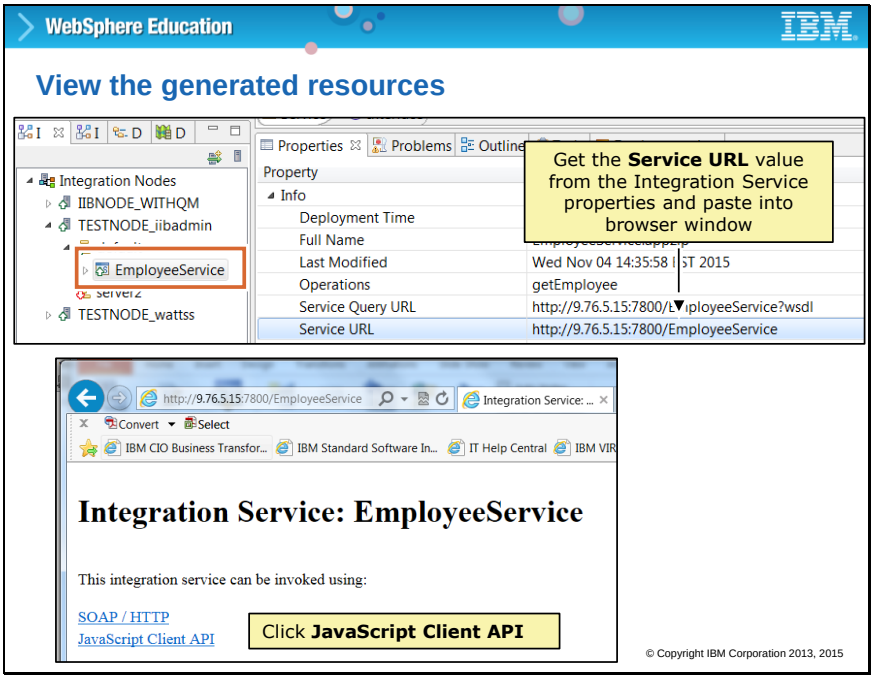


Deploy the integration service

After you generate the JavaScript client API, deploy the integration service so that you can access the Service URL.

In the Integration Toolkit, you can drag the integration service from the **Application Development** view to the integration server in the **Integration Nodes** view.

Slide 31

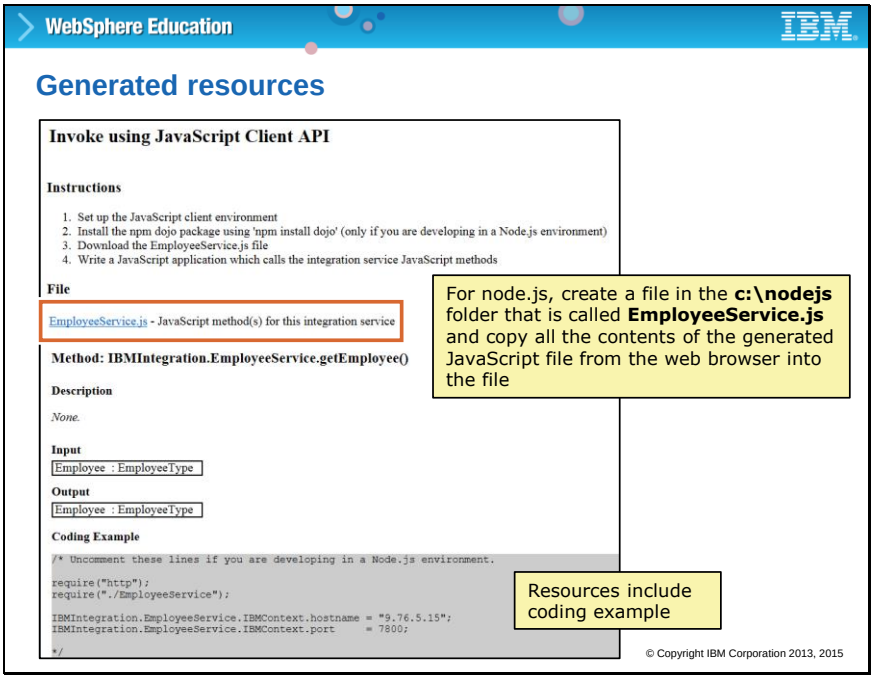


View the generated resources

The service endpoint URL is available on the integration service properties at run time. In the Integration Toolkit, select the integration service in the **Integration Nodes** view and then display the **Properties** view.

To view the generated resources, copy the Service URL from the integration service properties and then paste it in a browser. In the web page, click the **JavaScript Client API** link to open the resources page.

Slide 32

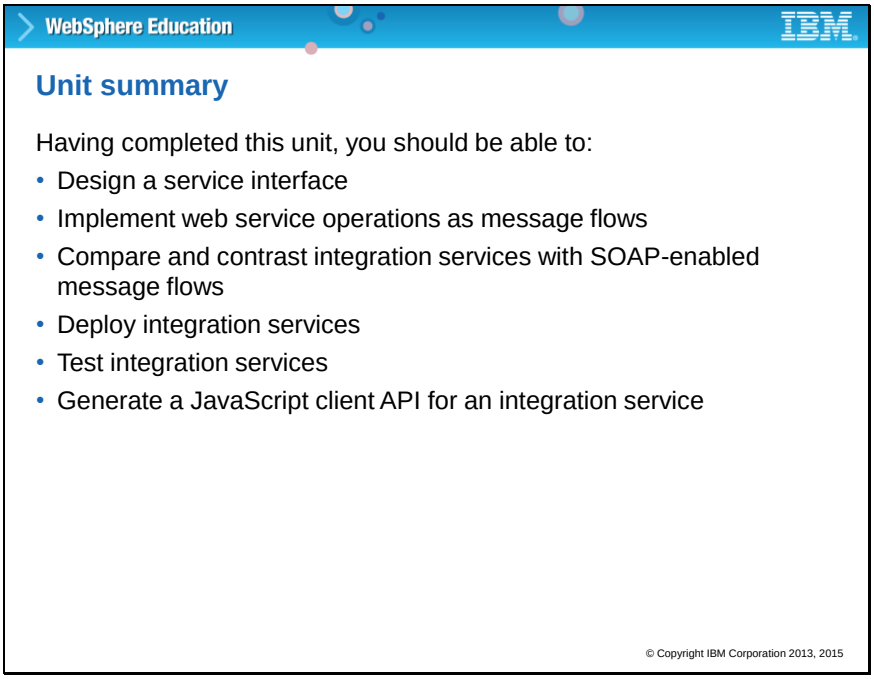


Generated resources

The JavaScript Client API web page describes the JavaScript client API that is generated for the service. From the web page, the JavaScript developer can download or reference the JavaScript client API code and copy sample code directly into JavaScript applications.

This slide is an example of the generated service for the EmployeeService.

Slide 33



Unit summary

An integration service is a specialized application with a defined interface and structure that acts as a container for a web services solution. In this unit, you learned how to create and implement IBM Integration Bus integration services.

Having completed this unit, you should be able to:

- Design a service interface
- Implement web service operations as message flows
- Compare and contrast integration services with SOAP-enabled message flows
- Deploy integration services
- Test integration services
- Generate a JavaScript client API for an integration service