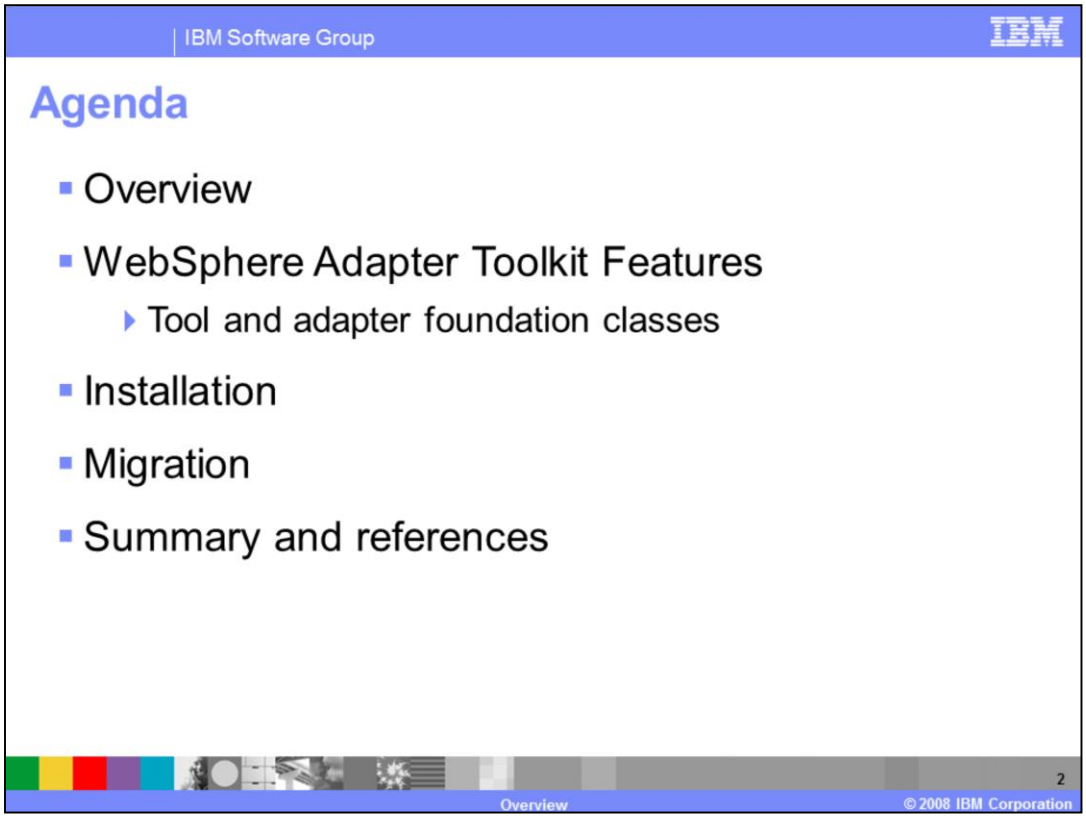
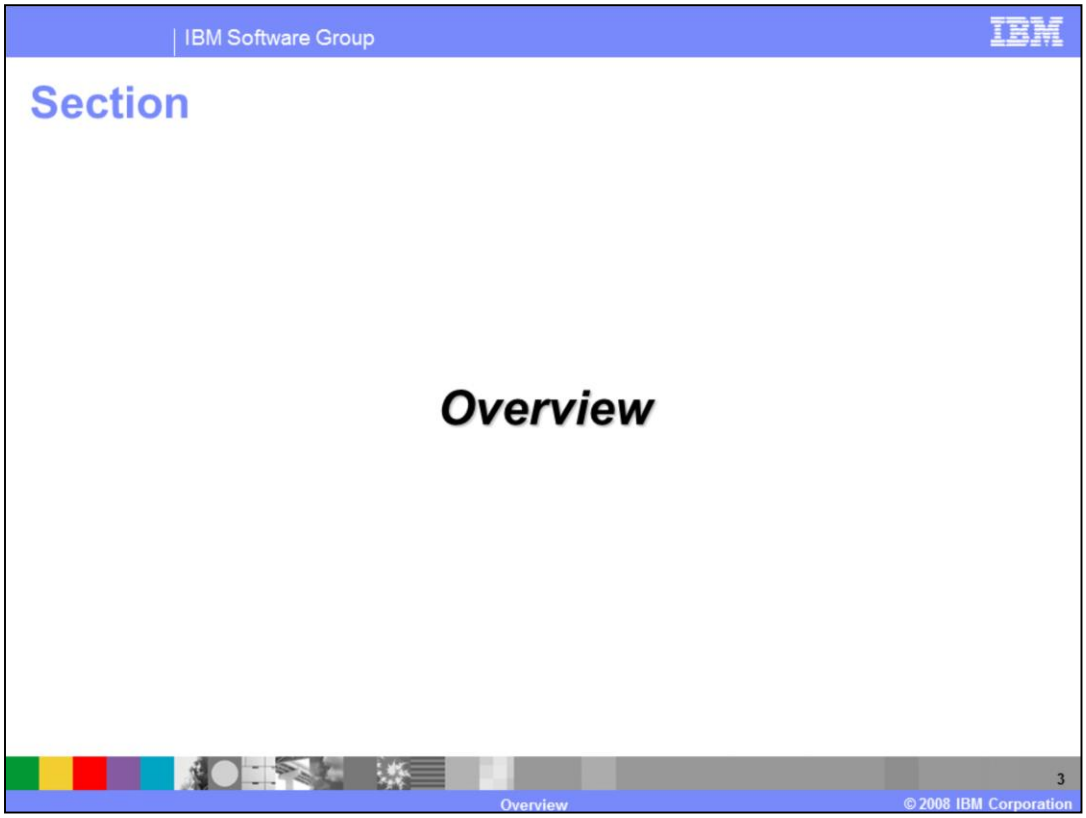
This presentation covers the high level details on how to use the Adapter Toolkit to develop custom WebSphere® JCA adapters. The goal of the presentation is to provide a brief overview of the adapter toolkit and foundation classes and how they can be used in developing a custom WebSphere JCA adapter.

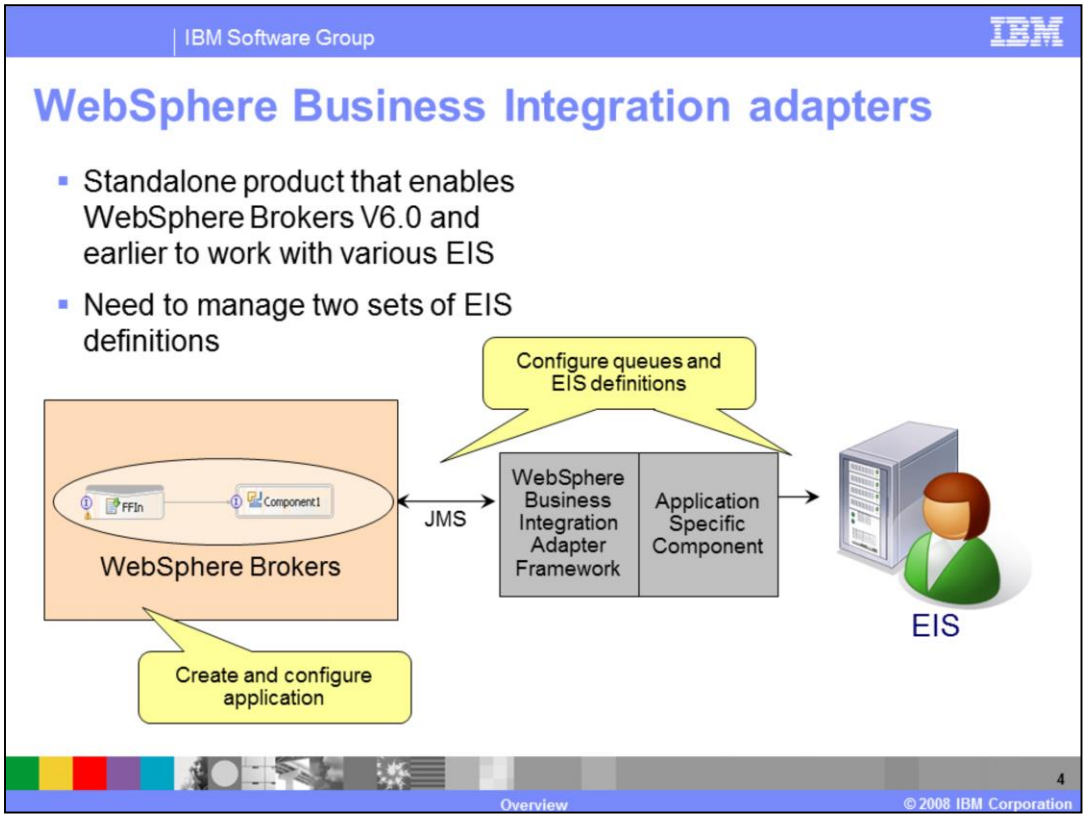


The page shows the agenda of this presentation. The presentation starts with an overview of a WebSphere Adapter Toolkit, introduces the features provided by the toolkit which include the wizard and base class support.

The various base classes that need to be extended and methods that need to be implemented for inbound, outbound and meta data discovery are also presented along with topics like logging, tracing support, installation and migration.

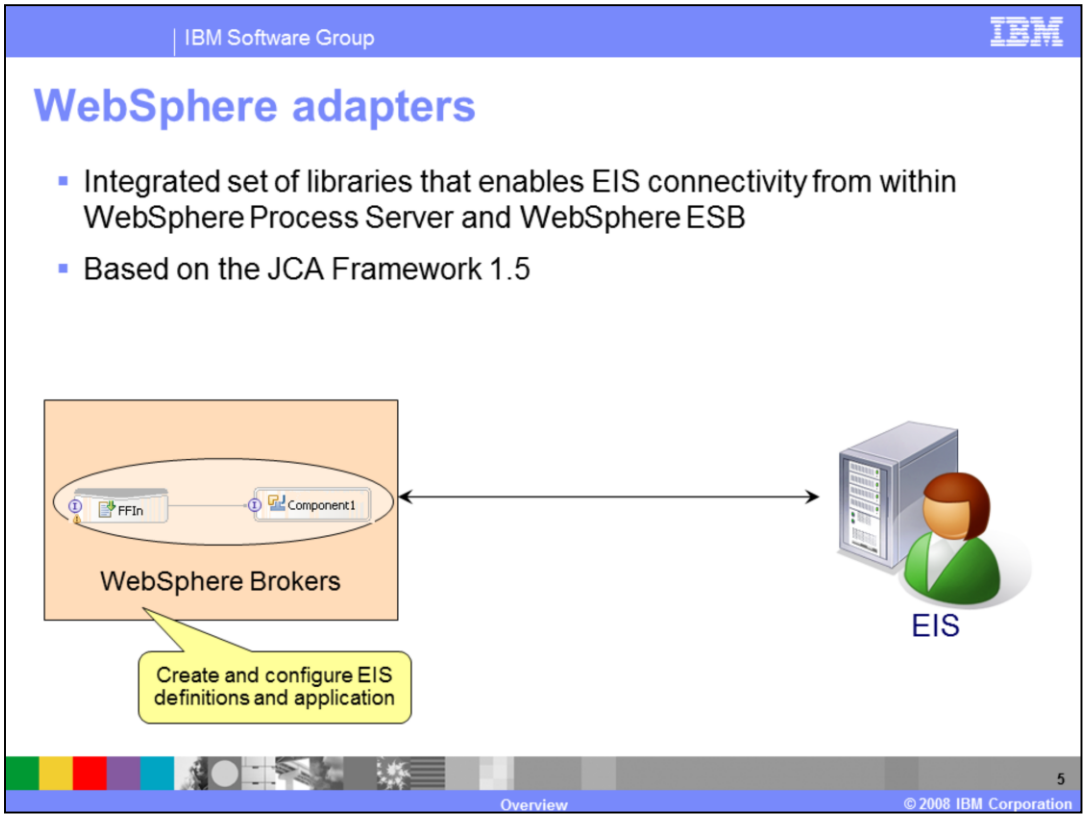


The next section provides a brief overview of the adapters provided by IBM.



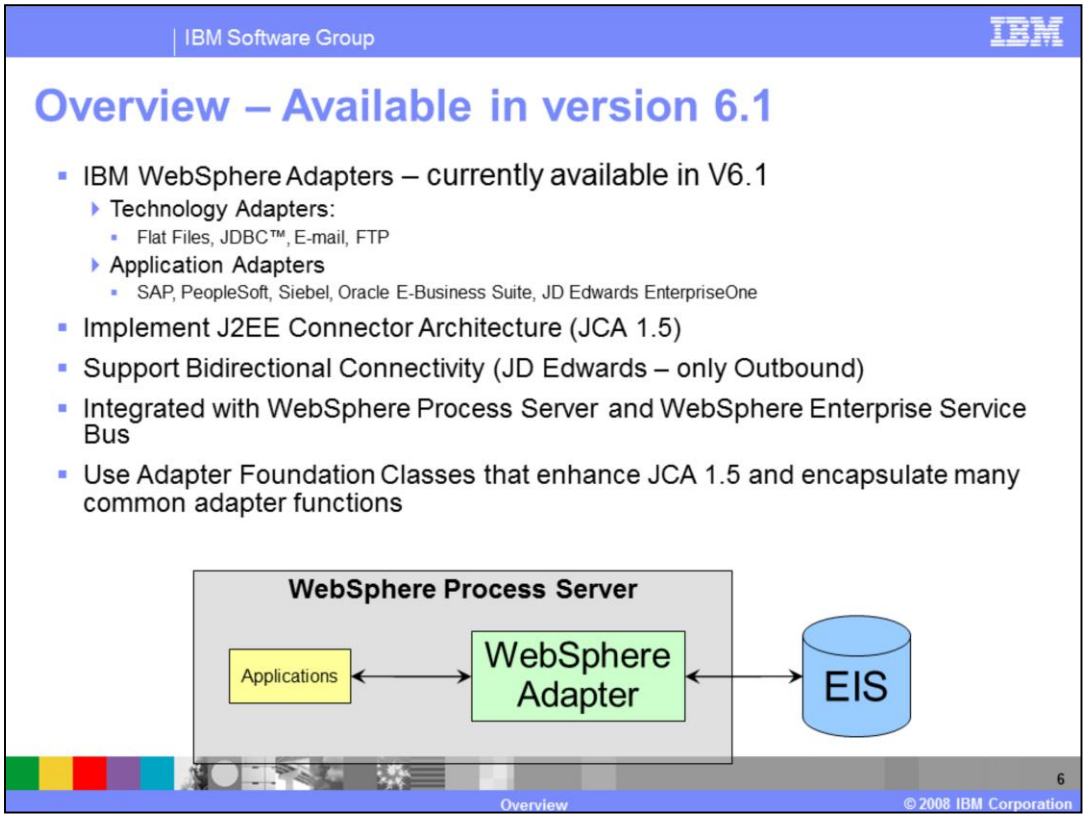
IBM WebSphere Business Integration adapters are a suite of ready-to-use adapters for interfacing to enterprise information systems, or EISs. These adapters are based on a common framework. This allows you to configure and operate the adapters in a consistent manner, while using them in different deployment configurations within the WebSphere software platform to meet your business integration needs.

These legacy adapters are not integrated directly into the broker environment. They are stand-alone adapters running outside of the broker runtime. You use the integrated development environment to design and develop your application and use the JMS binding to communicate with the stand-alone WebSphere Business Integration adapters. This means that you have to configure and manage two integration technologies.

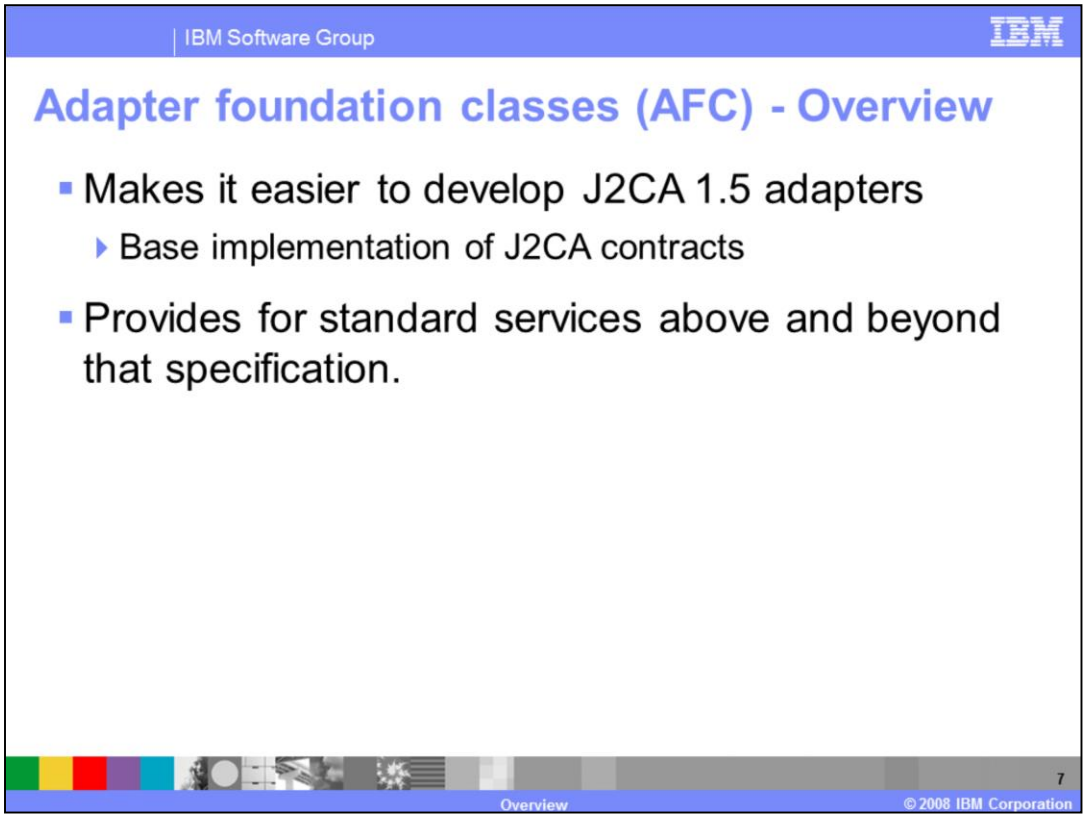


WebSphere adapters are enhanced to make development, deployment, and integration of adapters simple. Enhancements include the ability to discover EIS objects and services, ensure “assured once” inbound delivery of events and other capabilities that are discussed in this presentation. The adapter uses common foundation classes across each of the adapters. These classes enhance the JCA 1.5 functions and encapsulate many common adapter functions to simplify development tasks.

Adapters are exposed as EIS import or export components and can be configured directly using the WebSphere Integration Developer. The adapter components are used in your application which is deployed to the runtime and can be administered directly using the broker tools.



Here is the list of adapters that are available for version 6.1. They include the Flat File, FTP, E-mail and JDBC technology adapters along with the SAP, PeopleSoft, Siebel, JD Edwards EnterpriseOne and Oracle E-Businesses application adapters. These are all based on a set of foundation classes that enhance the JCA 1.5 specification.

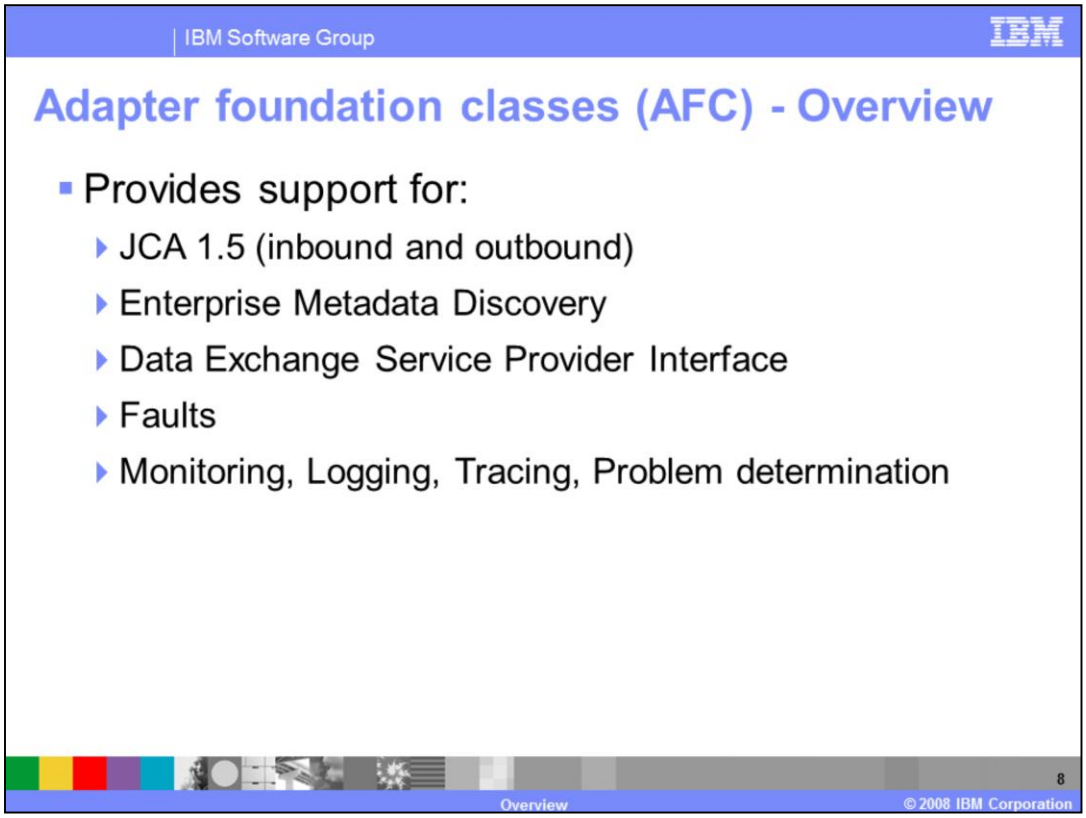


The goal of the base classes is to establish a standard for building resource adapters that conform to the Java 2 Connector Architecture 1.5 specification.

J2EE Connector Architecture defines a series of contracts that must be provided by a resource adapter. So anyone can develop a resource adapter without need of additional tools or support beyond what is provided by the J2EE SDK. However, repeatedly writing resource adapters from scratch is obviously not an efficient approach as implementation of many of the contracts defined by the connector architecture are similar regardless of the underlying enterprise information system (EIS). So developing resource adapters can be made more effective (and consistent) by identifying those areas of the JCA contracts that are generic and then providing a means for developers to take advantage of this common logic.

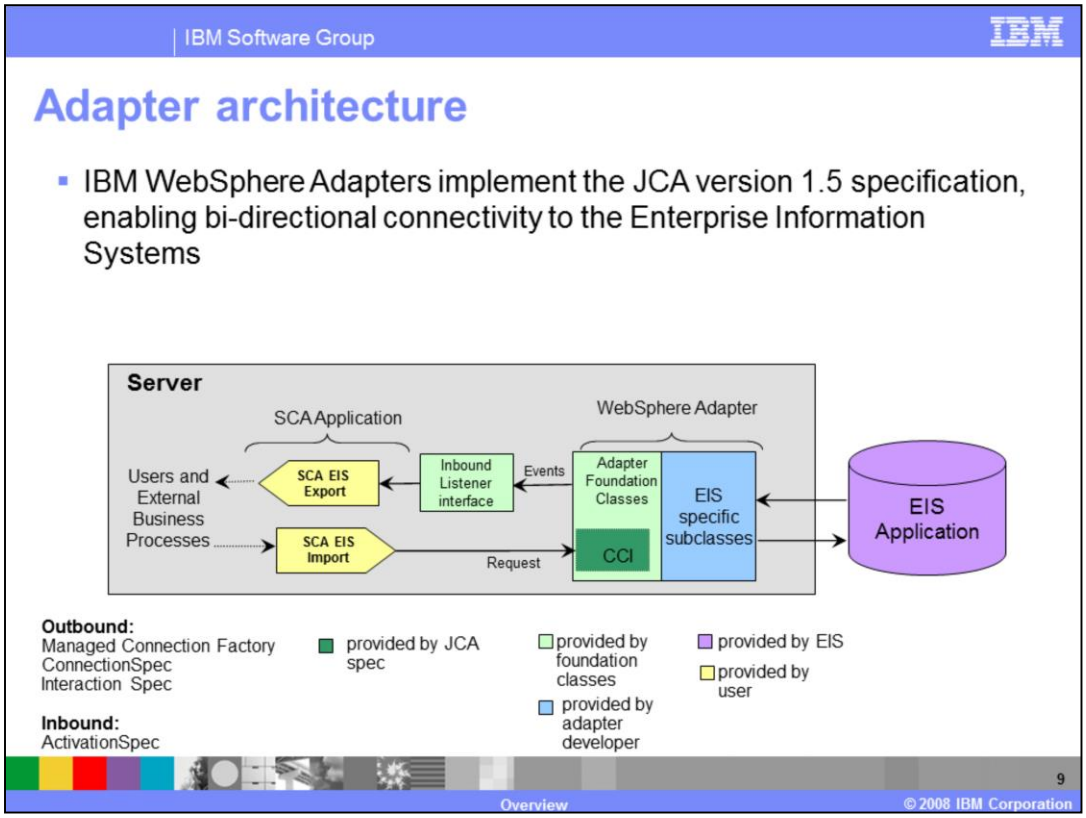
The chosen approach was to provide a set of abstract “base” classes that can be extended by developers. This set of base classes provides implementations of those contracts and methods which can be done generically and so can they can be re-used by more than one adapter. Any methods for which EIS-specific logic is required are left abstract and the individual adapter developer needs to provide implementation for those methods.

For most of the resource adapters, developers can implement the abstract methods and have a working resource adapter. For those resource adapter designs requiring functionality that is different or beyond that which is provided by the base implementation, developers are left free to override the methods of interest in the connector specification.



The base classes or the adapter foundation classes provide support for Inbound, Outbound and Enterprise Metadata discovery.

The JCA 1.5 specification is designed around EISs that provide 'push' models for event delivery where EIS calls the adapter when event is available and provides little support for polling models. To bridge this gap base classes provide pre-built event management logic that can be used by adapter developers. Outbound support involves sending service request to the EIS. Enterprise Metadata Discovery or a discovery service is a component within an adapter that enables the generation of business object definition and other artifacts required by SCA. Adapter foundation classes also provide a set of base faults, data exchange implementations for SDO, Java Bean, and much of the monitoring and problem determination capabilities that adapters need.

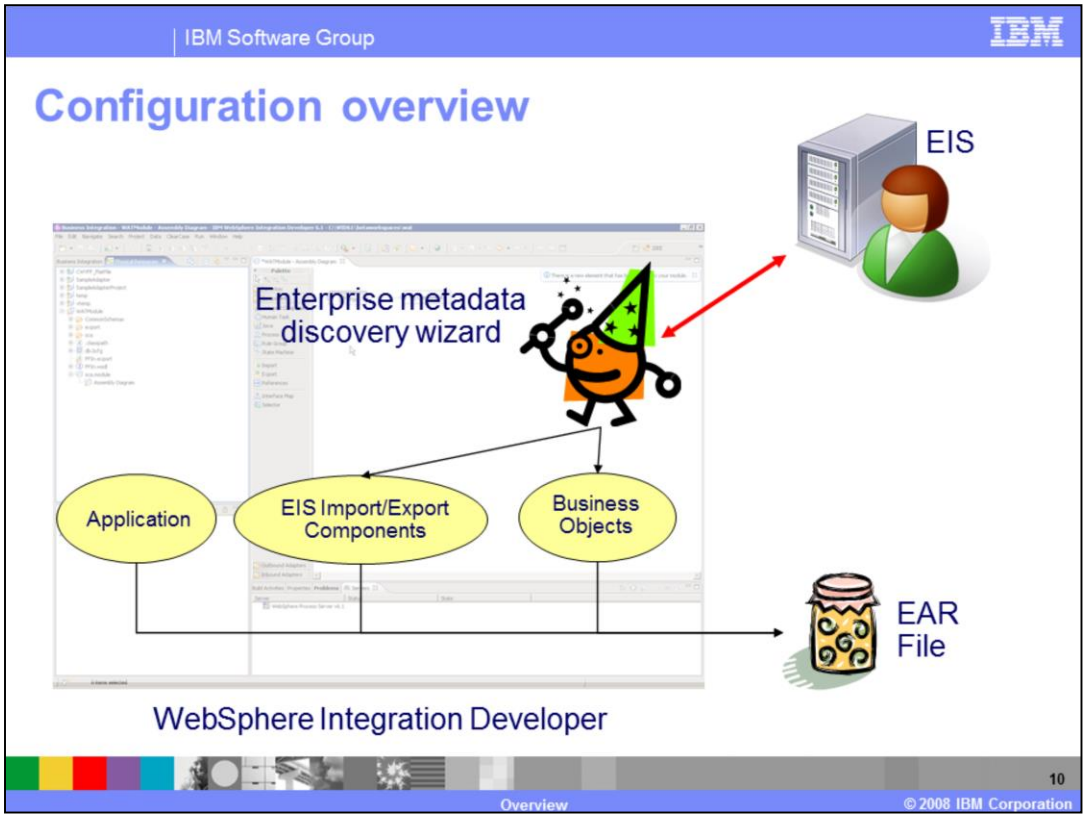


The diagram shows the high-level architecture of different components that play a role in the end to end invocation of the outbound or the inbound request. The Enterprise Service Discovery tool in the WebSphere Integration Developer is used to create the SCA EIS components and the associated Business Objects. For the outbound request, the SCA EIS Import component is created, and for the inbound request, the SCA EIS Export component is created.

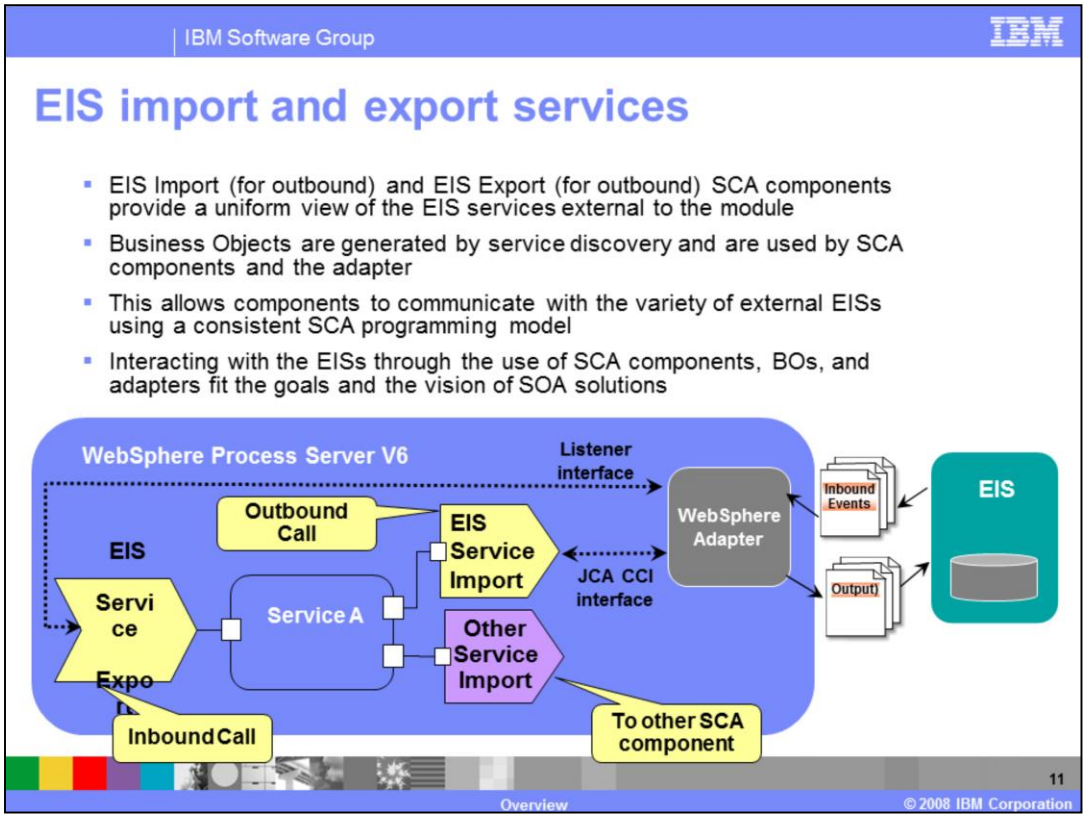
The SCA Clients interact with the SCA EIS Export and Import components to drive an outbound request or receive an inbound request, as shown in the diagram. The Adapter contains the implementation of JCA specifications and has extensions provided by the Adapter foundation classes. The SCA Import component passes a Business Object wrapped in a J2C CCI Record object. The Adapter extracts the Business Object from the Record object and determines the appropriate function to call and passes along the required arguments.

The two main interfaces to a JCA adapter are the SPI (Service Provider Interface) and the CCI (Common Client Interface). The SPI is the application server's view of the adapter. It contains the contracts necessary to work well with an application server; connection creation and matching, security, work management, and so on. The CCI is designed to provide a common view of data and interaction with the adapter. The CCI defines the data

model and provides a common mechanism to interact with the adapter.



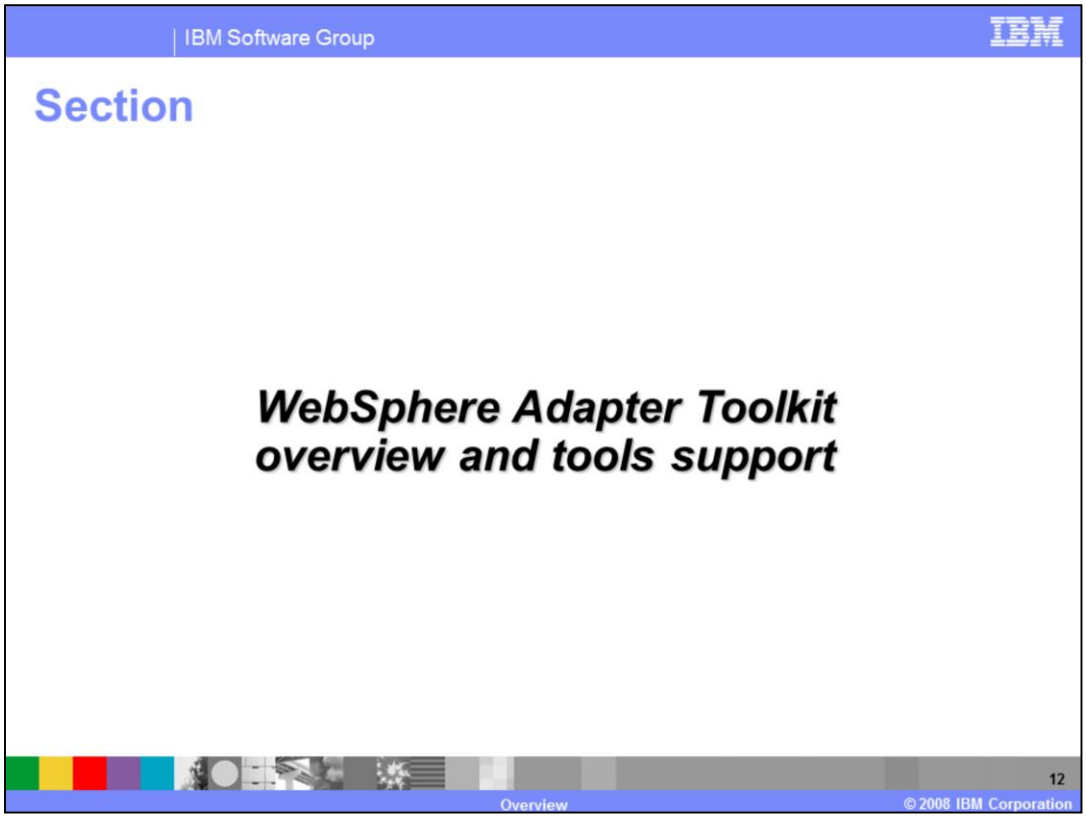
This slide depicts the steps involved in configuring the adapters and creating a deployable application. WebSphere Integration Developer provides a wizard that connects to the EIS, introspects and imports the metadata information and creates the necessary artifacts for you to build your application. The artifacts generated by the wizard are the import or export components, business objects and WSDL files. The import or export component created is different based on the interaction style you choose when configuring using the wizard. The supported interaction styles are inbound and outbound. Once the artifacts are created for you, you can use these generated artifacts to configure the EIS export or import components and include them in the SCA application based on your application logic. The finished application project along with your resource archive project representing the adapter becomes the components of the EAR file that you create to deploy to the runtime.



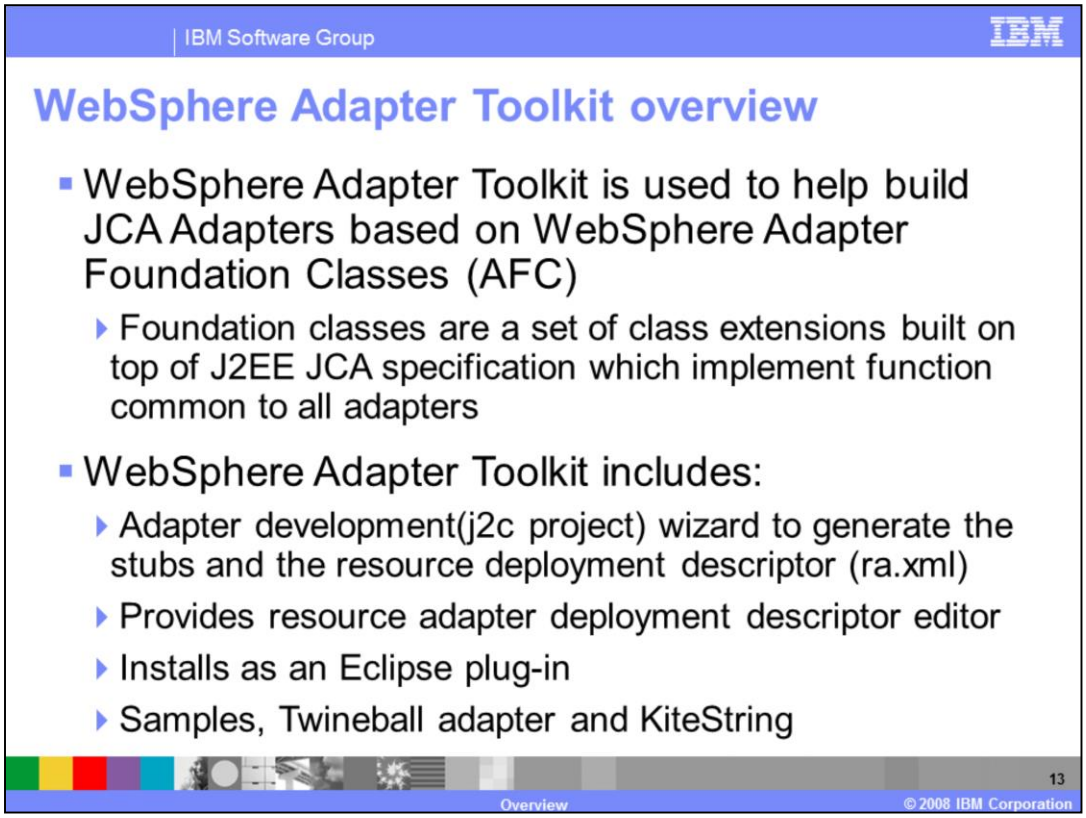
The Enterprise Service Discovery Tool in WebSphere Integration Developer creates an EIS Import SCA component for an outbound request, and creates EIS Export SCA component for an inbound event request. The Business Objects for the outbound or inbound requests are also created. Using the SCA components for the adapter, they can be wired with other SCA components to create a business application.

In the diagram, the SCA component representing “Service A” is wired with the Adapter EIS Export and EIS Import component. Also shown is the wiring from Service A to other SCA components through the Import. The implementation of Service A can be BPEL or any other support implementation like Java, Human task and so on.

For SCA clients, the adapter functionality is exposed through the EIS Import and EIS Export SCA components.



The next section provides a brief overview of the adapter tool kit and the wizards provided.

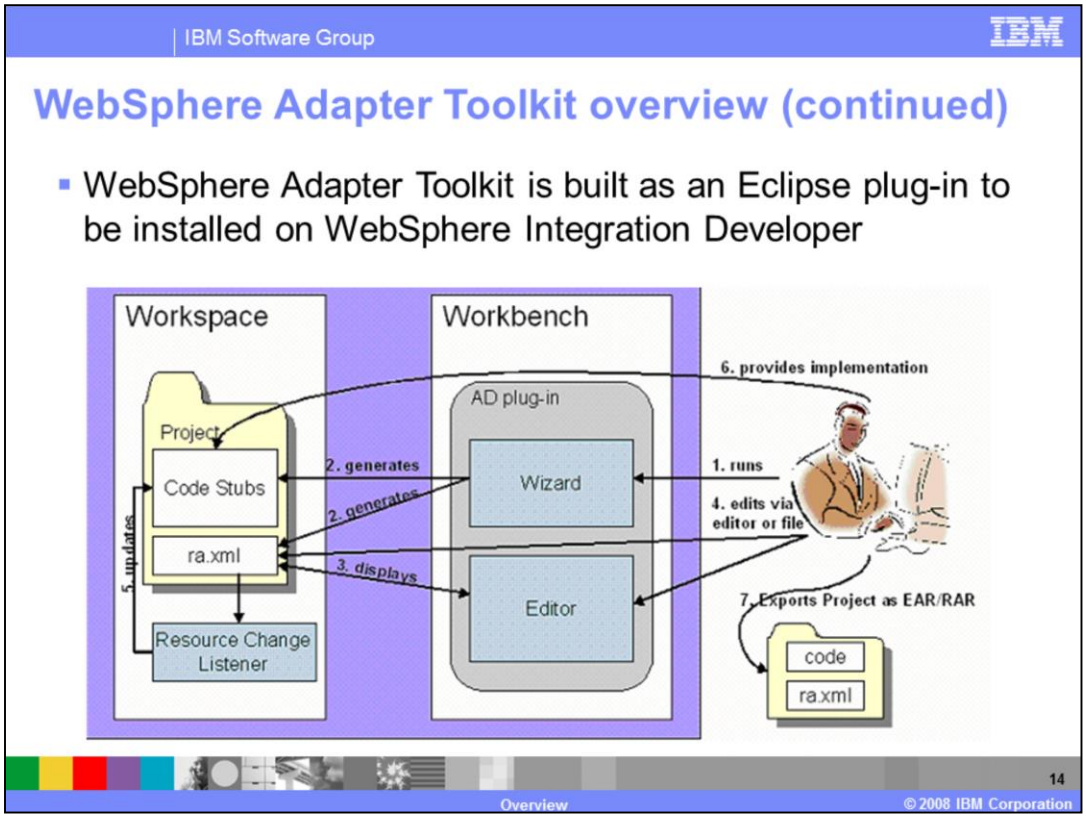


WebSphere Adapter Toolkit is used to help develop custom WebSphere JCA adapters based on foundation classes. The foundation classes make up a set of extensions that implement functionality that is common to all adapters; these classes are built on top of the J2EE JCA 1.5 specification.

WebSphere Adapter Toolkit provides an adapter development wizard which can be used to generate stubs and the deployment descriptor file which is the ra.xml file.

The tool provides deployment descriptor editor which can be used to edit and view the deployment descriptor

WebSphere Adapter Toolkit installs on WebSphere Integration Developer as an eclipse 3.0 plug-in and the installation includes a sample Twineball and KiteString applications which can be used as reference when developing your own custom adapters.

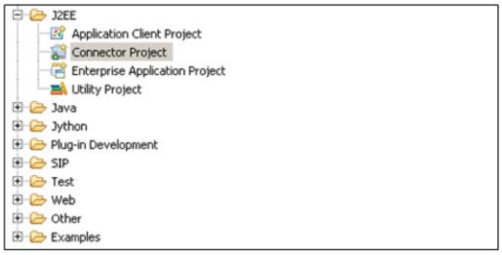


The picture shows how an adapter developer interacts with WebSphere Adapter Toolkit.

You use the wizard to generate the code stubs and the ra.xml. The deployment descriptor editor can be used to view or modify the ra.xml file. Any changes that are made to the ra.xml using the editor, like adding configuration properties, will result in the get and set methods added to the appropriate code stubs. You have to provide implementation for the code stubs and the project can be exported as RAR or EAR file for deployment.

Tool – J2C project wizard

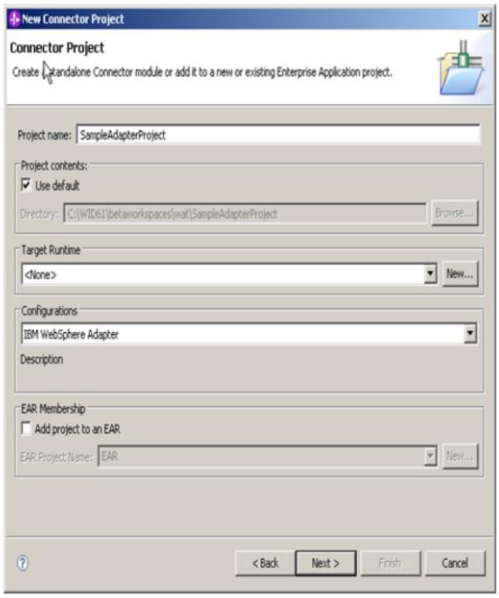
- WebSphere Adapter Toolkit provides a wizard that generates code stubs for the inbound, outbound, data binding and EMD
  - File → New → Project
    - Select J2EE → Connector Project



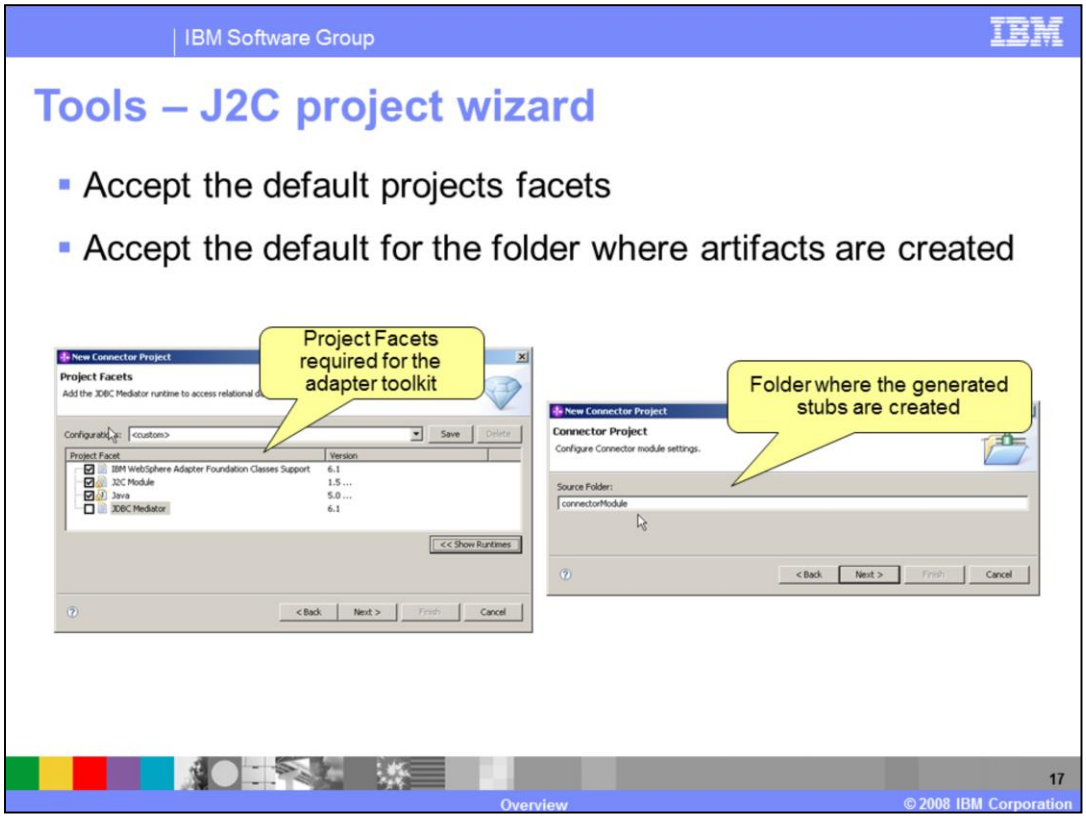
WebSphere Adapter Toolkit provides a wizard which can used to generate the code stubs for the inbound, outbound and Enterprise Metadata Discovery (EMD)

Tools – J2C project wizard

- Provide project name
- Select target runtime as None
- Select Configuration as "IBM WebSphere Adapter"



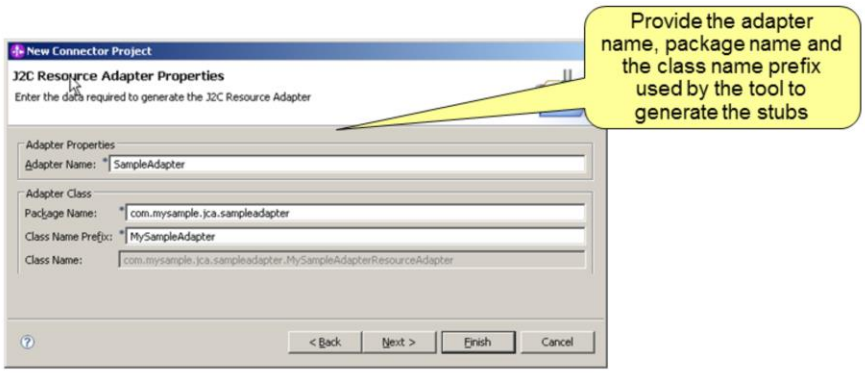
This slide shows a screen capture of the project configuration panel in J2C wizard. Provide a project name, select target runtime as None and set configurations to IBM WebSphere Adapter.



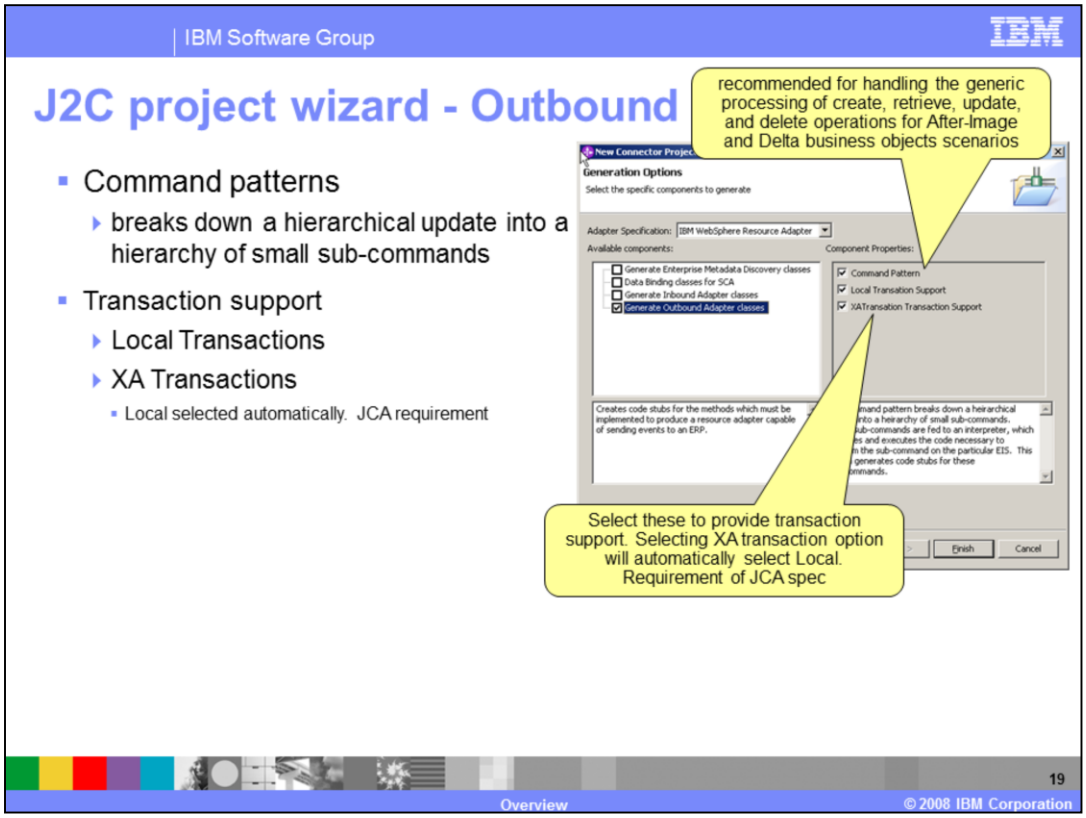
In the project facets panel, accept the defaults. In the next panel where you are prompted for the source folder for the generated code stubs, accept the default value.

Tools – J2C project wizard

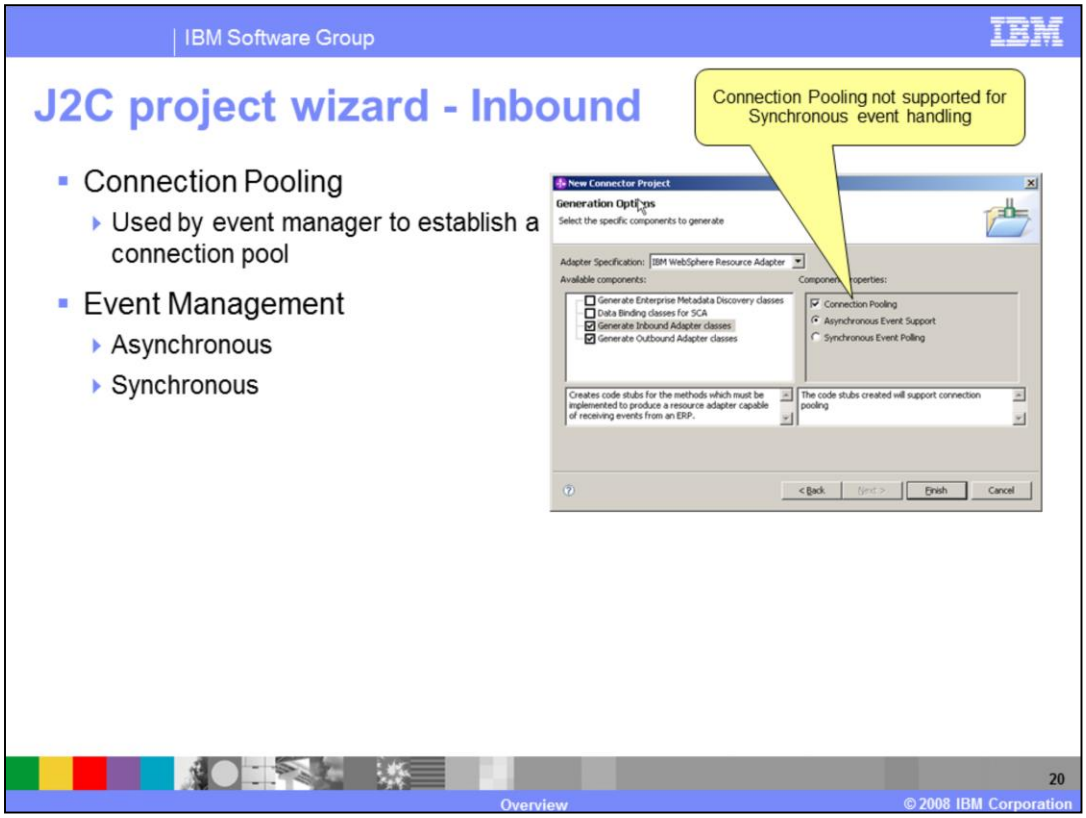
- Resource adapter project properties
  - Provide adapter name, package name and prefix for the generated classes names



The next panel provides you the option to configure the resource adapter properties. In this panel you will provide the name for your custom adapter, the package name to be used for the generated code stubs and the prefix to be used for the classes generated.



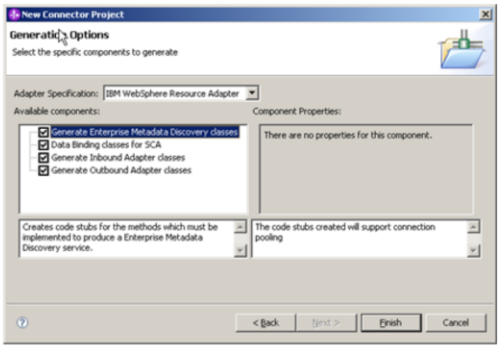
In the generation options panel, you have the option to choose the functionality that you want to support for your adapter. The supported interaction styles are inbound and outbound. For outbound, you have the option to choose command pattern and transaction support. Command patterns support breaks down a hierarchical update into a hierarchy of small sub-commands. The code stubs for these sub commands are created for you. You can provide your implementation logic in these code stubs. More information on command patterns is discussed in the details presentation.



For inbound, you can select options like connection pooling, synchronous or asynchronous event support. The options you select here determine the code stubs generated for you by the wizard.

J2C project wizard – EMD, data binding

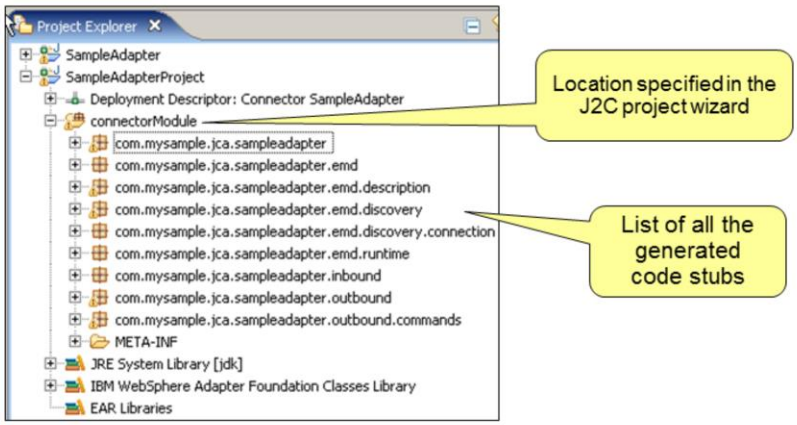
- Enterprise Metadata Discovery
  - Provides the functionality to introspect the EIS
  - Generate business objects and other service component architecture (SCA) artifacts
- Data binding
  - Convert from an SDO to a Record and vice-versa



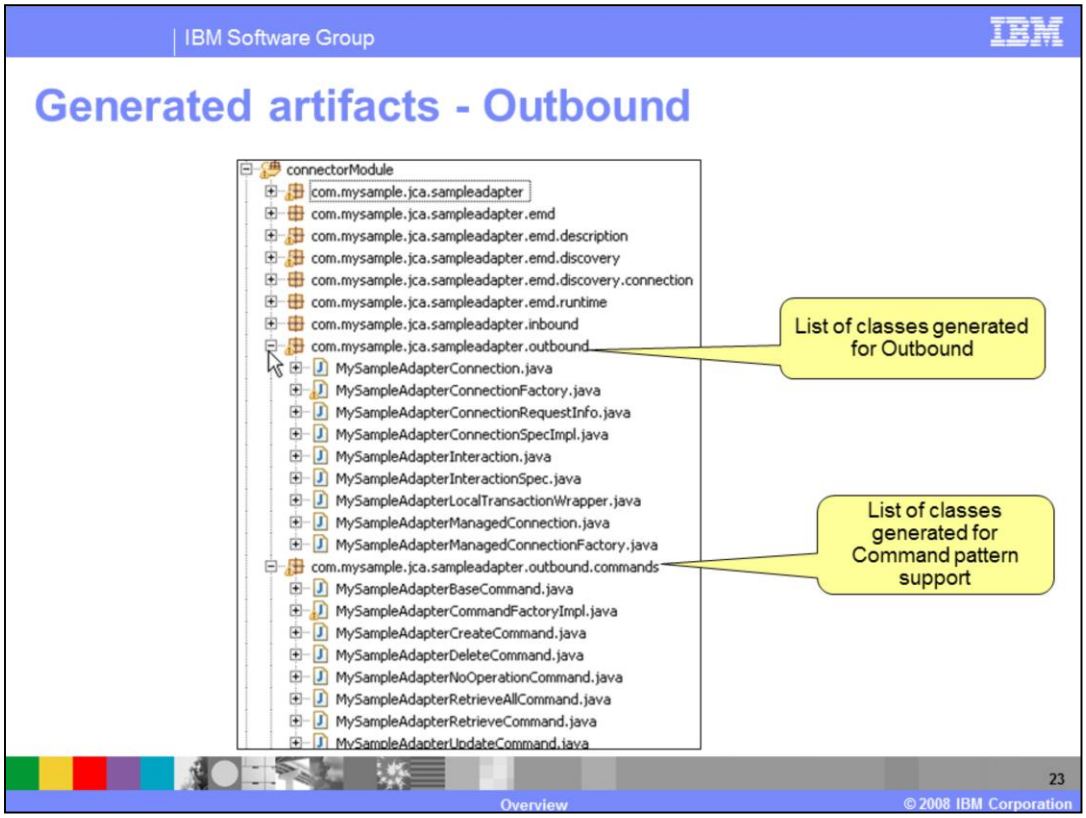
Data bindings convert from an SDO to a record and vice-versa. You will need an adapter-specific data binding and data binding generator.

Most of the data binding generator code is provided in the adapter foundation classes.

J2C project wizard – Generated artifacts

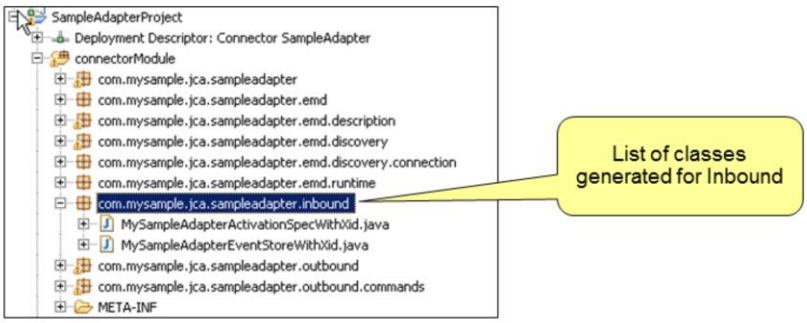


Based on the selection you make in the wizard, the generated artifacts can be different. This slide shows a screen capture of the generated artifacts when you select all the options available.

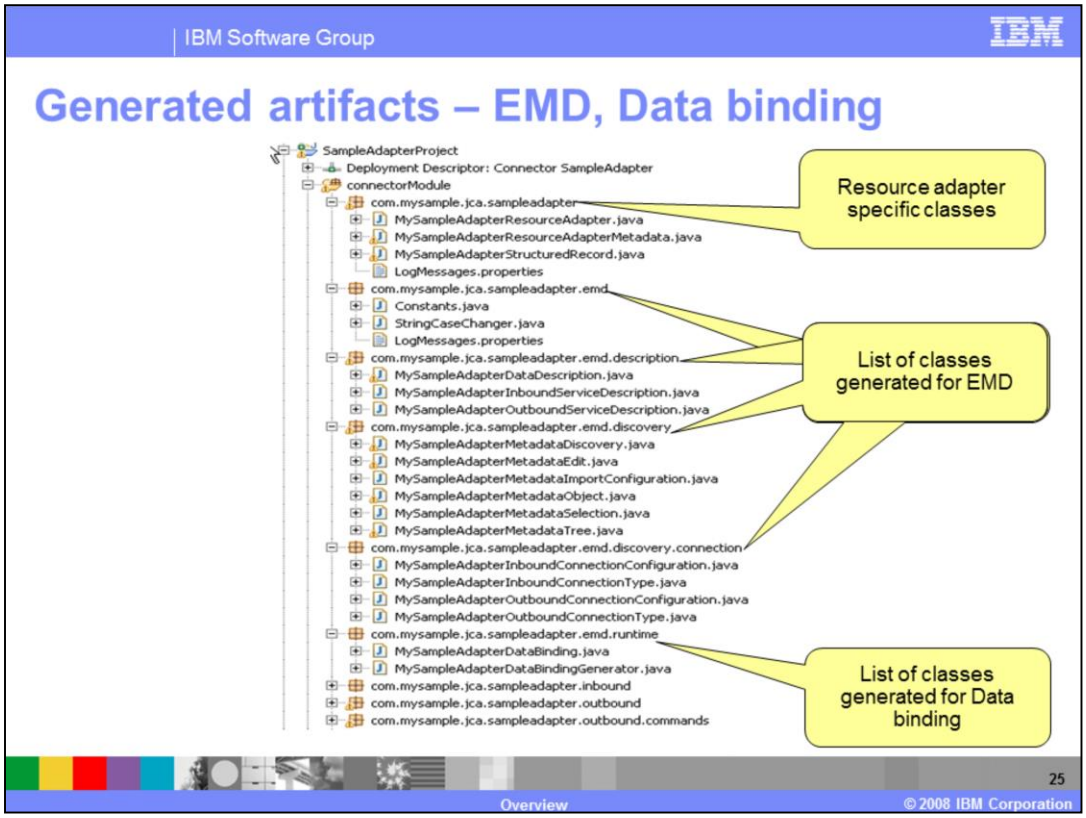


This slide lists the classes that are generated for the outbound support. More information on what you need to implement in each of these classes in provided in the details presentation.

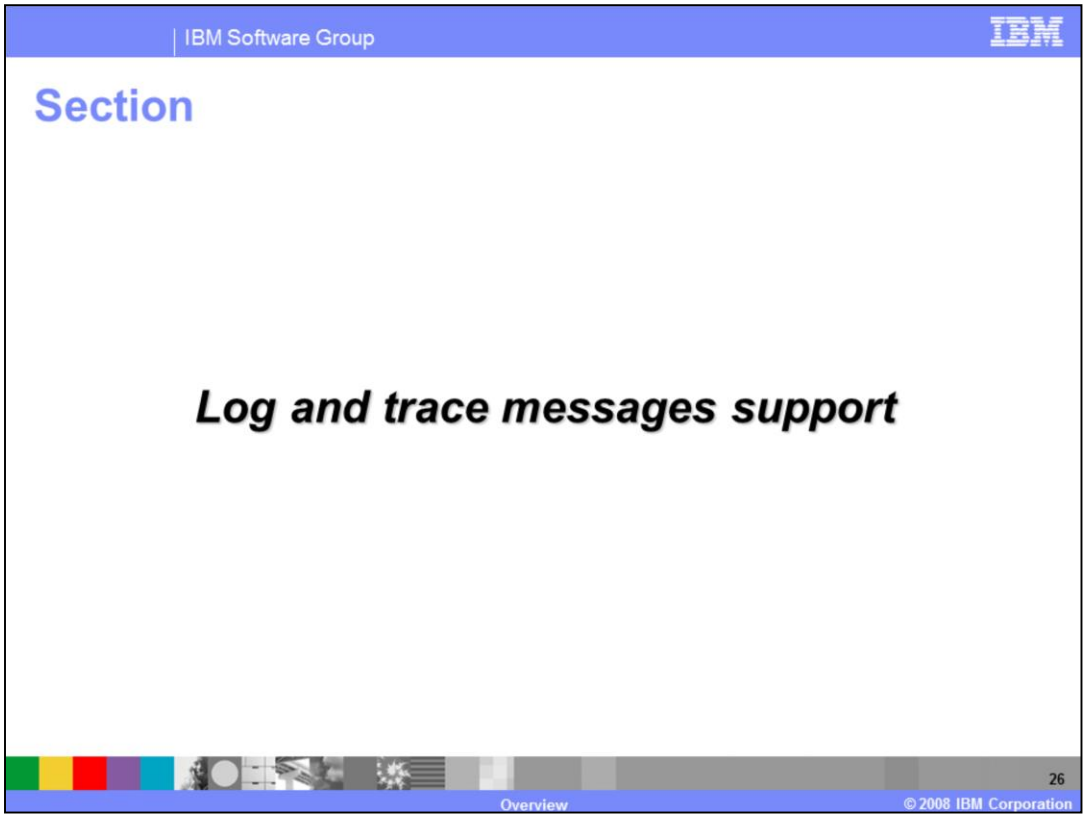
Generated artifacts - Inbound



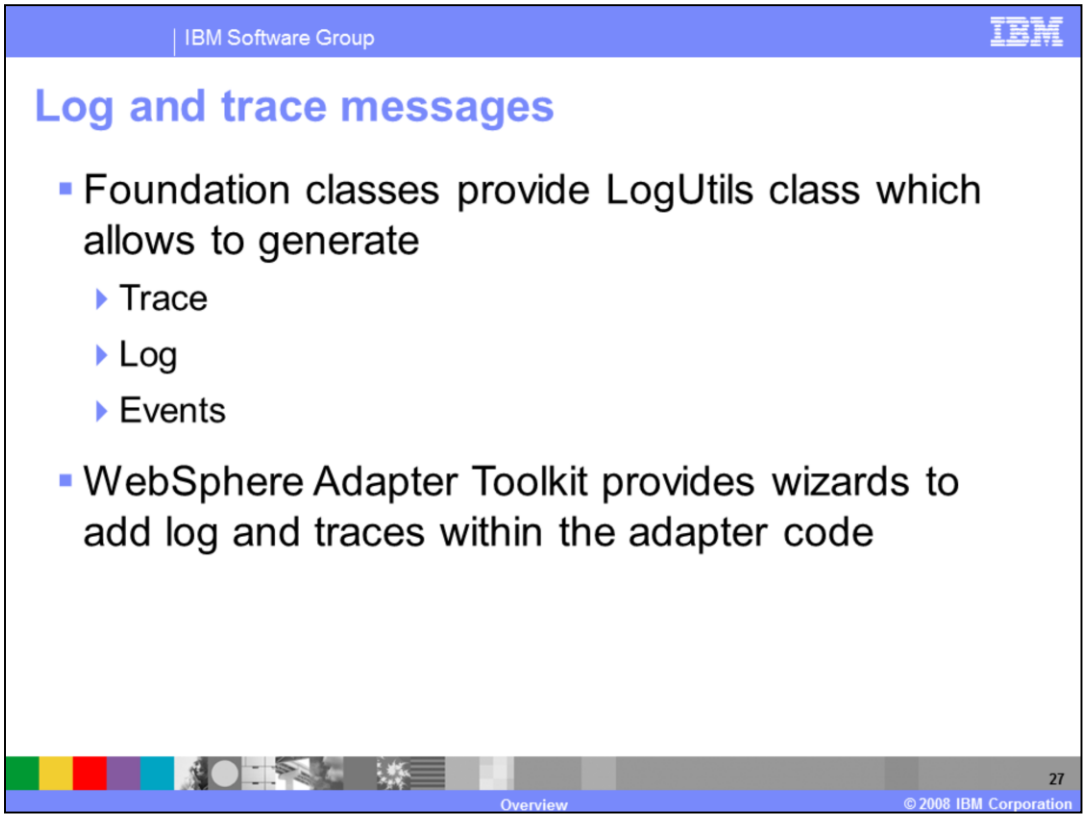
This slide lists the classes that are generated for the inbound support. More information on what you need to implement in each of these classes in provided in the details presentation.



This slide lists the classes that are generated for the meta data discovery support. More information on what you need to implement in each of these classes in provided in the details presentation.

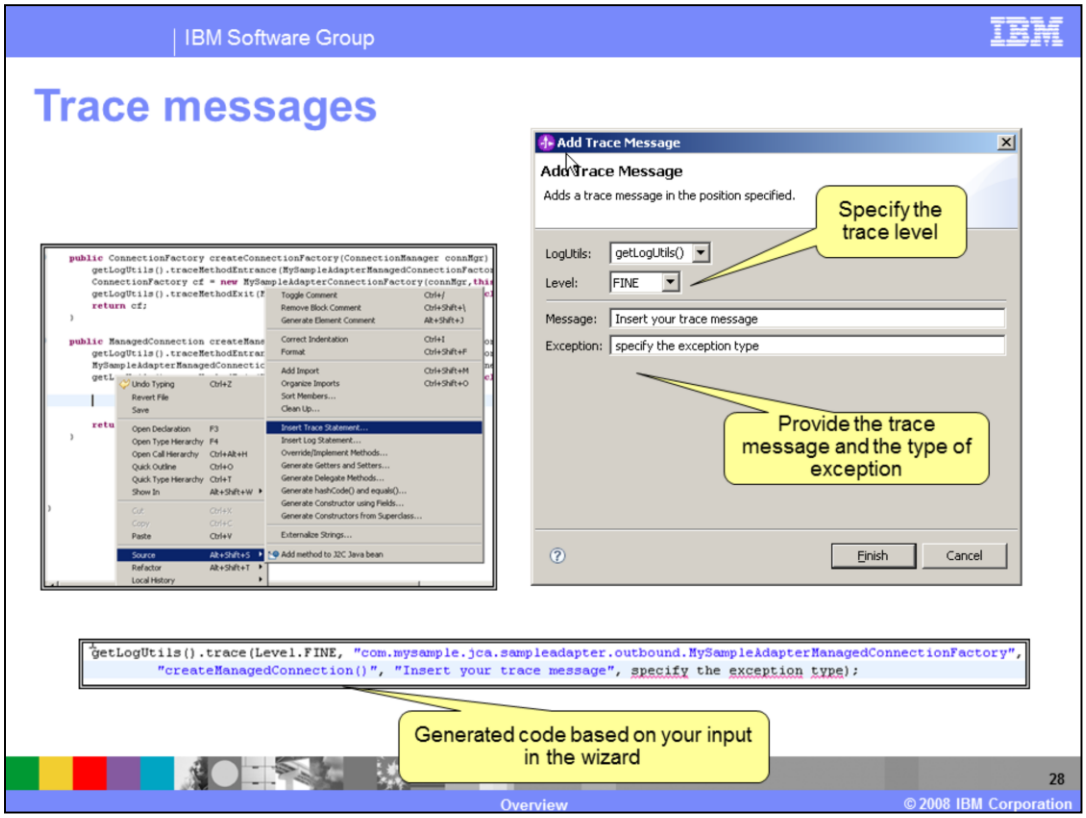


The next section provides details on the support provided by the toolkit for you to include log and trace messages.

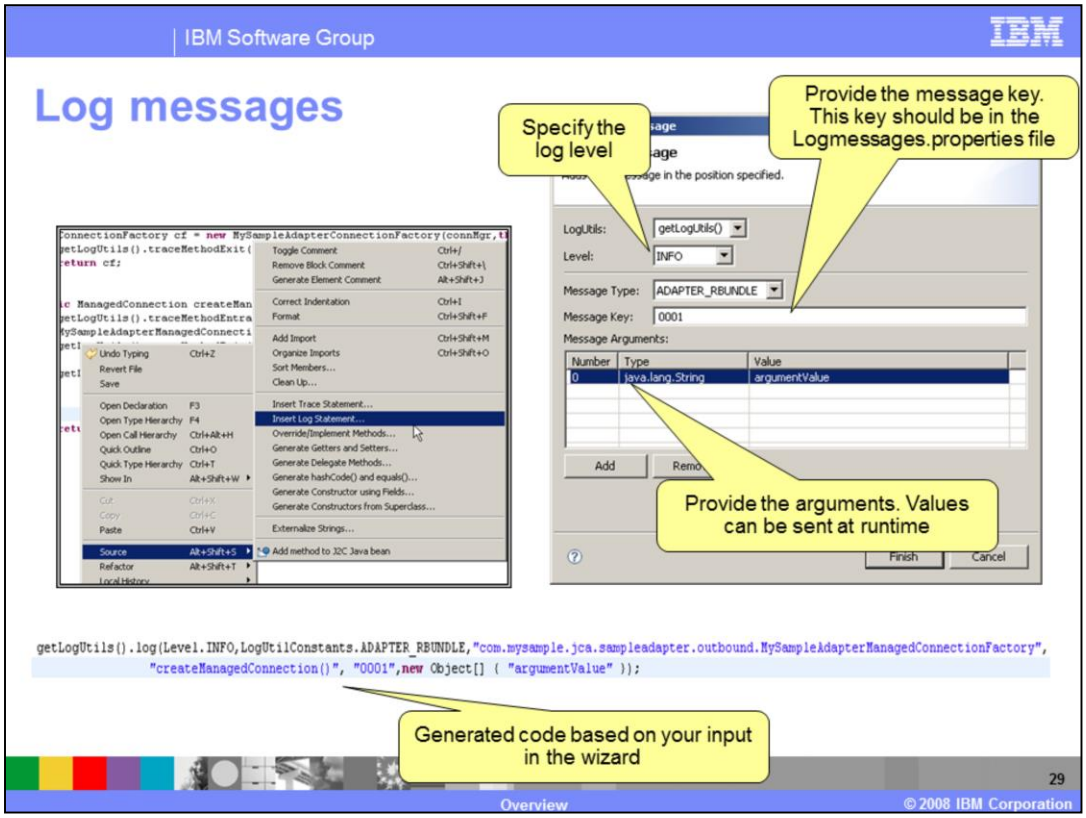


The JCA 1.5 specification provides very minimal support for communicating information to the user. It defines a single stream to which the adapter writes any information. Without additional tools or support, users cannot filter or analyze information.

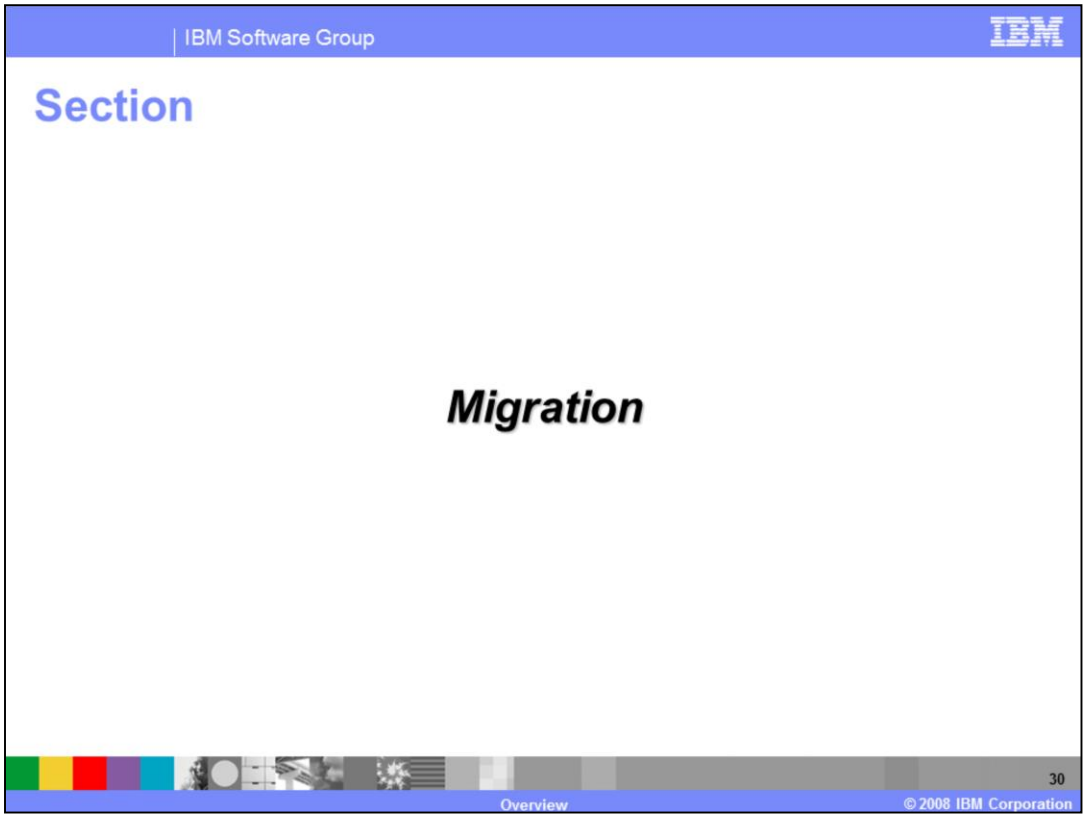
LogUtils class enables you to target different information to different user roles and filter information. Providing information about the runtime state of the adapter is a critical aspect of adapter development. This information is invaluable to support teams trying to resolve problems and the external users looking to track operations and monitor the adapter.



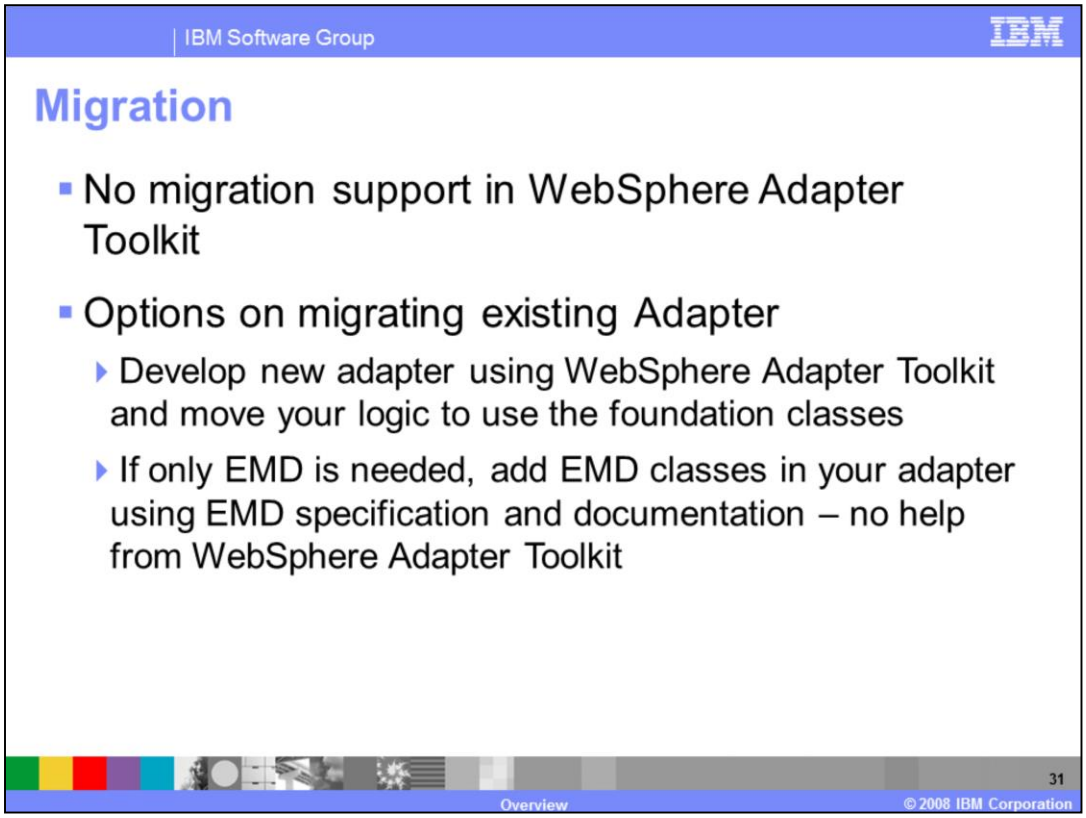
This shows the tools support for including trace messages in your code. Instead of manually writing the code, the tool simplifies the process of including trace statements by providing user interface where you can select various options like the trace level, trace message and the exception. Once these options are provided by you, the tool automatically generates the appropriate trace method call.



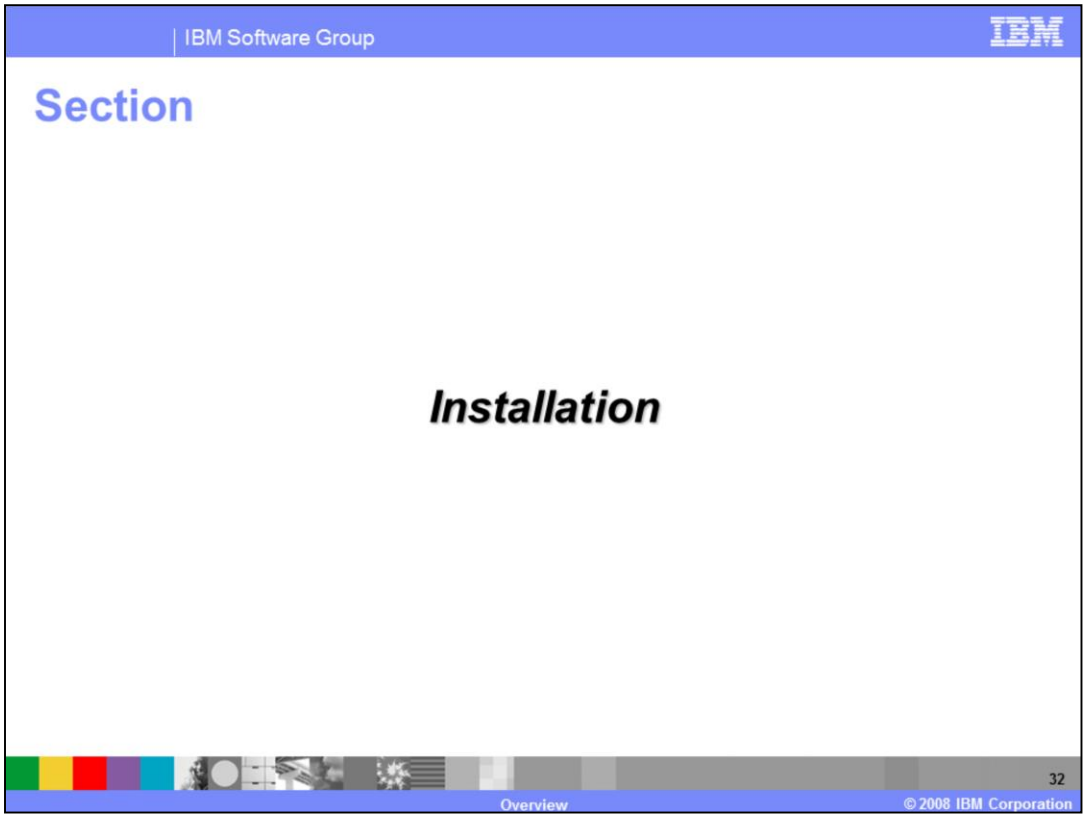
This shows the tools support for including log messages in your code. Instead of manually writing the code, the tool simplifies the process of including log statements by providing user interface where you can select various options like the log level, trace message and the exception. Once these options are provided by you, the tool automatically generates the appropriate log method call.



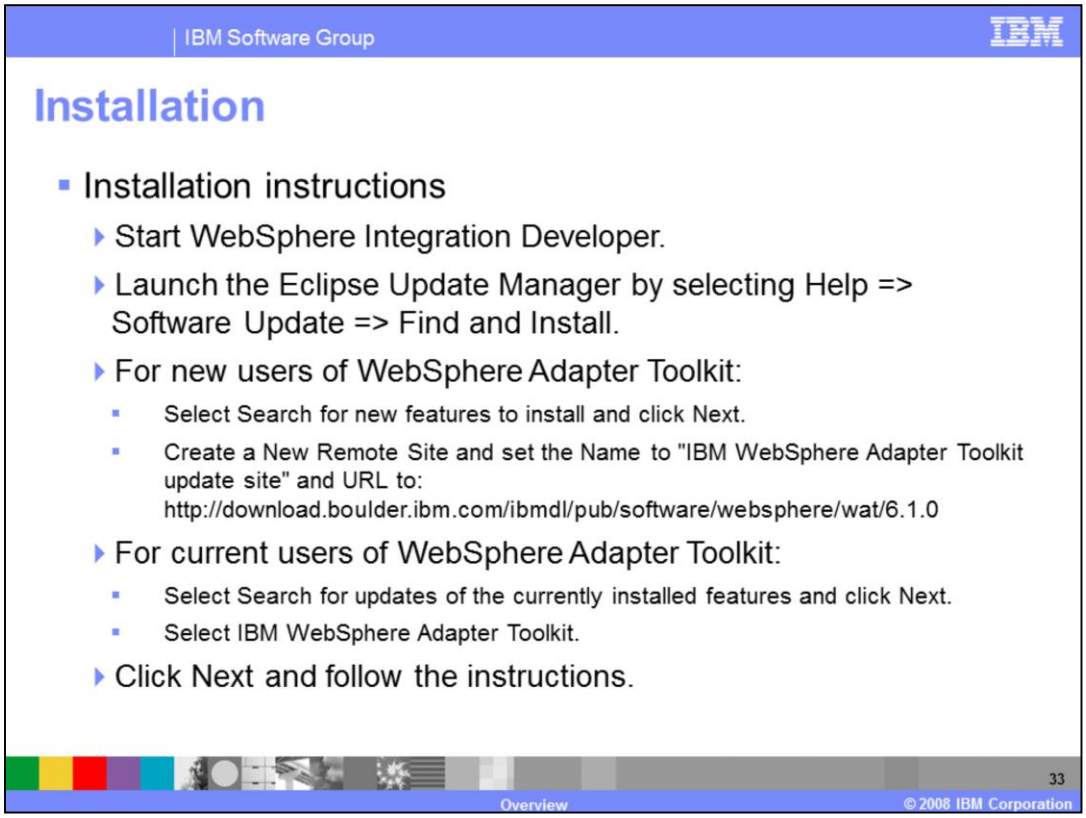
The next section provides details on the migration support provided by the toolkit.



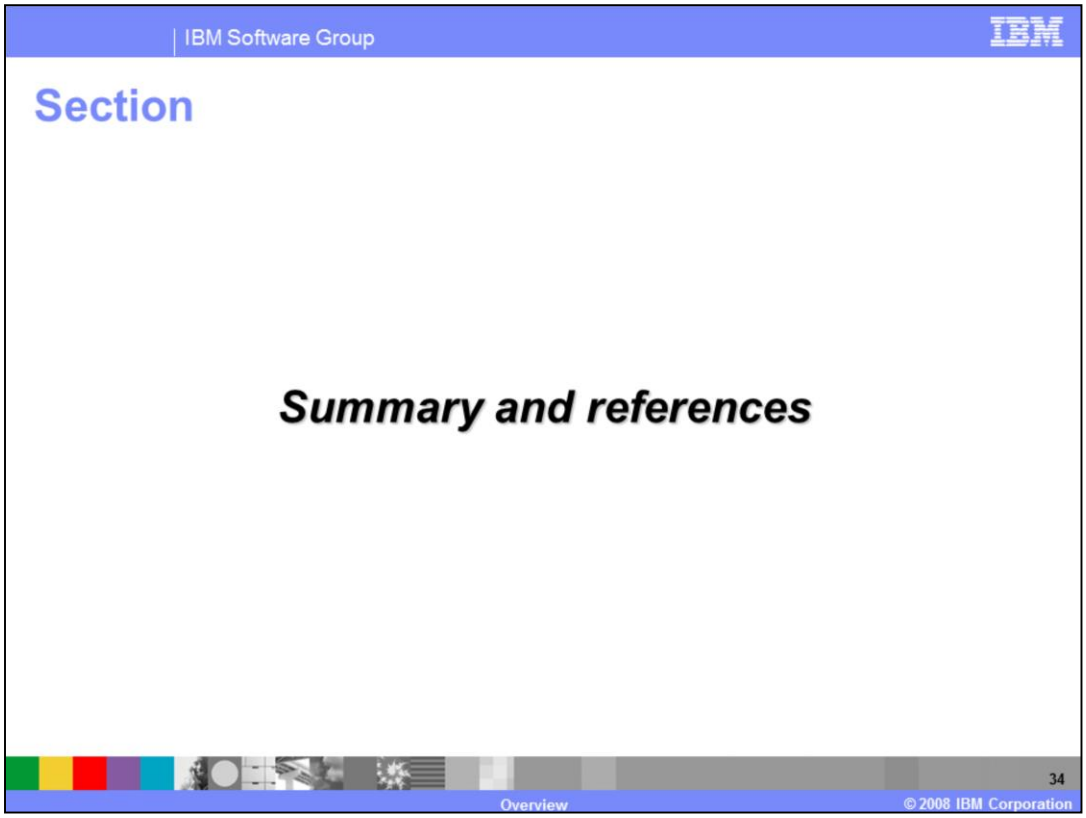
There is no migration support provided by the WebSphere Adapter Toolkit. Options on migrating your existing Adapter include develop new adapter using WebSphere Adapter Toolkit and move your logic to use the foundation classes. If only Enterprise Metadata (EMD) support is needed, add EMD classes in your adapter using EMD specification and documentation.



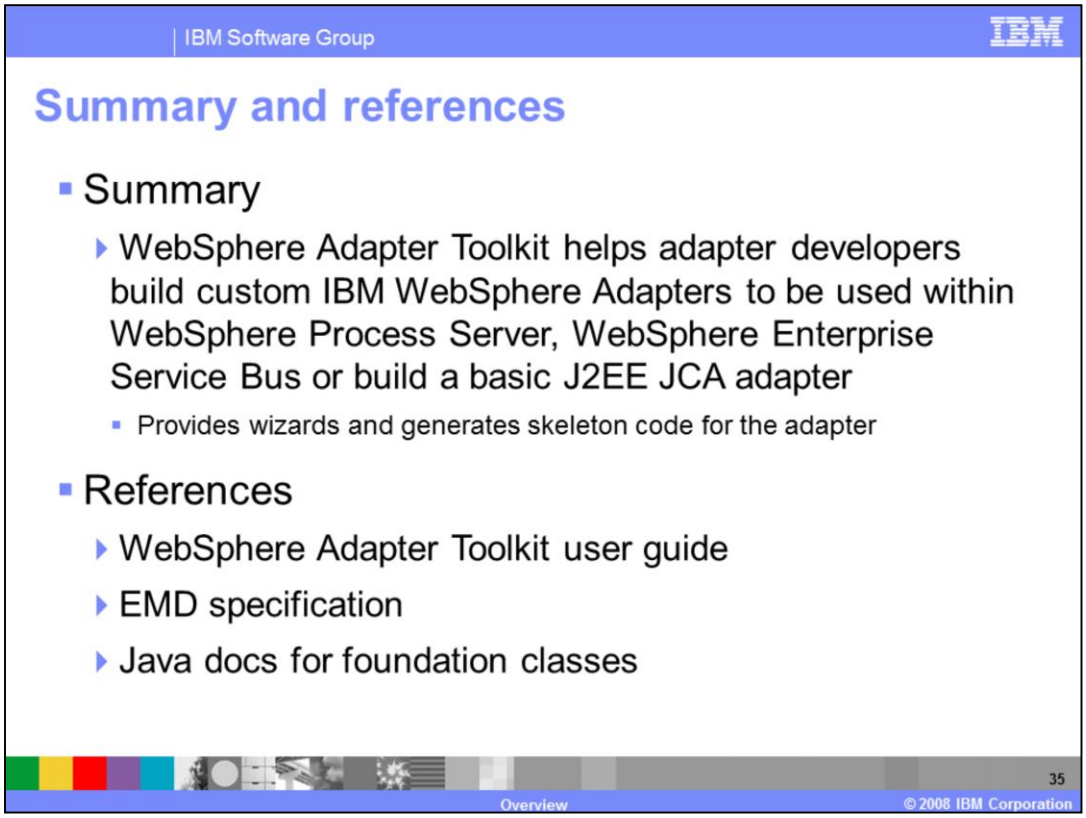
The next section provides the instructions on how to install the WebSphere adapter toolkit.



WebSphere Adapter Toolkit installs on WebSphere Integration Developer as an Eclipse plug-in. The instructions for installation are provided on the slide for reference.



The next slide summarizes what is covered in the presentation and provides references to material that can be used when using the toolkit.



WebSphere Adapter Toolkit helps adapter developers build custom IBM WebSphere Adapters to be used within WebSphere Process Server,WebSphere Enterprise Service Bus or build a basic J2EE JCA adapter. The underlying adapter foundation classes used by the toolkit simplify the process of adapter development by providing implementation for most generic contracts so you only provide the implementation based on the backend you are connecting to.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

J2EE, JDBC, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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.

Information is provided "AS IS" without warranty of any kind. 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.