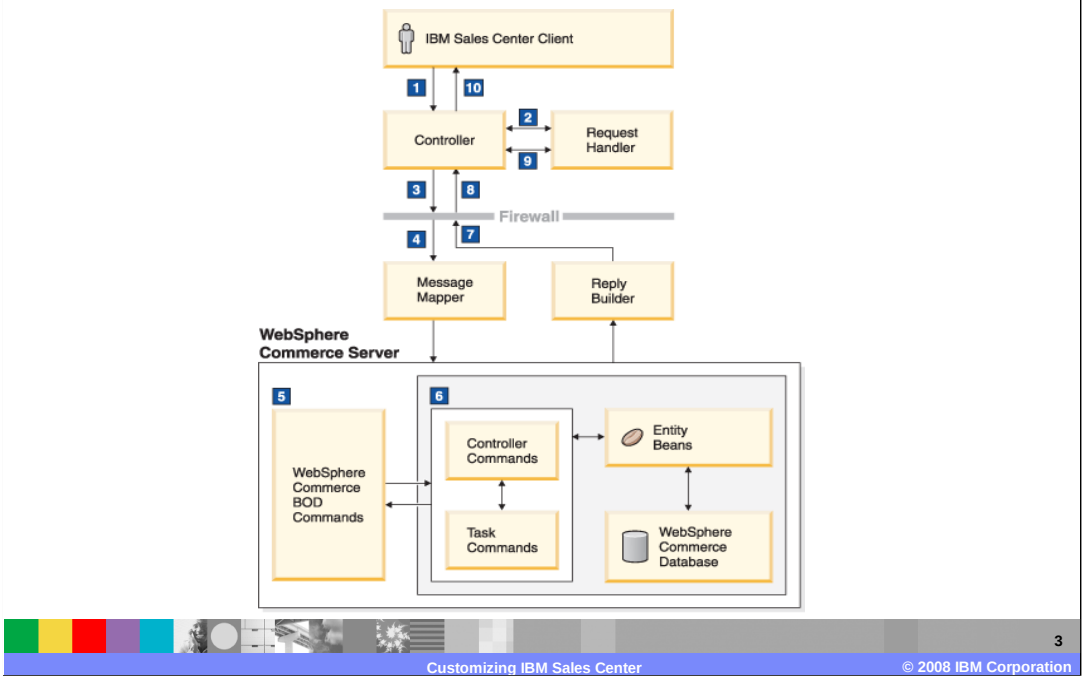
Welcome to the WebSphere Commerce V6 presentation. This presentation discusses customization of IBM Sale Center for WebSphere Commerce

Agenda

- Describe the process necessary to modify IBM Sales Center and the rich client platform
- Outline the behavior of the IBM Sales Center development environment
- List the customizations possible in IBM Sales Center

This presentation describes the steps necessary to modify the behavior of the IBM Sales Center component in WebSphere Commerce. It will outline the behavior of the rich client platform and its role in the extension of IBM Sales Center.

IBM Sales Center architecture



The IBM Sales Center architecture consists of the IBM Sales Center client, WebSphere Commerce Server, and a messaging architecture that allows for customization and extensibility.

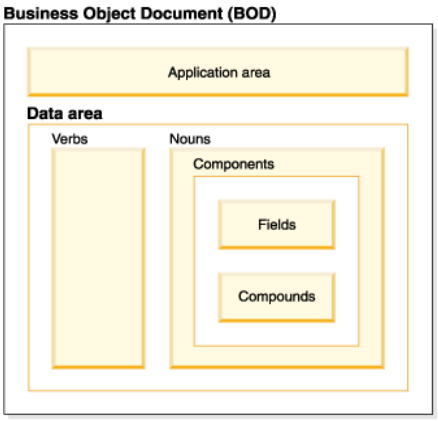
IBM Sales Center uses this messaging architecture to communicate with the WebSphere Commerce Server. It is possible to customize the behavior of any of the three components of this architecture. The customization allows you to change the appearance or behavior of the client, modify or create messages, and customize and extend the server itself.

The following steps take place during the Sales Center request flow processing:

1. The IBM Sales Center client performs a service request.
2. The service request handler prepares a Business Object Document (BOD) message.
3. The message is sent from the client to the host machine.
4. The message mapper receives the message and maps the BOD to a WebSphere Commerce BOD command.
5. The WebSphere Commerce BOD command is invoked.
6. The WebSphere Commerce BOD command calls a WebSphere Commerce Controller command, which might call one or more task commands.
7. The reply, or response, builder constructs the response BOD.
8. The response is returned to the client machine.
9. The request handler receives and handles the response BOD.
10. The client user interface is updated on screen.

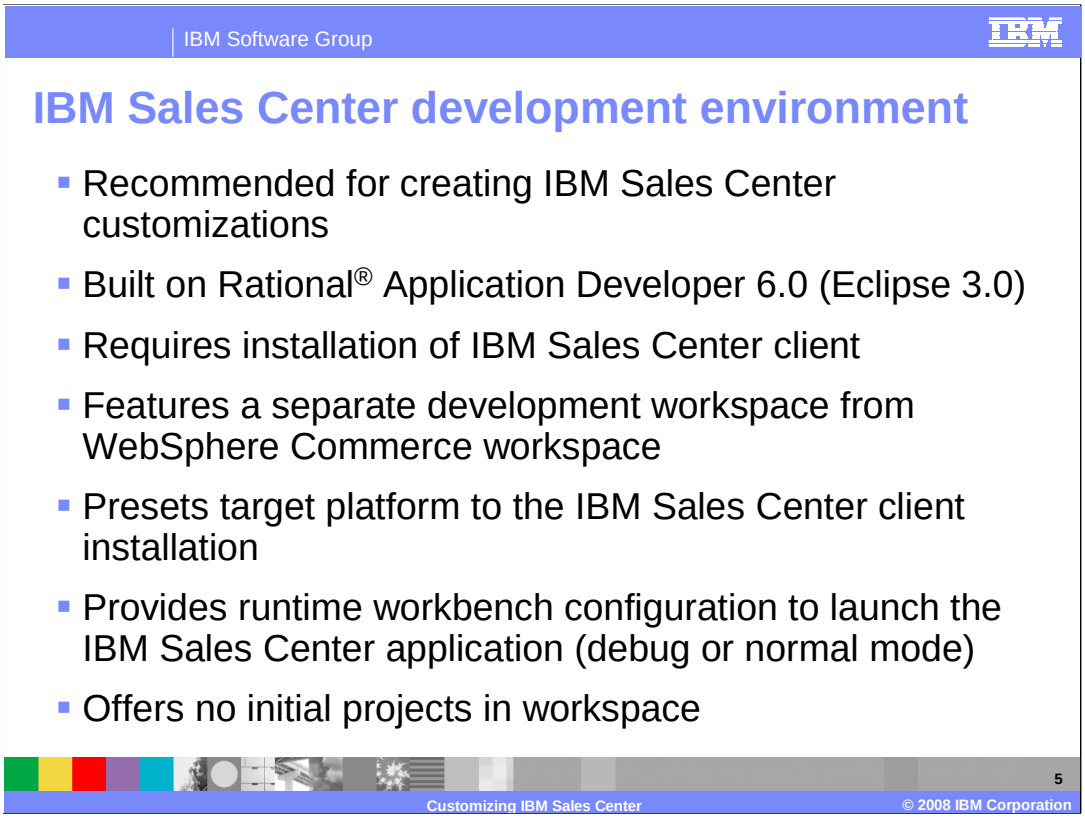
Business Object Documents

- Open standard for horizontal message architecture
- Business messages exchanged between software applications or components



The Business Object Document (BOD) is an open standard for a common horizontal message architecture developed by the Open Applications Group. BODs are business messages exchanged between software applications or components. In this case, they are exchanged between the IBM Sales Center client and server.

For more information on Business Object Documents, see the presentation **OAGIS message architecture**.



The IBM Sales Center development environment is the recommended tool for creating IBM Sales Center customizations. The IBM Sales Center development environment is built on Rational Application Developer V6.0. In addition to Rational Application Developer, the IBM Sales Center development environment also requires that the IBM Sales Center client is installed.

Once installed, the IBM Sales Center development environment is launched in a workspace separate from the WebSphere Commerce workspace. It is configured to launch the Plug-in Development Environment perspective. The target platform is set to the IBM Sales Center client installation, and a runtime workbench configuration is available to launch the IBM Sales Center application (either in debug or normal mode). No projects are initially available in the workspace.

Launching the development environment

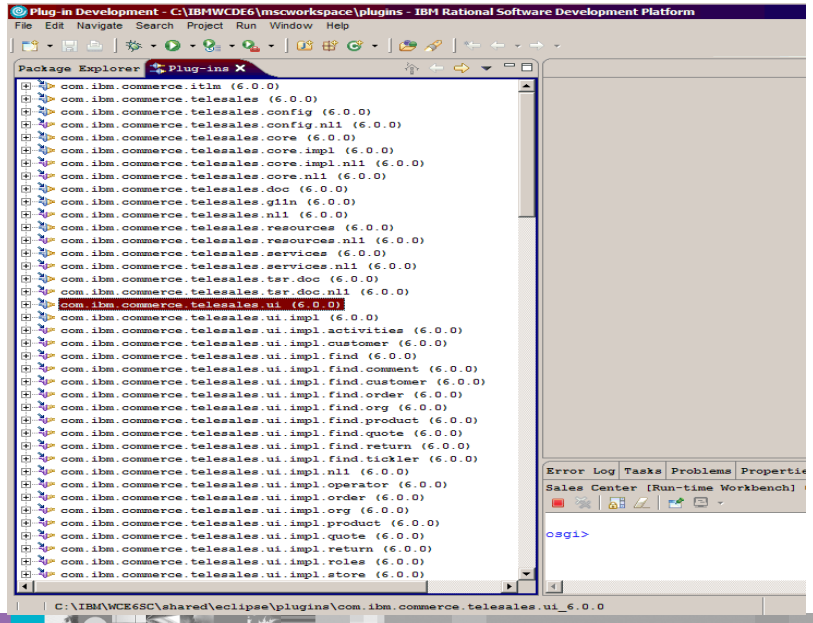
- Select: **Start** > **WebSphere Commerce Developer** > **IBM Sales Center development environment**
- Use the preconfigured Runtime-Workbench configuration in the Run dialog to test customizations
- To launch the IBM Sales Center client within the development environment:
  - Choose **Run...** from the **Run** menu
  - Select **IBM Sales Center** under **Run-time workbench**
  - Click **Run** to launch the application
- Use a running server to log on (either a WebSphere Commerce Server toolkit or a WebSphere Commerce Server)
- Configure server settings before logging on

The IBM Sales Center development environment is launched from the Windows Start menu. The details of menu navigation are shown in this slide.

To test customizations to the IBM Sales Center, a preconfigured Runtime-Workbench configuration is available in the Run dialog. To launch the IBM Sales Center client within the development environment, follow steps shown in this slide.

In order to log on, you need to have a running server. This can either be the WebSphere Commerce Server toolkit running on the same or another system, or it can be a WebSphere Commerce Server. The server settings must be configured before you can log on. Administrators can select the "Connectivity" button on the logon dialog and enter the appropriate host information.

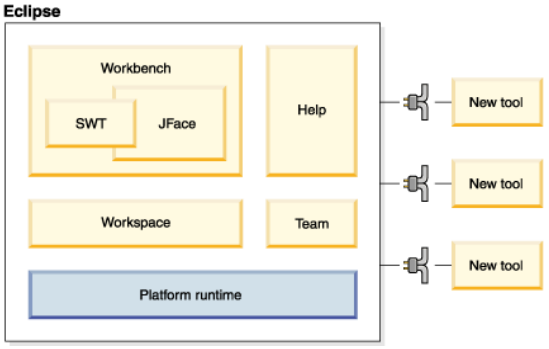
Plug-in development environment



The screen capture in this slide shows the plug-in development interface of IBM Sales Center development environment.

Client and the rich client platform

- Rapid development of integrated features based on the Eclipse plug-in model
- Plug-ins configured using a manifest file (plugin.xml) in Eclipse



The IBM Sales Center client is built on a rich client platform (RCP). This platform provides rapid development of integrated features based on a plug-in model. Plug-ins are structured components that use a manifest file (plugin.xml) in the Eclipse runtime system. The RCP combines the easy management of a browser interface with the interactivity and performance of a client-server interface.

New tools interface with the RCP through a mechanism called a plug-in. Plug-ins are structured components that are configured using a manifest file (plugin.xml) in the Eclipse runtime system. For in-depth coverage of plug-ins, refer to the Eclipse platform documentation.

IBM Sales Center editors

- Used to create and manage orders, quotes, ticklers, and returns
- Where Customer Service Representatives mainly work
- Composed of one or multiple editor pages
- Opens multiple editors and multiple instances of the same editor
- Performs customization by using Eclipse product extension point architecture

There are several types of editors in the IBM Sales Center. These editors are used to create and manage orders, quotes, ticklers, returns, and customers. Editors are where the Customer Service Representatives do the majority of their work.

Editors can be composed of one or multiple editor pages. The default Sales perspective consists of an editor area and one or more views. Any number of editors can be open at the same time, but only one editor can be active. In addition, multiple instances of the same editor type can be open. However, if one editor is already open for a specific store or order, a second editor instance for that store or order cannot be opened.

The contents of editor pages are user interface elements known as controls and managed composite controls. There are several editors already provided with the IBM Sales Center. Customization of these editors involves using the Eclipse product extension point architecture.

Types of editors in IBM Sales Center client

- Store Summary editor
- Customer editor
- Organization Summary editor
- Order editor
- Quote editor
- Product Compare editor
- Create Tickler editor
- Work on Tickler editor
- Web browser editor

This slide lists the various IBM Sales Center editors. Additional editors are available through the Eclipse framework.

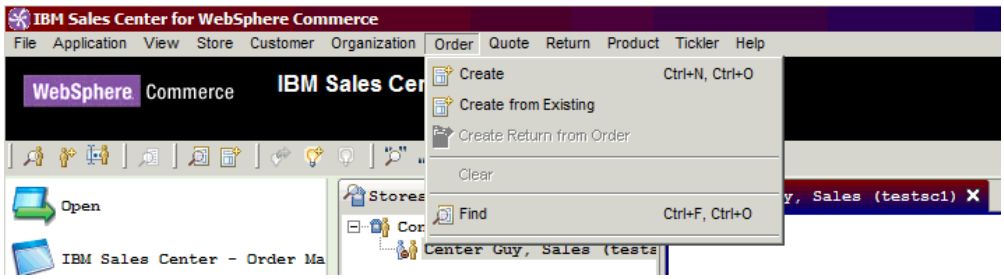
IBM Sales Center dialogs

- User interface action elements
- Standard Eclipse widget toolkit layout architecture
- Eclipse product extension point architecture performs customization

Dialog is a common user interface element in the IBM Sales Center. Dialogs are used when a user chooses to perform an action and needs to provide additional information. For example, the interface that a user uses to log on, or search for an order, is a dialog. The contents of dialogs are user interface elements known as controls and managed composite controls. Customization of these dialogs involves using the Eclipse product extension point architecture.

Menus to access editors

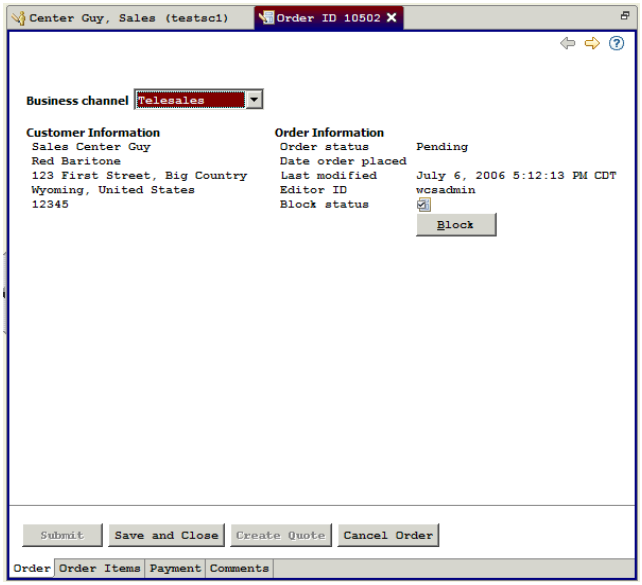
- Menus provide access to editors in the IBM Sales Center client interface. Shown below is the menu to support order management



The IBM Sales Center representative, or customer service representative, has access to editors through a menu system.

Editor example

- A customer service representative creates an order for a shopper after selecting **Order -> Create** from the menu



The screen capture in this slide shows the order editor interface an IBM Sales Center representative or a customer service representative uses to create orders for a shopper.

Controls and control factories

- UI elements are used to construct the layout of *editors* and *dialogs*
- A *control* has a unique ID, a type, and various properties and attributes
- A *composite control* is a control that has child controls
- Layout defined by compositeDefinitions extension point
  - Grid
  - Form

The layout of the IBM Sales Center dialogs and editors are constructed by using configured controls. A configured control is a user interface element that is declared by using the controls extension point. The definition of a configured control includes a unique identifier, the type of control, and additional attributes and properties that are specific to the type of control being defined. The control type is used to locate a control factory class that will construct the new configured control.

The individual user interface elements in the IBM Sales Center user interface are controls. The groups of controls in the IBM Sales Center are known as composite controls. If the control type is "composite" then the composite control factory will construct a configured composite control. A composite control is a control that has child controls. The layout and children of a composite control are defined by using the compositeDefinitions extension point. Composite definitions can be either grid layout or form layout.

Controls and control factories

- *Control* behavior (initializing, refreshing, saving, removing) is managed by *widget managers*
- *Control* factories interpret control declarations and construct new configured control instances

The behavior of the controls on the IBM Sales Center user interface is managed by widget managers. A widget manager is declared by using the widgetManagers extension point. Widget managers initialize, refresh, save, and dispose of configured controls. They are also responsible for providing the content of table cells. Widget Managers can add complex behavior to controls by adding listeners and handlers to the control during initialization. A standard widget manager is provided with the IBM Sales Center and is used with standard controls and tables. When a configured control is defined, it can be declared with a manager type. The manager type is read by the widget manager to decide if the control should be managed by the widget manager. Widget managers will only manage controls that have a manager type that they recognize. If a control does not declare a manager type, it is assumed to be "standard".

Control factories are responsible for interpreting control declarations and constructing new configured control instances. The controlFactories extension definition is declared with a specific control type. This type is used to map control declarations to the appropriate control factory. The IBM Sales Center includes control factories for all of the control types used to construct the user interface. You can reuse these control types if you are customizing the IBM Sales Center, or you can define your own control factories.

User interface customization elements

- Preference pages are useful when a CSR needs to work with specific orders, customers, and stores
- Accelerator key bindings are Eclipse hot-keys that associate a shortcut with a command
- An infopop is a small window associated with a widget

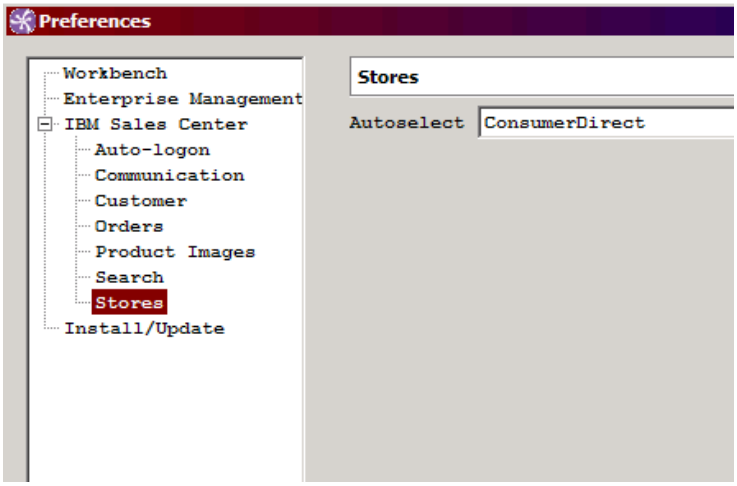
There are preference pages for communication preferences, logon preferences, customer preferences, order preferences, store preferences, search preferences, and image preferences. Preference pages in the IBM Sales Center are defined by using the preferencePages extension point.

The default IBM Sales Center accelerator keys are defined by the commands extension point in the telesales plug-in. This extension point allows you to define a command and a keyBinding for that command. The commands are associated with Actions by calling the setActionDefinitionId method on the Action class. Commands can also be associated with actions by specifying the command ID on the definitionId attribute of the action element of the actionSets extension point. The key bindings are associated with a key configuration.

An infopop displays when you put focus on the widget and presses the F1 key. It displays context-sensitive help and links to related help topics for the widget. They are registered by using the help context extension point. For the IBM Sales Center user interface elements, the mappings are done programmatically by registering the infopop ID, using the setHelp method on the class.

Preferences

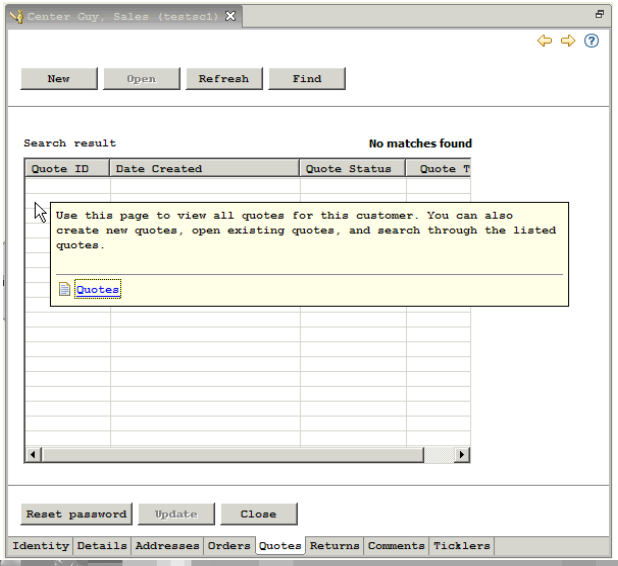
- Set a preference for ConsumerDirect to be the default store



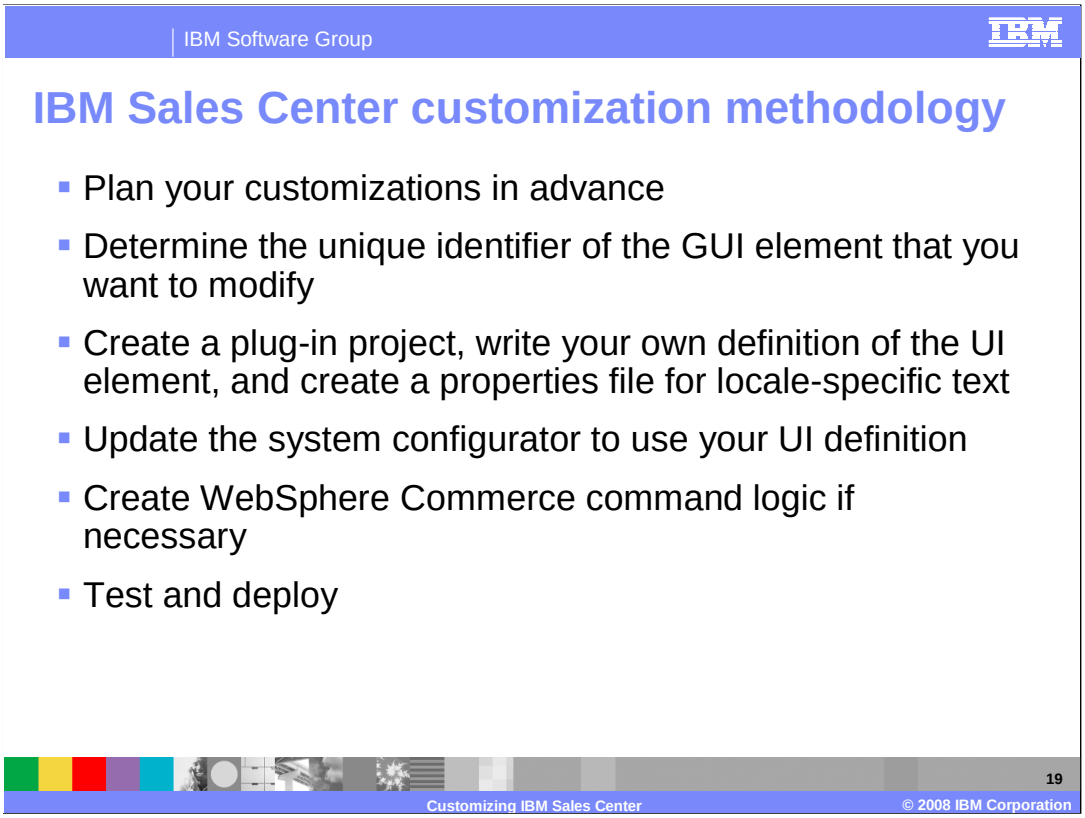
You access the Preferences window by using the File -> Preferences menu.

Infopop usage

Pressing F1 after focusing on a widget displays a Help window:



To view an infopop in the IBM Sales Center client, press F1 key after you place focus on an element.



Here are the general steps to customize the IBM Sales Center:

1.Determine what you want to customize.

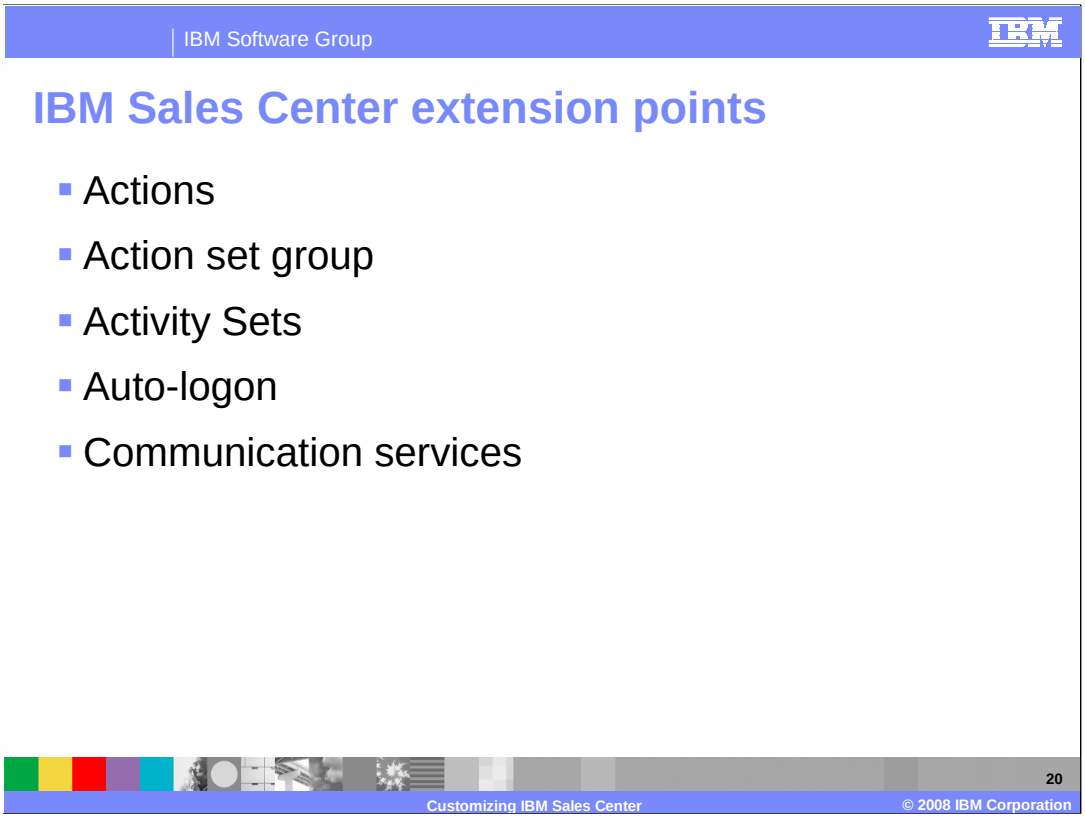
2.Determine the unique ID of the graphical user interface element that you want to modify.

3.Create a plug-in project in the IBM Sales Center development environment. Write your own definition of user interface element. If you want to reuse existing user interface elements, use the referenceId to reference the existing definition instead of redefining it in your plug-in. If data must be sent from the client to the server, you must add information to your user interface elements. Create a properties file to store any locale-specific text. Extend the resource bundle extension point to indicate the name and location of your properties file.

4.Declare the system configurator using the system configurator extension point by creating a config.ini file. In this file, indicate that the IBM Sales Center should use your definition of user interface element instead of the default definition.

5.If necessary, write code on the WebSphere Commerce server side to process data.

6. Test your changes. Once you are satisfied with your changes, export the plug-in and distribute the plug-in.



A variety of extension points are available within the IBM Sales Center. Additional extension points are available through the Eclipse framework

An actions extension point allows plug-ins to implement Multichannel IBM Sales Center actions.

An action set group extension point allows plug-ins to define a group of actionSets which are loaded by the Multichannel IBM Sales Center perspectives.

Activity Sets allow plug-ins to define which activities WebSphere Commerce roles support.

An auto-logon extension point allows plug-ins to identify which Web applications support auto-logon.

A communication services extension point notifies a communication service when a service request needs to transmit a message to a server. The communication service implementation is expected to obtain a response, which is customizable by way of JAX-RPC or another implementation.

IBM Sales Center extension points

- compositeDefinitions
- controlFactories
- controls
- Currency format
- dynamicIdResolvers

A compositeDefinitions extension point is used to declare configured composite definitions.

A controlFactories extension point is used to declare control factories.

A controls extension point is used to declare controls for use in the IBM Sales Center user interface. Controls can be added to and removed from existing user interfaces or assembled together into a new user interface.

A currency format extension point currencyFormat is to allow the globalization plug-in to implement locale-specific currency formatting.

A dynamicIdResolvers extension point is used to declare dynamic ID resolvers. Dynamic ID resolvers are responsible for resolving a dynamic ID into a valid extension ID .

IBM Sales Center extension points

- dynamicIds
- Editor pages
- Feature validation service
- formLayouts
- gridDatas

A dynamicIds extension point is used to declare identifiers that are dynamically resolved at runtime.

An editorPages extension point provides a grouping of pages for an associated editor so the plug-ins can register their pages for a defined editor.

A feature validation service extension point allows plug-ins to register features for validation with the WebSphere Commerce server before authentication.

A formLayouts extension point is used to declare form layouts.

A gridDatas extension point is used to declare grid data that can refer to grid data declarations.

IBM Sales Center extension points

- gridLayouts
- Images extension point
- managedComposites
- Model object extension point
- Resource bundle extension point

A gridLayouts extension point is used to declare grid layouts that can refer to grid layout declarations.

An images extension point allows custom images to be used instead of the default images.

A managedComposites extension point is used to declare managed composites for use in the IBM Sales Center user interface. Managed composites encapsulate a configured composite and one or more widget managers.

A model object extension point supports extensions to the IBM Sales Center model objects. Model objects represent client side business objects.

A resource bundle extension point centralizes the resources for the Multichannel IBM Sales Center implementation plug-ins, allowing others to provide custom resources definitions.

IBM Sales Center extension points

- Roles
- rowDatas
- rowLayouts
- Service requests
- stackLayouts

A roles extension point allows plug-ins to define which activities WebSphere Commerce roles support.

A rowDatas extension point is used to declare row data that can refer to row data declarations.

A rowLayouts extension point is used to declare row layouts for use in the Multi-Channel Sales Center user interface.

A service requests extension point is used to define a service request used to locate and run the associated request handler class.

A stackLayouts extension point is used to declare stack layouts that can refer to stack layout declarations.

IBM Sales Center extension points

- Statusline contribution items
- System configurator
- tabFolderDefinitions
- tableDefinitions
- widgetManagers

A statusline contribution items provide a contribution item for the status line manager that is displayed for a perspective so plug-ins can register their contributions.

A system configurator extension point specifies a system configurator for identifying which extensions are used over the default extensions so plug-ins can specify the location of their system configurator.

A tabFolderDefinitions extension point is used to declare tab folder definitions that can refer to tab folder declarations.

A tableDefinitions extension point is used to declare configured table definitions that can refer to table declarations.

A widgetManagers extension point is used to declare widget managers that are responsible for providing behavior to configured widgets.

Extension point customization examples

The System Configurator extension point defines a configurator for each plug-in:

- Identifies which custom extensions are used over the default extensions
- Allows changes to menus, editors, views, dialogs, preferences, UI widgets, model objects, and request handlers
- Uses the file name **config.ini** (java.util.Properties format) and must be located in a directory called **config**

```
<extension point="com.ibm.commerce.telesales.configurator">
  <configurator path="config"/>
</extension>
```


The system configurator extension point is defined in the telesales plug-in. The IBM Sales Center platform provides the ability to specify a system configurator for each plug-in to identify which extensions are used over the default IBM Sales Center extensions. The purpose of this extension point is to allow plug-ins to specify the location of their system configurator file. The file name must be config.ini. This file is a text file that allows you to provide a substitute extension ID that is used to replace a standard IBM Sales Center extension ID. The code shown in the slide is a sample of a system configurator extension definition. The configurator file config.ini must be located in a directory called config that is located in the root directory of the plug-in that is defining the extension.

To suppress an extension, use an ID value of null.

Extension point customization examples

- An example of customizing the logon dialog using the *dialogs extension point*

```
<extension point= "com.ibm.commerce.telesales.ui.dialog" >
        <dialog id= "com.ibm.commerce.telesales.logonDialog" class=
"com.ibm.commerce.telesales.ui.dialogs.LogonDialog" >
        </dialog>
</extension>
```


The code shown in this slide is an example of customizing the logon dialog using the dialogs extension point.

Extension point customization examples

- The Resource Bundle extension point defines where the Sales Center client first looks for a property key:

```
<extension point=
"com.ibm.commerce.telesales.resources.resources.resources">
        <resourceBundle baseName="com.mycompany.resources">
        </resourceBundle>
</extension>
```


This slide shows a code example of a resource bundle extension point.

Extension point customization examples

- The images extension point indicates that images found in the declaring plug-in override base IBM Sales Center images

```
<extension point="com.ibm.commerce.telesales.resources.images"
id="com.mycompany.resources.images" name="Images"/>
```


This code shows an example of an images extension point and indicates that images found in the declaring plug-in override base IBM Sales Center images.

Potential extension point customizations

- View product availability
- Create quotes and turn quotes into orders
- Create, update, block, and cancel orders
- Process payments
- Create and manage Ticklers (“To Do’s”) for themselves or other CSRs
- Find and visually compare products
- View cross-sell, up-sell, and promotional information
- Work with multiple stores, customers, and orders simultaneously
- View and override contract and list pricing (products, orders, shipping)

This slide lists possible customization scenarios using the capabilities of the IBM Sales Center.

IBM Sales Center data model

- Used to cache business objects on the client
- Contains model objects that represent the operator, customers, orders, products, and other entities with which a Customer Service Representative interacts
- Subclasses the base model object when implementing ModelObject
- Customizes the model object by storing additional properties or adding listeners
- Uses a factory class, TelesalesModelObjectFactory, to create new instances of model objects

The data model is used to cache business objects on the client. The TelesalesModelManager class provides access to the model and contains convenience methods for accessing and updating child objects. The default IBM Sales Center data model contains model objects that represent the operator, customers, orders, products, and other commonly used objects. All of the objects found in the client cache are instances of ModelObject.

You can store additional properties in a model object by using the setData and getData methods. You can also add listeners to model objects that are notified of any change to the model object. If a property is a list of model objects, use the ModelObjectList instead of a generic container object to ensure that property notification is properly handled.

There is a factory class, TelesalesModelObjectFactory, that should be used to create new instances of model objects. There is also a model object extension point that lets you register new model objects. This factory and extension provide support for replacing existing model object implementations with a new implementation. The new implementation must subclass the base model object.

Integration with WebSphere Commerce

- Browser integration allows a CSR the ability to switch to the WebSphere Commerce application
- Auto-logon provides automatic login to Accelerator and Organization Administration Console
- Business Object Document messages exchanged between IBM Sales Center and WebSphere Commerce Server are mapped to controller commands

```
<serviceRequest
    label="Logon“
    requestHandlerClass="extensions.ExtendedProcessLogonRequest“
    id="extensions.logon“
    commServiceId="com.ibm.commerce.telesales.services.TsCommunication">
</serviceRequest>
```

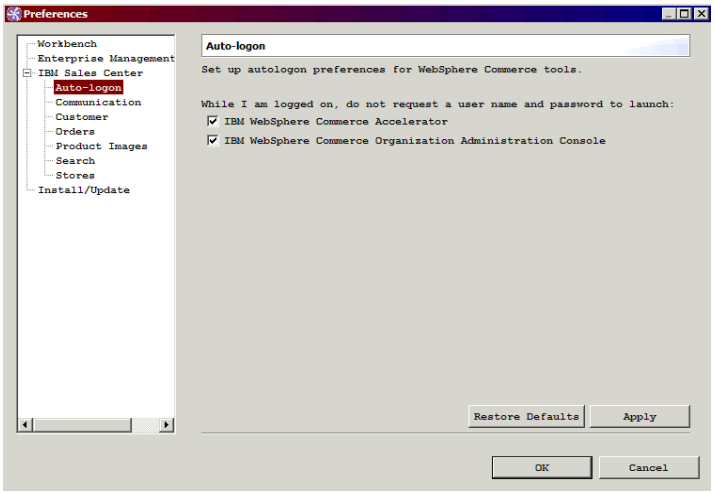

You can integrate the IBM Sales Center application with an existing WebSphere Commerce application. You can leave part of the application browser interface intact and switch to the browser to interact with that portion of the application. Browser integration support is provided by declaring an extension to the WebSphere Everyplace Deployment extension point.

The IBM Sales Center client provides default auto-logon support for the WebSphere Commerce Accelerator and the WebSphere Commerce Organization Administration console. The auto-logon feature can be extended to any external web-based application. The IBM Sales Center uses the browser application support provided by WebSphere Everyplace Deployment. The extension point that is used to define the IBM WebSphere Commerce Accelerator and IBM WebSphere Commerce Organization Administration Console applications is WctWebApplication.

Business Object Document (BOD) messages sent from the IBM Sales Center client to the WebSphere Commerce Server are mapped to WebSphere Commerce controller commands through the WebSphere Commerce message mapper. The WebSphere Commerce configuration file contains this message mapper entry.

Auto-logon

- Set a preference to log in automatically to WebSphere Commerce administration interfaces



The screen capture in this slide shows the Preferences window where you can setup auto-logon with WebSphere Commerce. You can access the Preferences window by using the File -> Preferences menu.

Summary

- Explain the IBM Sales Center and the rich client platform
- Describe the IBM Sales Center development environment
- Outline IBM Sales Center customization scenarios
- Recognize the IBM Sales Center data model
- Understand how to integrate IBM Sales Center and WebSphere Commerce

This presentation described IBM Sale Center platform and its development environment. It introduced the steps necessary to modify the behavior of the IBM Sales Center component in WebSphere Commerce. It also discussed how to integrate IBM Sale Center with WebSphere Commerce applications.

References

- The Business Object Document (BOD)
  http://www.openapplications.org
- WebSphere Commerce V6 Information Center
  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp
- Eclipse 3.0 documentation
  http://www.eclipse.org/documentation/main.html

For more information regarding WebSphere Commerce Sale Center, visit the sites indicated in the presentation.

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_wcs60_CustomizingIBMSalesCenter.ppt

This module is also available in PDF format at: ../wcs60_CustomizingIBMSalesCenter.pdf

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

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 Rational WebSphere

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

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation 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.