IBM WebSphere InterChange Server

Implementation Guide for WebSphere InterChange Server

V4.2.2

IBM WebSphere InterChange Server

Implementation Guide for WebSphere InterChange Server

V4.2.2

Note!
Before using this information and the product it supports, read the information in “Notices” on page 303.

19February2004

This edition of this document applies to IBM WebSphere InterChange Server, version 4.2, IBM WebSphere Business Integration Adapters 2.3.1, and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email doc-comments@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this document

IBM(R) WebSphere(R) InterChange Server and its associated toolset are used with IBM WebSphere Business Integration Adapters to provide business process integration and connectivity among leading e-business technologies and enterprise applications.

This document provides an overview of the tasks for implementing a business integration system with WebSphere InterChange Server.

Audience

This document is for consultants and developers who develop WebSphere InterChange Server integration components and systems.

Prerequisites for this document

Before using this document you should first have read the *Technical Introduction to IBM WebSphere InterChange Server*.

Related documents

The complete set of documentation available with this product describes the features and components common to all WebSphere InterChange Server installations, and includes reference material on specific components.

You can install the documentation from the following sites:

- For InterChange Server documentation:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
- For collaboration documentation:
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- "For WebSphere Business Integration Adapters documentation:
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter

These sites contain simple directions for downloading, installing, and viewing the documentation.

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic, *italic*	Indicates a variable name or a cross-reference.
blue text	A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.

[ ]	In a syntax line, square brackets surround an optional parameter.
...	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
< >	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><connector_name>tmp.log`.
/, \	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All IBM WebSphere InterChange Server product pathnames are relative to the directory where the WebSphere InterChange Server product is installed on your system.
UNIX:/Windows:	Paragraphs beginning with either of these indicate notes listing operating system differences.
%text% and $*text*	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a UNIX environment is $*text*, indicating the value of the *text* UNIX environment variable.

Using this guide

This guide is designed to:

- Introduce the high level stages involved in an implementation of WebSphere InterChange Server, from assessment of customer requirements to deployment of a live system
- Provide a sequential guideline to the tasks that developers perform in developing and configuring the components and interfaces of a WebSphere InterChange Server solution
- Provide a hub document from which to navigate to appropriate development or system documents for specific development tasks

This guide contains both new information and some configuration information that previously resided in the *System Administration Guide*. For tasks that might be performed by both developers of WebSphere InterChange Server components and administrators of a live, deployed system, this guide refers to the *System Administration Guide*.

New in this release

New in release 4.2.2

For this release, the following changes have been made to the product:

- The workflow for deploying components has changed. For more information, see “Deploying components using the deployment wizard” on page 68.
- You can now deploy components by using drag-and-drop in System Manager. For more information, see “Deploying components using drag-and-drop” on page 72.
- A new perspective named Collaboration Debugger allows you to pause the execution of a collaboration object and view the data of a flow as it is processed. For more information, see Chapter 13, “Using Collaboration Debugger,” on page 273.
- There have been significant changes to the Integrated Test Environment. For more information, see Chapter 12, “Using Integrated Test Environment,” on page 215.
- There have been minor changes to several System Manager interfaces. For more information on System Manager, see Chapter 4, “The IBM WebSphere InterChange Server tools and environment,” on page 31.
- The tool referred to as web-based System Monitor in previous releases is now referred to as System Monitor. The functionality of the tool referred to as Windows-based System Monitor in previous releases has been incorporated into a view in System Manager named InterChange Server Component Management.

New in release 4.2.1

For this release, the following changes have been made to the product:

- System Manager now features an interface to configure the memory checker thread. For more information, see “Using the memory checker thread” on page 298.
- You can now validate packages in System Manager prior to deploying them. For more information, see “Validating a package using System Manager” on page 66.
- In the event of a database deadlock, InterChange Server now retries the deadlocked transactions instead of failing. You can configure the number of retries and the interval of time in between each retry. For more information, see “Configuring database properties using System Manager” on page 103.
- A new option `-nc` has been added to repos_copy to specify the encoding of a text file from prior to release 4.2.0 that is being imported. For more information, see Chapter 6, “Using repos_copy,” on page 113.

New in release 4.2.0

For this release, the following changes have been made to the product:

- CrossWorlds System Manager has been replaced with the Eclipse-based tooling framework, and its functionality is provided by a plug-in named System Manager within the framework. For more information, see Chapter 4, “The IBM WebSphere InterChange Server tools and environment,” on page 31.
- A new tool plug-in named Integrated Test Environment provides the ability to test an entire integration interface.

- The InterChange Server development model has changed significantly from previous release. For an overview of the new development model, see Chapter 1, "The IBM WebSphere InterChange Server development model and process," on page 1.

For this release, the following changes have been made to the *Implementation Guide for WebSphere InterChange Server*:

- Chapter 2, "Preparing environments," on page 13 has been added to make recommendations on how to set up environments.
- Chapter 4, "The IBM WebSphere InterChange Server tools and environment," on page 31 has been added to document System Manager and the deployment procedure.
- Chapter 5, "Configuring WebSphere InterChange Server," on page 97 has been added to document the two tools that are available to configure InterChange Server.
- Chapter 6, "Using repos_copy," on page 113 has been added to document repos_copy and some common usage scenarios.
- Chapter 11, "Using Test Connector," on page 201 has been added to document the use of Visual Test Connector.
- Chapter 14, "Performance tuning," on page 293 has been added to centralize documentation on some of the features and configurable properties in the WebSphere InterChange Server system that can affect performance. It is not a comprehensive reference, but is designed to be a foundation for future information.

New in release 4.1.1

For the IBM CrossWorlds 4.1.1 release, InterChange Server, its tools, and its associated APIs have been internationalized. Internationalization (I18N) allows the IBM CrossWorlds system to support locales, which bring together information about cultural conventions particular to a particular country, territory, and language, as well as a character encoding. The IBM CrossWorlds components can then be localized to support a particular locale and its native encoding.

This manual documents the following internationalized components:

- InterChange Server (ICS)

  The system configuration file provides a new parameter, LOCALE, to specify the locale that ICS uses. If LOCALE is not set, ICS uses the locale of the operating system.

  For non-English versions of the system configuration file, the translated system configuration file has the name: `InterchangeSystem_<locale_name>.cfg` (where locale_name is the appropriate locale name).

  ICS generates its error, tracing, and status messages in the locale's character encoding. For non-English versions of the system message file, the translated system message file has the name: `InterchangeSystem_<locale_name>.txt` (where locale_name is the appropriate locale name).

- Connector Designer

  A new connector configuration property, Locale, specifies the locale that the connector framework uses. If Locale is not set, the connector framework uses the locale of the operating system.

  The CharacterEncoding connector configuration property specifies the character encoding that the connector framework uses.

Most connector configuration properties can now support the native encoding for the property values. However, the names of connector configuration properties use only English (ASCII) characters.

New in release 4.1.0

Only editorial changes have been made in the *Implementation Guide for WebSphere InterChange Server* since the 4.0.0 release.

New in release 4.0.0

The version of the *Implementation Guide for WebSphere InterChange Server* for CrossWorlds 4.0.0 is the first release of the document.

Chapter 1. The IBM WebSphere InterChange Server development model and process

The WebSphere InterChange Server system includes a core infrastructure (the InterChanger Server and business integration toolset) and configurable, modular elements work together to integrate applications and automate business processes. These modular elements—referred to in this guide as **integration components**—include connectors, collaboration templates, collaboration objects, business objects, database connection pools, maps, and relationships.

This chapter provides an overview of WebSphere InterChange Server system implementation and contains the following sections:

WebSphere InterChange Server business integration system

This section defines some of the integration concepts that are referenced in subsequent sections.

Integration components

The IBM WebSphere InterChange Server business integration system includes a core infrastructure (InterChange Server and the toolset), which support the development and execution of configurable, modular **integration components**. The integration components fulfill different roles and you develop them to work together. For example, connectors communicate with applications, business objects hold data, and collaboration templates define business process logic.

Interfaces

Typically you develop integration components in the context of an **interface**. An interface is a collection of integration components that work together to automate a business process. For instance, one interface might synchronize employee records between PeopleSoft and SAP. Another interface might automate order processing between Siebel and Oracle. Yet another interface might synchronize customer records between Siebel and SAP. An interface typically centers around a collaboration object, which is an instance of a collaboration template, with its ports bound to the components that are appropriate for the interface.

Solutions

You might also develop integration components in the context of a **solution**. Solutions are collections of components and sometimes of interfaces as well that are designed to address broad business process needs, typically across an entire industry (such as retail or automotive).

Integration component libraries

Interfaces and solutions are conceptual groupings of integration components. That is, the components are grouped in the sense that they serve a particular purpose in the context of the interface or solution.

Integration component libraries, on the other hand, are groupings of the components in your development environment. You define a library in System Manager and System Manager creates a directory in the file system that represents the library. Within the library directory is a number of subdirectories for each type of integration component. When you create an integration component it is created as a file or group of files and stored in the proper subdirectory within the directory of the library.

Libraries might contain components that do not functionally belong to the interface you are working on, but which belong to another interface that another developer is working on in the business integration system. Libraries allow you to store all the component definitions involved in all the interfaces being developed at the customer site as part of the business integration system. You can even have libraries that correspond to systems at sites you no longer work at. Although there are no restrictions on how you create and use libraries, it is recommended that you maintain a one-to-one correlation between integration component libraries and InterChange Server instances.

User projects

User projects are the software implementations that support interfaces and solutions in the WebSphere InterChange Server toolset. An interface is a conceptual structure; you group components into the idea of an interface because according to your design the involved components are needed to automate that business process. An integration component library is not a conceptual structure—you create and work with it in the toolset—but it does not have to be particularly orderly; as described in “Integration component libraries” on page 1, a library can contain definitions for components that do not belong to the interface you are working on.

User projects are designed to support the need to organize integration components together so that they are viewed as belonging to an interface. User projects are collections of shortcuts to integration components in one or more libraries, so you can create one user project for each interface. Each user project might have shortcuts to some of the same components in a library because interfaces frequently share components. A customer synchronization interface and an order processing interface that both involved the SAP application, for instance, would both require the adapter for SAP, so the user projects that correspond to those interfaces would both have shortcuts to the definition for the SAP connector.

Business integration systems

Although interfaces and solutions can satisfy business requirements themselves, they are in turn typically developed in the context of a **business integration system**. A business integration system is a collection of the interfaces and solutions required to satisfy the business integration and automation requirements of a customer site. The development of a business integration system might occur in phases, where the interfaces are grouped by priority and developed and pushed into production by priority so that more pressing needs are met while maintaining realistic goals and schedules.

InterChange Server instances

WebSphere InterChange Server is the software infrastructure that provides business process integration and automation. When you install InterChange Server on a computer and start the server, you start a server **instance**. A server instance hosts the business integration system that is comprised of all the interfaces that satisfy the integration requirements of the customer. As described in “WebSphere

InterChange Server development model," each developer on the project should maintain their own local instance of InterChange Server. Additionally, there should at least be instances of InterChange Server dedicated to testing the business integration system as a whole and to hosting the production release of the system.

WebSphere InterChange Server development model

This section describes the development model you follow in order to develop a business integration system.

The life cycle of an interface

In most cases it makes sense for a single individual to develop an entire interface. You will follow the stages of implementation as described on page 7 to develop the individual integration components that will work together to satisfy the requirements of your interface.

Developing the interface locally

When you develop integration components you use the WebSphere InterChange Server toolset.

You first prepare your development environment. This involves installing WebSphere InterChange Server on your local system. For more information, see Chapter 2, "Preparing environments," on page 13.

Next you define an integration component library to store your component definitions and a user project to represent your interface. You then create and store the integration components for your interface in the library and add shortcuts for them to the user project as you do so. For more information, see "Working with integration component libraries" on page 45 and "Working with user projects" on page 47.

Deploying the interface locally

While you are developing the integration components they only exist in your local file system. When the interface is complete you deploy the user project to your local InterChange Server instance, where the component definitions are stored in the repository. Although prior to release 4.2.0 of InterChange Server you worked with component definitions directly in the repository, you now work exclusively with component definitions in the local file system and then deploy them to the server. This development model is called "static metadata administration". It separates development activities from runtime activities and reflects the J2EE development model. For more information on deploying user projects, see "Exporting components to a package using System Manager" on page 63.

Testing the interface locally

After you have deployed an interface to your local InterChange Server instance, you should test the interface to verify that it meets the customer requirements. For more information on testing, see Chapter 11, "Using Test Connector," on page 201 and Chapter 12, "Using Integrated Test Environment," on page 215.

After you have validated the integrity of your interface in your local InterChange Server instance, you must coordinate with the other developers on the project to combine it with their interfaces as a business integration system. For more information, see "The lifestyle of a business integration system" on page 4.

The lifestyle of a business integration system

"The life cycle of an interface" on page 3 describes how you develop an interface on your local system. This section describes how you work with other developers to deliver a business integration system.

Deploying to a development integration environment

When you finish developing and testing the interface on your local system, you must coordinate with the other developers who are also working on interfaces in the project to create a business integration system. Each developer must deploy their interface to a common environment—frequently called a development integration environment.

To do this, you create an integration component library in the development integration environment, and then each developer exports a package that contains the components in their interface and imports the package into the new library.

After all redundancies have been resolved and modifications have been made, you deploy the components that represent the collection of interfaces to the local server instance in the development integration environment.

Performing integration testing

When the interfaces for a business integration system (or a particular phase of the implementation of the system) are combined, they should be tested again. This ensures that each interface still satisfies its business requirements as defined by the customer when executing along with other interfaces. If there were any problems or oversights in the migration from local development environments to the integrated development environment—such as failing to properly update a shared component—then they will be identified through rigorous testing at this stage.

Depending on the site, the integration testing might either be done in the development integration server or in a dedicated testing environment. If you maintain a separate test environment then you can combine all the interfaces in the development integration environment and validate its integrity before altering the test environment. This way, tests can be performed in the test environment for any existing interfaces there while new interfaces are prepared in the development integration environment. This approach does, however, require that the customer purchase the equipment necessary to support another environment, and that the environment be configured.

Performing performance-testing

After you have performed integration testing to verify that the collection of interfaces still satisfy their functional requirements, you might perform stress-testing to ensure that the business integration system can meet the customer's performance requirements.

If you configure the performance-testing environment to be identical to the production environment then the simulated throughput will be a more accurate prediction of what the production performance will be like. Production environments can be expensive, however, and the customer might not make the investment to create a performance-testing environment that mirrors production closely.

To migrate the business integration system into the performance-testing environment you follow the same procedure that you did when migrating into the development integration environment:

- Export a package that contains the components in the business integration system.
- Create an integration component library in the stress-testing environment.
- Import the package into the new library.
- Update the component definitions as necessary for environment-specific properties.
- Deploy the package of modified components into the stress-test server repository.

If you find that performance is not satisfactory then some components might need modification or re-configuration. For more information on some configurations and techniques that affect performance, see Chapter 14, “Performance tuning,” on page 293.

Going into production

After you have validated that the business integration system satisfies the functional and performance requirements, you migrate the collection of interfaces into a production environment.

To migrate the business integration system into the production environment you follow the same procedure as when migrating from previous environments:

- Export a package that contains the components in the business integration system.
- Create an integration component library in the production environment.
- Import the package into the new library.
- Update the component definitions as necessary for environment-specific properties.
- Deploy the package of modified components into the production server repository.

Figure 1 on page 6 illustrates the flow of integration components from integration component library to user project to server repository to exported package across the typical environments.

The development team must perform a knowledge transfer to equip the site administrator with the skills and information they need to manage the collection of interfaces. You are then either done with the integration implementation, or begin development of a new interface as part of the next phase.

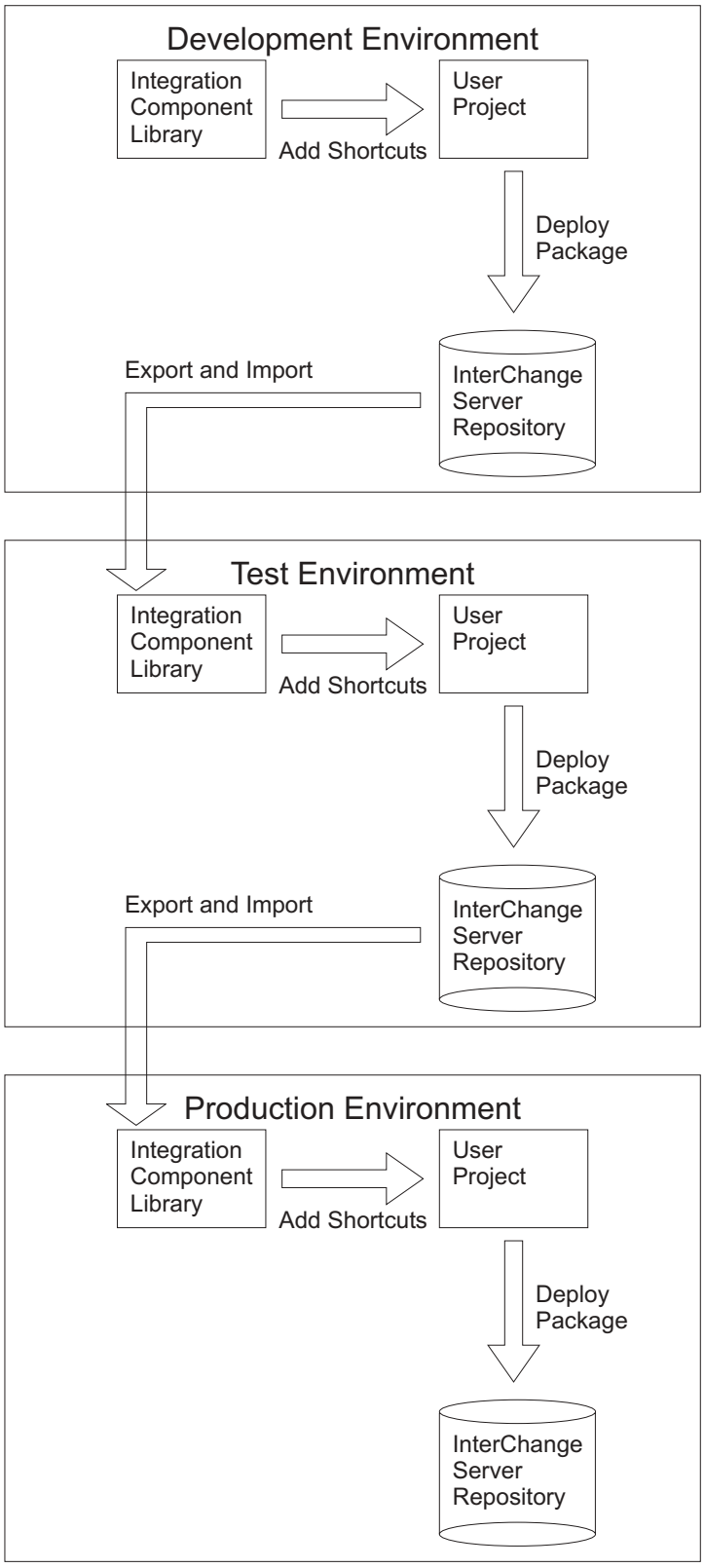

Figure 1. WebSphere InterChange Server development model

Stages of an implementation

It is recommended that you approach the development of a business integration system in stages. The exact details and the nature and timing of deliverables produced in each stage may vary according to the organization that is performing the implementation. Typically, however, the development of a business integration system involves the following stages:

Discovery and assessment of requirements

This stage begins the implementation process by identifying the business goals for the project, the system requirements, and the overall scope of the development effort.

Discovery starts at a high level and proceeds to lower levels of detail. It should begin with the following high-level questions:

- What is/are the specific business problem(s) that must be solved for the customer?

  The answer to this question is important, because it establishes the functional requirements for interfaces during the testing stages. If testing reveals that the business problem has not been solved then the interfaces were not designed or developed properly.
- What enterprise-level business processes need to be integrated or automated to address the business problem?

  Ask the following questions and others that might be relevant:
  - What are the names and specific versions of the applications that require integration?
  - Which are the source applications?
  - Which are the destination applications?
  - Which application is the system of record?
  - Are the applications in production or are they still being implemented?
  - Were the applications developed by a software vendor or by developers employed by the customer? If the applications are custom products of developers at the customer site, are those developers available for input?
- What is the technology environment—including applications, databases, and APIs—in which the business processes need to be integrated?

  Determine the characteristics of the technology environment. Examine each of the following:
  - Database vendor and version
  - Platform, operating system, and version
  - APIs that exist for the applications
  - Location of all the application client and server platforms
  - Network environment
  - Anticipated transaction volume

To identify the interfaces needed for the implementation and the components that will be used, you must research information in lower levels of detail, identifying and describing the specific business processes that you intend to implement, the business logic and data transformations that are required, and details of the applications and databases that will interact. Your research may include the following information-gathering tasks:

- Identify and describe business processes that need to be integrated or automated in order to solve the business problem. Ask the following questions:
  - What is the current series of automated and manual tasks involved in the business process?
  - What event initiates the process?
  - Who are the people involved in the business process?
  - Who are the people affected by the business process (the end-users)?
  - What are the inputs and outputs?
  - Are there prerequisites or dependencies for the data?
  - Are there filtering requirements?
  - If there are multiple destination applications, what determines how the entity being processed must be routed?
  - Is the interface bi-directional?
  - How frequently does the process occur?
  - Is there a time frame in which the transaction process needs to complete? Do other processes depend upon the entities processed by it and its successful completion?
  - What is the volume of data?
  - Must the business process occur in real-time or batch?
  - Is the interface synchronous (where the initiating program requires a response) or asynchronous?
  - What is desired procedure for responding to errors?
- Describe the structure of the entities that are processed by the interface.
- Identify data transformations that must be performed between the source and destination application entity structures.
- Illustrate the business process flow with a flow diagram (using a tool such as Holosofx). The flow diagrams will help you analyze the functions that need to be performed, and later can be useful for comparisons with existing collaboration templates. This will help you determine which collaboration to use and the level of modification that may be needed.

This stage is not described extensively in the IBM WebSphere InterChange Server documentation. Although some questions were presented in this section, the process of requirements elicitation is very specific to different consulting organizations. You should follow whatever method is appropriate. Other stages are less dependent on specific consulting practices and more dependent on the software.

Preparation of environments

In this stage, you prepare the environments that are required to provide the customer with a business process integration solution.

Each developer typically prepares their own development environment to develop their interface.

It is recommended that a system architect prepare and manage the development integration, stress-testing, and production environments.

For more information on preparing environments, see Chapter 2, “Preparing environments,” on page 13.

Design

Evaluation and design are dependent upon the detailed information gathered during discovery.

When you have determined the detailed requirements of an interface and the integration components that it comprises, you are ready to evaluate existing components to see if any will meet your needs. You may find that for some requirements, components already exist and can be used as is, that for other requirements existing components need to be extended (revised according to your needs), and that for other requirements you will need to create new (custom) integration components.

Evaluate each component both individually and in terms of how it relates to other components in the overall interface. You cannot complete the design of one component until you have also begun the design of the components with which it interacts in the interface.

For detailed information about designing components, see the following guides:

- *Connector Development Guide for Java or C++*
- *Business Object Development Guide*
- *Map Development Guide*
- *Collaboration Development Guide*

Development and configuration

In this stage you develop integration components according to the specifications that were derived in the “Design” stage.

To develop the interface, you modify existing components that are available, such as collaboration templates and business objects, and create new components that are environment-specific (such as maps).

The development and configuration stage is iterative and you may have to re-develop components and modify configurations as you progress.

It is recommended that you unit test components as you develop them to ensure that they satisfy their roles in the interface as you have designed it. When you have developed and unit-tested all of the components in the interface, you perform string tests to validate that the interface as a whole works as designed (at least in your local development environment).

Development and configuration is typically performed in a prescribed sequence of steps, as described in Chapter 3, “Developing business process interfaces,” on page 23.

For detailed information about developing components, see the following guides:

- Chapter 3, “Developing business process interfaces,” on page 23
- Adapter guides in the WebSphere Business Integration Adapters software release

- *Connector Development Guide for Java or C++*
- *Business Object Development Guide*
- *Map Development Guide*
- *Collaboration Development Guide*
- Chapter 7, “Configuring connectors,” on page 125
- Chapter 9, “Configuring collaboration objects,” on page 165

Validation

During the validation stage, you perform system testing in a controlled test environment to ensure that all the interfaces satisfy the requirements produced during the Discovery and assessment of requirements stage, as they have been developed. You perform functional, performance, and regression testing as required:

- **Functional Testing**

  You perform functional testing to ensure that the interfaces in the business integration system satisfy the business process automation and integration goals of the project.
- **Performance Testing**

  You perform performance testing to ensure that the throughput, response time, and latency are within the expectations of the customer.
- **Regression Testing**

  If you modify any components as a result of the functional or performance testing (for instance, you modify a collaboration template that did not satisfy the business requirement, or modify a connector that did not perform adequately), then you must perform regression testing to ensure that the changes made did not cause the component to violate requirements in some new way.

During the validation stage you do the following:

- Create a requirements matrix that documents all functional and performance requirements.
- Prepare a system testing environment.
- Identify and develop a series of test cases that will address the functional and performance requirements.
- Migrate the business integration system from the development integration environment to the test environment.
- Develop, coordinate, and obtain client concurrence for system testing plans and procedures. Assign specific tests for each test matrix item.
- Execute the functional and performance tests described in the test matrices.
- Identify, document, and resolve defects or inadequacies revealed by the testing.
- Perform regression testing as required.
- Create and publish a report about the testing and subsequent configuration or re-development.
- Obtain client acceptance of the developed system.

For more information on testing components and interfaces, see the following:

- Chapter 11, “Using Test Connector,” on page 201
- *Map Development Guide*

Deployment

The objective of the deployment stage is to migrate the business integration system into the production environment and start the production InterChange Server instance.

During the deployment stage you do the following:

- Develop and obtain client acceptance for an overall deployment plan, ensuring uninterrupted continuing operations with the existing systems and processes.
- Migrate the developed and tested system from the test environment to the production environment.
- Obtain and document client-specific business process and data conversion requirements for systems integration.
- Develop, implement and test modifications required to support unique production environment requirements such as email addresses, application database resources, and passwords.
- Install and test the business integration system in the production environment.
- Modify components and the production environment to optimize performance for that specific environment.
- Obtain client acceptance of the deployed production system.

For more information on migrating to production, see the following:

- Chapter 4, “The IBM WebSphere InterChange Server tools and environment,” on page 31
- *System Administration Guide*

Chapter 2. Preparing environments

This chapter describes how to prepare the various environments that are required to take an interface through development, testing, and production stages.

This chapter refers to chapters that appear later in this guide, which describe some of the tools you need to use to prepare an environment. It is placed at this point in the guide, however, because you typically prepare the environments at a site after doing the initial design and before beginning any development, and this guide is designed to reflect the life cycle of an implementation. It is recommended that you read the guide through one time and then revisit this chapter.

This chapter contains the following sections:

Preparing your individual development environment

As described in "The life cycle of an interface" on page 3, you will typically be responsible for developing a single business process interface, and will do so in a development environment you have prepared on a computer only you use, such as a laptop. Although the *System Installation Guide for Windows* contains most of the information you need to install and configure the required software on your computer, this section provides some suggestions for modifying your environment for more efficient and effective development.

Installing required software

Although IBM WebSphere InterChange Server requires a number of enterprise-class softwares to provide the integration infrastructure, you can typically run all the required software on a workstation or laptop-quality computer, provided that you do not expect to process a significant number of transactions through the system. Such a computer is adequate to support the development and unit-testing of components, but cannot support a large volume of flows. The following list addresses some of the required software:

- IBM WebSphere InterChange Server

  You should install InterChange Server on your local system so that you can develop your interface.

- IBM WebSphere Business Integration Adapters

  It is recommended that you install the adapters needed to communicate with the applications in your interface so that you can verify configuration of the connector and validate connectivity to the application. For some adapters you might have to install the application client as well. Refer to the guides for the adapters required for your interface for information on what other installation requirements they have.

- IBM Java Object Request Broker (ORB)

  You must install the IBM ORB so that clients such as the tools and connectors can communicate with InterChange Server.

- WebSphere Workbench (the Eclipse-based framework delivered with IBM WebSphere InterChange Server, and installed if you choose to install the toolset when installing InterChange Server)

You can use WSWB or WSADIE to support the System Manager and Integrated Test Environment plug-ins. It is recommended that you use WSWB because it provides all the necessary support for the plug-in, and is not as expensive on resources as WSADIE.

- WebSphere Application Server or Jakarta Tomcat

  Although System Monitor is designed primarily for administrators of a production system, it has many functions for managing the state of components that are useful in some development situations. You will probably find Tomcat to be less demanding on system resources.

- Database server

  Each developer on a project will have their own development environment usually, and require their own repository database. Although you can have a database created for each developer on a single database server host at the site, it is recommended that you install a database server on your computer and host your repository on it. This will permit you to continue development when you do not have access to the database server at the site.

  Because you develop all components in the local file system, you can still develop when you do not have access to a server repository. You cannot, however, test the components without a repository, and it is recommended that you test frequently, so it is best if you do have a repository available.

- WebSphere MQ

  Although WebSphere MQ is recommended for production environments because it provides efficient persistence of data, it is not necessary for a development environment and can be expensive on system resources. It is recommended that you use IIOP instead of WebSphere MQ in your individual development environment. To do this, use Connector Configurator to set the **DeliveryTransport** property of your connectors to the value `IDL`. For more information on Connector Configurator and connector standard properties, see Chapter 7, “Configuring connectors,” on page 125.

Installation recommendations

Consider the following suggestions when you install the required software:

- When you install IBM WebSphere InterChange Server you specify a name for the server instance. The server instance must be unique within a network because the server is exposed as a named CORBA object that clients (such as tools and connectors) must communicate with. If there are multiple objects with the same name in a network then the clients will not be able to locate the appropriate server. If all of the developers at a site accept the default server name when installing on their local machines then there will be many duplicate server names.

  It is recommended, therefore, that you specify the name of your computer as the name of your InterChange Server instance during installation, because computers must have unique names within a network as well.

  Furthermore, you might find it useful to include other information in the server name as well. For instance, if you regularly work with both the InterChange Server and WebSphere MQ Integrator Broker types of brokers, you might want to include `WICS` in the server name to distinguish the type of broker. You might want to include the numbers `420` to indicate the release version if you install multiple server instances from different releases on your system. You might include `DEV` to indicate that the server is a development server, rather than a test or production-environment server.

- You might find it helpful to install the product into a directory that has the same name as the name you specified for the InterChange Server instance. This can help you distinguish one installation from another if you install the product multiple times on the same system.
- You also might find it helpful to include the name of your InterChange Server instance in the name of the program group created in the Start menu during installation.
- Many of the softwares that you must install add entries to the PATH and CLASSPATH environment variables. These variables have a 512 character limit on Windows systems, and it is possible that you will run out of space for them, which would leave one or more programs unable to start. To overcome this limitation, you can share the folder of an application and then map a network drive to it, then replace PATH and CLASSPATH entries to the original installation directory with the mapped network drive.

  For instance, if you install InterChange Server to the following directory:

  ```
  D:\ProgramFiles\IBM\WICSInstallations\WebSphereICS420DEV
  ```

  you can share the `WICSInstallations` directory, map a network drive such as `F:` to it, and then be able to use `F:\WebSphereICS420DEV` in the PATH and CLASSPATH entries.

Working with InterChange Server in design mode

It is recommended that you start your local InterChange Server in design mode, which will enable you to incrementally assemble your integration components in a way that running in production mode would not. For more information, see “InterChange Server modes” on page 41.

Modifying batch files

It is recommended that you make the following changes to the `start_server.bat` batch file in your local development environment:

- Add the `pause` command to the blank line above the `endlocal` command at the end of the batch file.

  By default, the `start_server.bat` file is written so that it exits immediately if there is an error when InterChange Server starts. Oftentimes the server does experience an error when you first try to start it, and the console window with the error message that identifies the problem exits before you have a chance to read the message. If you add the `pause` command to the batch file then the console window will remain with the error message after an error that prevents InterChange Server from start. The end of the file will resemble the example below when you have made the change:

  ```
  %CWJAVA% -Djava.ext.dirs=%JRE_EXT_DIRS%;"%MQ_LIB%";"%DB2_LIB%"
  -Duser.home="%CROSSWORLDS%" -mx%CW_MEM_HEAP%m -DTEAgent=1200
  -DCW_MEMORY_MAX=%CW_MEM_HEAP% %ORB_PROPERTY% -classpath %JCLASSES%
  ServerWrapper -s%SERVERNAME% %2 %3

  pause

  endlocal
  ```

- Although prerequisite software such as the database server, WebSphere MQ, and IBM ORB must be running for clients to communicate with the server, you do not always have to have the server running, and it can be inefficient having those dependency software applications running when they are not needed. It is recommended that you configure the services for the third-party software applications as manual services and add `net start` commands for them to the

batch files for your server. For instance, you can add the following commands after the initial `setlocal` command in the `start_server.bat` batch file to start the WebSphere MQ and DB2 services before the rest of the batch file executes:

```
setlocal
net start "IBM MQSeries"
net start "DB2 - DB2-0"
net start "DB2 License Server"
net start "DB2 Remote Command Server"
net start "DB2 Security Server"
net start "DB2DAS - DB2DAS00"
```

If you use WebSphere Application Server to host System Monitor, you can create a batch file to start its service and required services, as well as to automatically launch the browser with URL for monitoring application, as in the following example:

```
net start "DB2 - DB2-0"
net start "DB2 License Server"
net start "DB2 Remote Command Server"
net start "DB2 Security Server"
net start "DB2DAS - DB2DAS00"
net start "IBM HTTP Administration"
net start "IBM HTTP Server"
net start "IBM WS AdminServer 4.0"
start iexplore http://devserver/ICSMonitor
```

Note: Be sure to replace any service names or URLs with what is appropriate for your environment.

Configuring logging in a development environment

It is recommended that you direct the server logging output both to the console and to a file in your development environment for the following reasons:

- It is most helpful to have the output directed to the console because you can "mark" the console by clicking it with your mouse. This causes the process to pause. You can use this to send a flow through the system and watch the console for signs of an error, then mark the console so that the process pauses before the error text is pushed off screen by more logging statements.

  Note: Be sure that you "un-mark" the console after you have researched an error. It is not uncommon to debug an interface and mark the console to examine an error, but then forget to un-mark it. The InterChange Server process remains paused until it is un-marked, and you will be unable to compile components in the repository or perform deployment actions during this time. If you find such a process hanging, seemingly without cause, check to see if the InterChange Server console is marked.
- It is helpful to have the output directed to a log file as well as to the console because if you need to consult with technical support you will want the relevant error information. If the error text is only in the console and it exits abnormally then you will not have the information technical support requires to troubleshoot the problem.

For information on how to configure InterChange Server logging and tracing, see Chapter 5, "Configuring WebSphere InterChange Server," on page 97.

Customizing your environment

Most developers work more efficiently when they customize their environment as described in the sections below.

Organizing your shortcuts

Most developers find it helpful to organize the shortcuts they use frequently to avoid having to use the program menu frequently. You might create a folder for the shortcuts on the desktop, or create a custom toolbar in the Windows taskbar. It is most helpful to include shortcuts to InterChange Server, any connectors that are needed, WSWB (System Manager), the database server administration tool, the WebSphere MQ console, System Monitor, a text editor, and so forth.

Modifying the properties of shortcuts

By default InterChange Server and connectors run in a console window with a small size, small screen buffer, and color scheme that is not conducive to reading. Do the following to modify their shortcuts:

1. Right-click your InterChange Server or connector shortcut and choose **Properties** from the context menu.
2. Click the **Layout** tab and make the following changes:
   - Set the **Height** field in the "Screen buffer size" pane to the value `9999`.

     This will cause the console to keep more information cached so that error messages are not replaced as quickly when the system is processing events.
   - Set the **Width** field to the value `120` and the **Height** field to the value `25` in the "Window size" pane.

     This will size the console in a way so that more information can be displayed horizontally and the console takes up less space vertically. You can usually fit three consoles with a height of `25` on a single screen, which makes it easy to tile console windows for the InterChange Server and the two connectors you typically need to test with for an interface.
3. Click the **Colors** tab and make the following changes:
   - Select the **Screen background** radio button and then select a color for it.
   - Select the **Screen text** radio button and then select a color for it.

     Note: It is most useful to use dark colors for the screen background and light colors for the screen text. Use different color schemes for the InterChange Server and connector consoles so that you can easily distinguish them from one another.
4. Click **OK**.

Configuring tools preferences

You should configure your tools preferences as described in "Using Eclipse-based workbenches" on page 85.

Setting your workspace

It is recommended that you set your workspace—the directory where projects you create in the workbench are stored by default—to a location that best suits your needs. Do the following to specify your workspace directory:

1. Navigate to the `bin` directory of the product directory.
2. Edit the file named **startcsm.bat**.
3. Set the value after the `-data` option to the directory you want to be your default workspace. For instance, to set your workspace to a directory named `Projects` within the WebSphere InterChange Server product directory, the file would appear as below:

```
setlocal
set WSWB_EXECUTABLE=%1
"%WSWB_EXECUTABLE%" -data "%CROSSWORLDS%"/Projects -vmargs
```

```
-Xbootclasspath/p:"%CROSSWORLDS%"\lib\vbjorb.jar
-Dorg.omg.CORBA.ORBClass=com.inprise.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.inprise.vbroker.orb.ORBSingleton
-DCWTools.home="%CROSSWORLDS%"\bin
```

4. Save and close the file.

Creating your integration component libraries

It is recommended that you create the following integration component libraries:

- One for each InterChange Server instance with which you have to interact. You should have a one-to-one correlation between libraries and server instances, so that your local development environment mirrors the repositories you deal with.

  It is recommended that you name this type of library in such a way as to associate it with the server it corresponds to. For instance, if the server name is `WICS420DEV`, you might name the library `WICS420DEVICL`. A similar convention is recommended for user projects in "Creating your user projects."
- A "staging" library. You might use this library as a container for components that do not explicitly belong to a server environment. The contents might include samples, demo materials, components that have been partially developed, and components that you frequently re-use across multiple libraries.

Creating your user projects

It is recommended that you create the following integration component libraries:

- One for each integration component library you have created. Since it is recommended that you maintain a one-to-one correlation between libraries and server, and you must have a user project to deploy components from a library to a server, it makes sense to dedicate one user project to each library so that you can deploy the entire library to the server if necessary.

  It is recommended that you name this type of user project in such a way as to associate it with the server it corresponds to. For instance, if the server name is `WICS420DEV`, you might name the user project `WICS420DEVUP`. A similar convention is recommended for integration component libraries in "Creating your integration component libraries."
- One library for each interface you are developing. This is the best use of a user project—if you create shortcuts in the user project to only the components that are required for the interface then you have a very convenient to use view of the subset of components in a library.

Configuring connectors

You might install and run connectors in your local development environment to do some basic testing of your interface. Follow the recommendations in these sections to use connectors as best as possible in this context.

Using IDL as the transport protocol

As described in "Installing required software" on page 13, it can be running WebSphere MQ in your local development environment is not necessary and can be costly on performance. Instead, set the **DeliveryTransport** property of your connector definitions to the value `IDL` to use IIOP instead.

Using key polling

If you have a connector that is responsible for event notification, it must poll an event table to detect new events. The polling is necessary, but the messages written to the console window of the connector can push important troubleshooting information off screen. You can use key polling so that the connector only issues a

poll call when you want it to. For more information on key polling and on configuring connectors, see Chapter 7, “Configuring connectors,” on page 125.

Preparing a development integration environment

As described in “Deploying to a development integration environment” on page 4, you will deploy the interfaces you develop to a centralized environment along with the other developers on the project. This environment might be entirely dedicated to coordinating interfaces, making sure that all of the cross-interface dependencies have been resolved, and modifying shared components, or it might be the testing environment, depending on the resources available and the complexity of the system.

It is recommended that one developer be appointed as a technical manager for the process of combining all the interfaces in the development integration environment. Although map definitions and collaboration objects are typically specific to a single interface, business objects, connectors, relationships, and collaboration templates are oftentimes shared by multiple interfaces. The technical lead should make sure that when the interfaces are combined together that all the shared components will still function as necessary for all the interfaces. For business objects and collaboration templates it is not difficult to do this as long as all the developers on the project communicate well throughout its course.

If one interface requires a modification to a shared component then the change should be made by the developer of the interface that required the change, and then the other developers whose interfaces share the component should import the changed version into their libraries before their interfaces are finished. The project documentation for the component and the affected interfaces should be updated to reflect the modification.

Some components, such as connectors and relationships, commonly need modifications when interfaces are combined in a development integration environment. For instance, an integration system might have a customer synchronization interface and an order processing interface, both of which involve SAP. The interfaces would typically be developed by different developers in their own environments. In each developer’s environment, the SAP connector definition would only have the configurations necessary to support that developer’s interface. The developer of the customer synchronization interface would not have the business objects and other components needed for the order processing interface because those components are still being developed. When the developers deploy their interfaces to the development integration environment, the adapter for SAP must be modified so that it supports the business objects and maps for both interfaces. The technical lead must ensure that the shared components are updated properly if necessary.

You should maintain thorough and accurate project documentation to track what components need to be changed whenever a new interface is deployed to the environment.

Using source code control

There are plug-ins available for the workbench tools that allow you to work with integration components under source code control. By implementing source code control, all the developers on a project can add an integration component library (or user project) to their workspace and work on the components in the library without risk of other developers making changes to the components.

Note: Although source code control can prevent developers from making changes to the same component definition, it cannot prevent developers from deploying components to the server. To prevent multiple developers from overwriting components in the server, have one person in charge of deployment and change the InterChange Server password.

Using Rational ClearCase

This section provides instructions on how to enable the ClearCase plug-ins and enable System Manager for ClearCase connectivity. It also provides some information on how ClearCase might be used, though you should consult the documentation for ClearCase and for the ClearCase plug-in for detailed information. Furthermore, this section does not cover prerequisite steps such as installing the ClearCase client, creating a view, and so forth.

Confirming the plug-ins are installed and enabled

Do the following to confirm that the required plug-ins are installed and enabled in your workbench:

1. Navigate to the \plugins directory of your Eclipse-based tooling framework (such as WSWB or WSADIE).
2. Look for a directory named `com.ibm.btools.csm.ui.cc` and explore it if it exists.
3. Look for a file named `plugin.xml`. If a file by that name does not exist, but one named `plugin.xml.off` does exist, then rename `plugin.xml.off` to `plugin.xml`.
4. Repeat steps 2 through 3 for the following plug-ins as well:
   - `com.rational.clearcase`
   - `com.rational.clearcase.help`

If any of these required directories is missing, or if they do not contain a file named `plugin.xml`, then you will not be able to use System Manager with ClearCase.

Configuring System Manager for ClearCase connectivity

Once you are sure the plug-ins are enabled in the file system, you must do the following to configure System Manager for ClearCase connectivity:

1. Start your workbench.
2. Select **Perspective > Close All** from the menu bar.
3. Open the System Manager perspective.
4. Select **Perspective > Customize** from the menu bar.
5. Expand the **Other** node.
6. Enable the **ClearCase** checkbox.
7. Click **OK**.

   The **ClearCase** menu is added to the menu bar.
8. Click the black down-turned arrow in the title bar of the WebSphere Business Integration System Manager view and choose **Show ClearCase decorations** from the context menu.

   Resources in the workbench will now have a context menu named **ClearCase** that you can use to perform source code control operations on them, and will display icons to indicate if they are checked out and so forth.

Connect to ClearCase

To connect System Manager to ClearCase, select **ClearCase > Connect to Rational ClearCase** from the menu bar.

Moving a project into ClearCase

For developers to access the same components, an integration component library must initially be created on one system and then moved into ClearCase. Do the following to move a library into ClearCase:

1. Right-click an integration component library and choose **ClearCase > Move Project Into ClearCase** from the context menu.
2. Navigate to the directory in your view where you want the integration component library moved to and click **OK**.

   The "Progress Information" dialog appears as the project is moved into source code control.
3. At the "Add Element(s) to Source Control" window, select the integration components you want to add with the library and click **OK**.

Adding a project to the workspace

Once the integration component library that will be shared by the developers has been moved into ClearCase by one developer, the other developers can do the following to add the project to their own workspace:

1. Select **ClearCase > Connect to Rational ClearCase** from the menu bar to connect to ClearCase.
2. Select **ClearCase > Add Project to Workspace** from the menu bar.
3. At the "ClearCase Project Location" dialog, navigate to and select the directory of the integration component library that was moved under source code control.
4. Click **OK**.
5. At the "Add Elements(s) to Source Control" dialog, click **OK**.

   The project folder is added to your workspace and you can check components in and out by right-clicking them and using the **ClearCase** context menu.

   Note: You might have to restart the workbench for the project folder to appear.

Using VCM

System Manager can also be used with VCM. For information on how to do so, see the documentation for VCM itself, and for the VCM plug-ins. To enable VCM connectivity, follow the steps described in "Confirming the plug-ins are installed and enabled" on page 20, but with the following plug-in directories instead:

- `com.ibm.btools.csm.ui.vcm`
- `org.eclipse.vcm.core`
- `org.eclipse.vcm.ui`

Chapter 3. Developing business process interfaces

This chapter describes a suggested approach for the development of the business process interfaces to be used in a WebSphere business integration system with InterChange Server. Depending on the specific circumstances at your site, you may develop multiple interfaces in parallel and may develop different interfaces at different points in the development cycle.

You should follow the sequence of steps described here for development of most interfaces. Although more iterations at any point in the sequence may be required in an actual implementation, these steps provide a framework for your overall development effort.

This chapter contains the following sections:

Overall development flow

It is recommended that you develop integration components in the following general order:

- **Connectors and data handlers**

  Connectors and application-specific business objects are the integration components that communicate directly with the applications being integrated.

  – If you need to develop a custom connector, that development work can proceed in parallel with other development efforts. The skillset required to develop a connector is different from the skillset required to develop other components so it is common to have a developer work only on connector development if that task is required. With this approach one developer can work on the connector and the rest of the development team can work on other components at the same time.

  – Collaborations and maps depend on generic objects and generic objects depend on application-specific business objects. You cannot develop application-specific business objects until you know how the connector that processes them uses their meta-data, so you must select the connector first to reduce the threat of having to redesign and develop dependent components at a later point in the implementation. Furthermore, many adapters feature Object Discovery Agents (ODAs), which can make it easy to develop application-specific business objects very efficiently.

- **Application-specific business objects**

  You should develop application-specific business objects after connectors because you must understand the connector in order to develop business objects for it. You should develop application-specific business objects before you develop generic business objects because generic business objects generally represent a superset of the application-specific business objects in an interface.

  It is recommended that you create application-specific business objects in multiple iterations. Develop the application-specific business object at first as a relatively simple structure and then test it to make sure that the connector can use the business object structure to exchange data with the application successfully. Then add a layer of complexity and re-test the business object to

make sure that the interface still works in spite of the modifications. Repeat this process until the application-specific business object is as large and complex as it has to be to satisfy the interface.

- **Generic business objects**

  After you have developed all of the application-specific business objects required for an interface you should develop the generic business object as a superset of the application-specific business objects in the interface.

  Be sure to investigate any existing generic business objects that might be provided on the installer or as part of an industry solution to avoid unnecessary custom development—you should not spend time doing work that has already been done. When you evaluate existing content for use, be sure that you understand the relevant licensing terms for the site and verify whether or not the license covers the use of the object.

  Ultimately, develop the generic business object so that it contains:

  – All of the attributes it requires to satisfy the interface in its current design
  – All of the attributes it requires to satisfy modifications or additions to the interface in the foreseeable future
  – No attributes it does not require to satisfy current or future requirements of the interface

- **Maps and relationships**

  Once you have identified or developed the business objects for an interface, you should develop the maps that transform application-specific objects into generic objects, and generic objects into application-specific objects. Use Map Designer tool to create a map, define the transformation rules, and unit test the map with a sample input generic or application-specific business object.

  After you have validated the map transformations through unit-testing, create any relationships required for the interface and then implement cross-referencing in the appropriate maps. Unit test the maps in the order in which they would execute during the execution of the interface to which they belong.

  See the *Map Development Guide* for details.

- **Collaboration templates**

  The collaboration template defines the business logic of the interface. The simplest collaborations merely route business objects between connectors. Other collaborations might also have complex interactions involving transactions, long-lived business processing, and delegation of processing to other collaboration objects. Regardless, collaborations are centered around the generic business object that represents a superset of the application-specific business objects in an interface.

  You do not need to use application-specific business objects, maps, or connectors during the development and unit testing of a collaboration. If the collaboration does not have to make significant modifications to the data of a generic business object at runtime, or otherwise perform operations that explicitly depend on the structure of the generic business object, then you do not even need the generic business object to be developed before developing the collaboration template. As long as the business process is well-defined you can develop the collaboration template and test it with simplified version of the generic business object until it is developed in whole.

  When you assess the need for collaboration templates, investigate which templates are already available, and determine if the licensing terms for the site permit the use of any available and appropriate collaboration templates. If there is no suitable existing collaboration template then you should develop a custom.

For more information, see the *Collaboration Development Guide* and documentation for the particular collaborations you plan to use.

- **Collaboration object**

  When you complete the collaboration template and all of the components with which it interfaces, you should create a collaboration object based on the template.

  You should unit test the collaboration object without the connector agents running by using Integrated Test Environment or Test Connector. This allows you to test the integrity of the interface without introducing connectivity issues and thereby rule out errors related to the map and collaboration logic. Then you should test the interface as a whole with the connector agents running.

Sequence of development tasks

Many integration components in an interface depend upon and reference one another, so the tasks of developing each of the components are intertwined and iterative.

It is recommended that you develop your integration components in the following order:

1. "Develop custom connectors"
2. "Develop or modify application-specific business objects"
3. "Configure connectors" on page 26
4. "Test connectivity and application-specific business objects" on page 26
5. "Develop or modify generic business objects" on page 28
6. "Configure relationships" on page 28
7. "Configure database connection pools" on page 28
8. "Develop and test maps" on page 28
9. "Develop and test collaboration templates" on page 29
10. "Configure connectors for business object support and associated maps" on page 29
11. "Create and configure collaboration objects" on page 29
12. "Deploy the interface" on page 30

Develop custom connectors

A developer with the required skillset should begin development of any required custom connectors as soon as the project begins.

For information on developing custom connectors, see the *Connector Development Guide for Java or C++*.

Develop or modify application-specific business objects

This topic provides a brief overview of developing application-specific business objects. For details about working with application-specific business object definitions, see the *Business Object Development Guide.*

When you create or modify an application-specific business object, do the following:

- Reference the guide for the adapter to determine if the adapter features an Object Discovery Agent (ODA) that can be used to generate application-specific business objects. ODAs can greatly expedite this stage of development.

- If the adapter does not feature an ODA, try to obtain a sample object for the application or an application-specific business object for the entity from another customer site. Be sure to not to violate any intellectual property agreements with respect to customer content.
- If the adapter does not have an ODA and there is no sample or existing business object to modify then you must develop the application-specific business object in Business Object Designer.

When you create the business object definition you should also create any source application triggers or other event detection mechanisms if necessary.

For details about designing and developing application-specific business object definitions, see the *Business Object Development Guide*.

Configure connectors

When you configure a connector for the purpose of unit testing an application-specific business object you may not yet have developed all of the business object definitions and maps that the connector requires to perform its role in the implementation. You can, however, add support for the business object definition you need to test and be able to test it successfully without those other components. When you have finished developing the other components you must then re-configure the connector definition to add support for the business object definitions and associate the required maps.

See the guide for your adapter for information about its application-specific properties. See Chapter 7, “Configuring connectors,” on page 125 for information about connector standard properties and information on how to use Connector Configurator.

Test connectivity and application-specific business objects

Once you have developed the application-specific business object and added support for it to a connector definition you should unit-test the business object to make sure that the connector can use it to exchange data with the application successfully. You do not need the generic objects, maps, or collaboration template that the interface will eventually use in order to perform this test. Do the following to unit-test an application-specific business object:

1. Create a pass-through collaboration template with the following design:
   - The collaboration template has two ports—one named `From` and one named `To`.
   - The `From` and `To` ports both support the application-specific business object definition.
   - The collaboration template has a single scenario named `Main`.
   - The `From` port is configured to be the triggering port for the `Main` scenario for all of the verbs belonging to the business object definition.
   - The `Main` scenario has the following elements:
     – A Start node that is connected to an Action node by a transition link.
     – A Service Call node that is attached to the Action node and that has the following characteristics:
       - The **Port** drop-down menu in the “Regular service call” pane is set to the `To` port.

- The **Verb** drop-down menu in the "Regular service call" pane is set to the `Create` verb (or whatever the primary verb for the business object in question is).
- The **BO Variable** field in the "Regular service call" pane is set to the value `triggeringBusObj`, which is the variable that can be used to reference the business object that triggers the collaboration.

– An End Success node that is connected to the Action node by a transition link.

For more information about creating collaboration templates, see the *Collaboration Development Guide*.

2. Configure the properties of the connector definition, add support for the business object definition, and enable the **Agent Support** checkbox for the business object definition.
3. Add support for the application-specific business object to another "dummy" connector definition. You do not have to configure this connector definition completely because you will only emulate it with testing tools. Ensure that the **Agent Support** checkbox is enabled for the business object definition.
4. If the connector is responsible for event notification, do the following:
   - Create a collaboration object based on the pass-through template with the connector to be tested bound to the `From` port and the "dummy" connector bound to the `To` port.
   - Deploy the connectors, collaboration template, collaboration object, and business object definitions to the server.

     For more information about deploying components, see "Exporting components to a package using System Manager" on page 63.
   - Start the connector agent of the connector you are testing.

     For more information about starting a connector, see your adapter guide and the *System Administration Guide*.
   - Start Test Connector, open the connector definition for the other connector to which you added support for the business object definition, and connect Test Connector to the agent.

     For more information on Test Connector, see Chapter 11, "Using Test Connector," on page 201.
   - Trigger an event in the source application.
   - If the test is successful then the connector will send a business object representing the event to the collaboration, and it will appear as a request in Test Connector.
5. If the connector is responsible for request processing, do the following:
   - Create a collaboration object based on the pass-through template with the connector to be tested bound to the `To` port and the "dummy" connector bound to the `From` port.

     For more information on creating collaboration objects, see Chapter 9, "Configuring collaboration objects," on page 165.
   - Deploy the connectors, collaboration template, collaboration object, and business object definitions to the server.

     For more information about deploying components, see "Exporting components to a package using System Manager" on page 63.
   - Start the connector agent of the connector you are testing.

     For more information about starting a connector, see your adapter guide and the *System Administration Guide*.

- Start Test Connector, open the connector definition for the other connector to which you added support for the business object definition, and connect Test Connector to the agent.

  For more information on Test Connector, see Chapter 11, "Using Test Connector," on page 201.
- Send a business object request from Test Connector.
- If the test is successful then the connector receives the business object request and processes it, creating a record for the entity in the application.

Develop or modify generic business objects

Determine if there is an existing generic object that is appropriate for the business process. Ensure that the licensing terms for the site permit the use of the object.

If no existing generic business object can be used, you will need to either extend an existing generic business object or create a new one. For information on creating new business objects, see the *Business Object Development Guide*.

Configure relationships

Configure any relationships at this point so that you can use them when developing maps in a later stage.

See the *Map Development Guide* for information about configuring relationships.

Configure database connection pools

Just as you configured any relationships required by the maps, you should do the same for any database connection pools. Configuring database connection pools beforehand allows you to code use of them into the maps as you develop them.

For information on configuring database connection pools, see Chapter 8, "Configuring database connection pools," on page 155.

Develop and test maps

With all of the required generic business objects, application-specific business objects, relationships, and database connection pools created, you can develop the maps that transform objects across the interface.

As you develop the maps you should unit-test them by using the debugging facilities of Map Designer. Unit-test your maps at the following points:

- After you have defined all the simple transformation rules, such as moves, splits, and joins.
- After you have developed the more complex transformation rules that require custom Java code.
- After you have incorporated the use of relationships and database connection pools.

  When you test a map that uses relationships, you must make sure that you test the maps in the order in which they would execute within the context of the interface. If you do not then the cross-referencing logic will not execute properly.

For more information on developing and testing maps, see the *Map Development Guide*.

Develop and test collaboration templates

Collaboration templates define the business process logic of an interface. You should research the existing collaboration templates available to determine if there is one that models your business process. If there is no collaboration that directly models the process, you should determine if there is one that is similar; if so, you may be able to modify it if the licensing conditions of the site permit that, and thereby accelerate the development process.

If there is no template that is similar then you must develop a new collaboration template. For information about how to develop a new collaboration template or modify an existing one, see the *Collaboration Development Guide*.

You can unit-test the collaboration template without using the maps, application-specific business objects, or connectors that you are developing for the interface. Do the following to unit-test a collaboration template:

1. Add support for the generic business object that is processed by the collaboration template to the `PortConnector` definition.
2. Create a collaboration object based on the collaboration template with the ports of the collaboration bound to the `PortConnector`.

   For more information on creating collaboration objects, see Chapter 9, "Configuring collaboration objects," on page 165.
3. Deploy the `PortConnector`, collaboration template, collaboration object, and business object definitions to the server.

   For more information about deploying components, see "Exporting components to a package using System Manager" on page 63.
4. Start Test Connector, open the `PortConnector` definition, and connect Test Connector to the agent.

   For more information on Test Connector, see Chapter 11, "Using Test Connector," on page 201.
5. Create an instance of the generic business object with data in it and send the event as a request from Test Connector.

   The collaboration receives and processes the event and each service call sends the generic business object to Test Connector, where you can edit it to examine the data for modification.
6. Examine the InterChange Server logging output to ensure that the collaboration logic executed as expected.

Configure connectors for business object support and associated maps

Before you can test all of the developed components together, you must reconfigure the connector definitions to add support for the application-specific and generic objects they require to participate in the interface. The system associates any maps that it can automatically, but in situations where there are multiple maps that could transform objects you must explicitly associate a map. For more information, see Chapter 7, "Configuring connectors," on page 125.

Create and configure collaboration objects

Once you have added support for the required business object definitions to the connectors you can create a collaboration object based on the template and bind its ports to the proper components.

For more information on working with collaboration objects, see Chapter 9, “Configuring collaboration objects,” on page 165.

Deploy the interface

After you have created all of the required components for the interface you must deploy it to your local InterChange Server instance to test the interface. For more information on deployment, see “Exporting components to a package using System Manager” on page 63.

Test the interface

After you have deployed the interface to your local InterChange Server instance you should test the interface as a whole to ensure that the components satisfy the business requirements when working together. Do the following to test the interface:

1. Ensure that all of the required components are active.
2. Trigger an event in the source system.
3. Examine the logging output of InterChange Server to ensure that the components in the interface execute properly.
4. Examine the destination system to ensure that the entity was processed successfully and that the data appears as it should.

Chapter 4. The IBM WebSphere InterChange Server tools and environment

This chapter introduces you to the WebSphere InterChange Server toolset you will use to develop business process integration interfaces. This chapter contains the following sections:

About System Manager, Integrated Test Environment, and Collaboration Debugger

System Manager, Integrated Test Environment, and Collaboration Debugger are plug-ins that run within the Eclipse-based tooling frameworks named WebSphere Studio Workbench and WebSphere Studio Application Developer Integration Edition. This section provides an overview of the Eclipse framework, WSWB and WSADIE, and of the IBM WebSphere InterChange Server tools plug-ins.

About the Eclipse Platform

The Eclipse Platform is an open-source integrated development environment (IDE) for the creation of tools. It provides tools developers with a development kit and runtime that enables the developer to write plug-ins that allow the user to work with a particular type of resource.

IBM has two branded versions of the Eclipse platform—WebSphere Studio WorkBench (WSWB) and WebSphere Studio Application Developer Integration Edition (WSADIE).

Plug-ins

Plug-ins are the modular extensions that software vendors develop to add functionality to an Eclipse-based workbench. Plug-ins encapsulate the perspectives, editors, and views that enable users of the workbench to work with particular types of resources.

For instance, one plug-in might provide the features of a text editor. Another plug-in might provide the features of an HTML editor. The WebSphere InterChange Server plug-ins provide the features to work with integration components. The benefit of this plug-in model is that the user has a single tool in which they can work with many types of resources, rather than using dedicated tools for each type of resource.

To install a plug-in, you extract one or more compressed archives that represents the plug-in into the `plugins` directory within the product directory of your workbench. The WICS plug-ins are extracted into the `plugins` directory by Installer.

IBM provides the System Manager, Integrated Test Environment, and Collaboration Debugger plug-ins with WebSphere InterChange Server to work with business integration resources. These plug-ins are embodied by a number of uncompressed directories in the `plugins` directory of your workbench within the `com.ibm.btools` namespace. Much of the primary interface you will use in creating integration components—the System Manager—is encompassed in the contents of the `com.ibm.btools.csm` plug-in directory, for instance.

Workbench

The workbench is the collection of perspectives, editors, and views that are active in your Eclipse-based tooling framework, which are in turn affected by the collection of plug-ins you have installed and enabled. It is a general term used to refer to the Eclipse-based interface in which you are working, independent of the fact that the interface changes depending on how you use it.

Workspace

A workspace is a container for projects. The workspace is a directory in the file system where, by default, you are prompted to store your projects.

Projects

Projects are user-defined groups of resources, and are ultimately directories in the file system.

One of your first tasks when developing a business process interface is to define an integration component library, which is a project that contains the components you develop. When you create the integration component library you specify the location in the file system where it is stored (by default this is the workspace directory). A folder is created in that location with the name you specify for the integration component library and within the library folder a number of folders is created for each type of integration component (for instance, there are folders named `Maps`, `BusinessObjects`, and `Connectors`).

You also create projects named **user projects**. User projects are collections of shortcuts that reference integration components. You must add integration components to a user project from integration component libraries in order to deploy components to an InterChange Server instance. Besides from being required to deploy components to a server, user projects are designed to allow you to functionally group components together. An integration component library is a collection of all components you might need to work with, but a user project is designed to let you group together the components you are working on for a specific interface.

Resources

Resources are projects, files, and folders that you work with in the workbench.

When you create an integration component, it is stored as a file in the appropriate folder within the integration component library project. The different types of integration components are stored with different extensions (for instance, maps are stored with an extension of `.cwm` whereas collaboration templates are stored with an extension of `.cwt`), but they are all stored in XML format.

Some components, such as maps and collaboration templates, also have Java source files in addition to their definition files.

Perspectives

A perspective is a grouping of editors and views designed to provide a particular user role with what it requires.

For instance, the System Manager perspective provides views for working with InterChange Server instances and integration component libraries and editors for collaboration object definitions and database connection pools. The Integrated Test Environment perspective has views for client emulation interfaces and business object test data, and editors for test units.

Editors

Editors allow you to open, save, and close resources in the workbench.

System Manager, for instance, has an editor in which you modify collaboration objects and another in which you configure InterChange Server.

Views

Views provide information about the resources with which you are working in the workbench.

System Manager, for example, has the WebSphere Business Integration System Manager view, which is the view to integration component libraries and user projects, and the InterChange Servers view, where you work with the InterChange Server instances you have registered.

About WSWB and WSADIE

WebSphere Studio Workbench (WSWB) is an IBM-branded release of the Eclipse platform. IBM delivers WSWB with WebSphere InterChange Server and you can install it along with the core infrastructure. WSWB is capable of running all the plug-ins necessary to develop WebSphere InterChange Server integration components.

WebSphere Studio Application Developer Integration Edition (WSADIE) is an IBM-branded release of the Eclipse platform, like WSWB, but can also be used to develop new plug-ins. WSADIE is not delivered with WebSphere InterChange Server because the ability to create new plug-ins is not required to develop integration components. If you have it installed, however, then you can use it to run the required System Manager and Integrated Test Environment plug-ins. When you install WebSphere InterChange Server you have the option to update an installation of WSADIE to install the necessary plug-ins.

About System Manager

System Manager is the perspective in which you work with the integration components and server instances in a WebSphere InterChange Server business integration system. You use System Manager primarily for the following tasks:

- Launching other tools in the WebSphere InterChange Server business integration toolset
- Developing and configuring some integration components
- Working with InterChange Server instances
- Deploying integration components to a repository

About Integrated Test Environment

Integrated Test Environment is the perspective in which you can test business integration interfaces you have developed. It provides graphical interfaces to emulate connectors, start the required components, and examine business object data. For more information on this perspective, see Chapter 12, "Using Integrated Test Environment," on page 215.

About Collaboration Debugger

Collaboration Debugger is a perspective in which you can conveniently troubleshoot collaboration logic. For more information, see Chapter 13, "Using Collaboration Debugger," on page 273.

Using System Manager

This section describes how to start and use the System Manager perspective.

Starting System Manager

When you install IBM WebSphere InterChange Server, you have the option to install the WebSphere Studio Workbench with support for the plug-ins required to interact with InterChange Server, or to install the plug-ins into an existing installation of WebSphere Studio Application Developer Integration Edition.

You must launch the workbench with several Java Virtual Machine arguments that allow the framework to communicate with InterChange Server over CORBA. To achieve this, a batch file named `startcsm.bat` is created in the `bin` directory of the WebSphere InterChange Server product directory and a shortcut to that batch file is created in the program group. For the workbench to function in a WebSphere InterChange Server environment, then, you must launch the custom shortcut instead of launching any shortcut in the workbench's own program group or instead of launching the workbench's executable directly.

To start System Manager, do the following:

1. Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Administrative > System Manager**.
2. Select **Windows > Open Perspective > Other** from the menu bar.
3. Select `System Manager` from the list of perspectives and then click **OK**.

   WebSphere Studio Workbench starts and appears. Figure 2 on page 35 shows the System Manager perspective and "System Manager interface" describes the interface and its elements.

System Manager interface

The System Manager perspective has several views and editors in the default configuration it opens with. Figure 2 on page 35 shows the System Manager perspective:

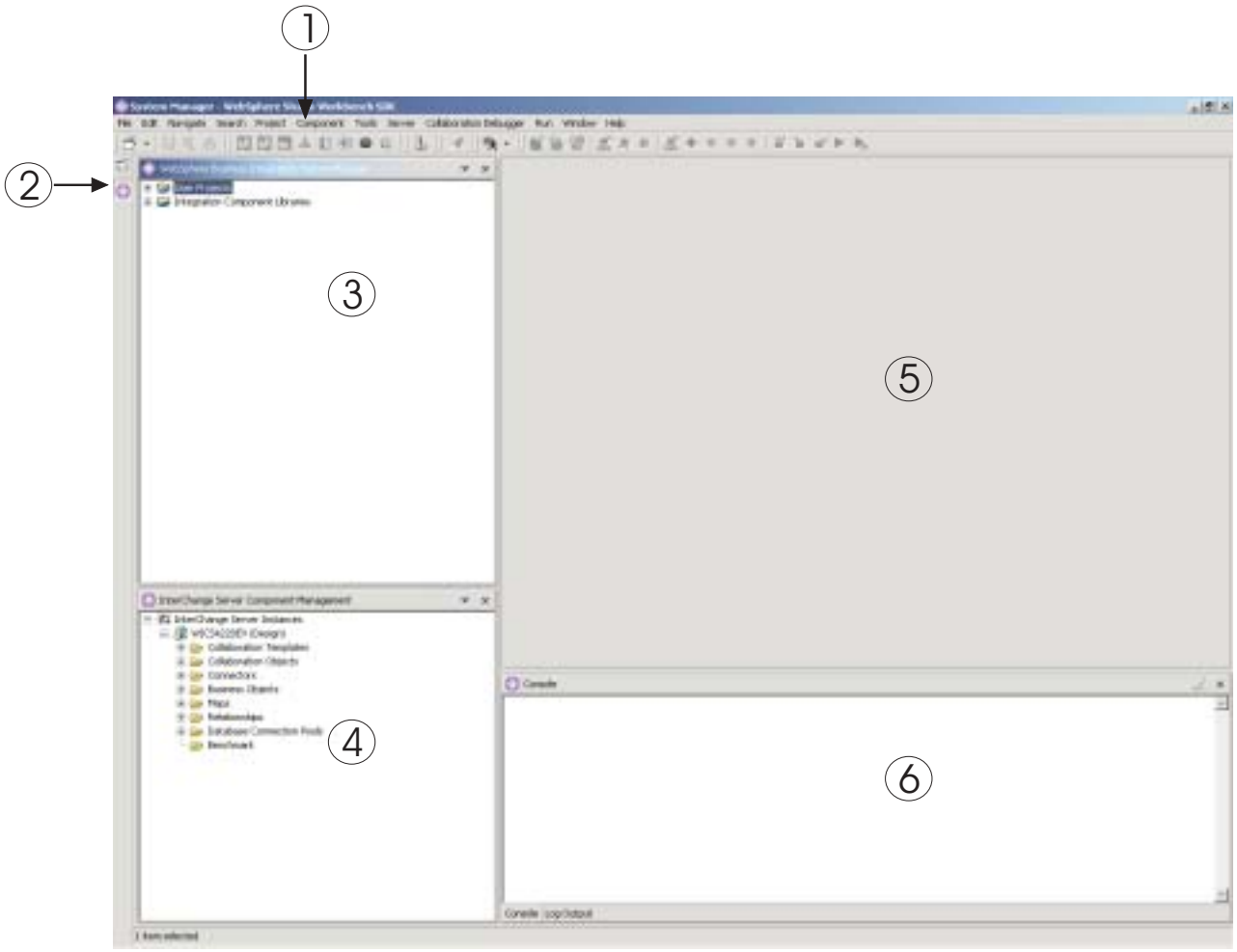

Figure 2. System Manager perspective

Table 1 describes the interface elements of the System Manager perspective, identified by the numbers in Figure 2 on page 35:

Table 1. System Manager perspective interface elements

Interface element number	Interface element name
1	"Menu bar and toolbar" on page 36
2	"Perspective shortcut bar" on page 37
3	"WebSphere Business Integration System Manager view" on page 37
4	"InterChange Server Component Management view" on page 37
5	"Editor view" on page 37
6	"Console view" on page 37

The following sections describe the interface elements of System Manager in greater detail.

Menu bar and toolbar

You use the menu bar and toolbar to work with the Eclipse-based tooling framework and to work with WebSphere InterChange Server components. Many of the menu bar items have toolbar equivalents, so the following sections only describe menu bars and their items.

File menu: This is an Eclipse-standard menu that is used to work with resources. You will use it primarily to create new integration component libraries and user projects.

For more information on creating integration component libraries and user projects, see "Working with integration component libraries" on page 45 and "Working with user projects" on page 47.

Edit menu: This is an Eclipse-standard menu that has many standard items such as Cut, Copy, and Paste.

For more information on cutting, copying, pasting, and deleting the components you create, see "Working with components in integration component libraries" on page 51.

Navigate menu: This is an Eclipse-standard menu that allows you to navigate among the resources in the workbench. For more information on this menu, see the workbench documentation.

Search menu: This is an Eclipse-standard menu that allows you to search for and search within resources. For more information on this menu, see the workbench documentation.

Project menu: This is an Eclipse-standard menu that has menu items for manipulating Project resources in the workbench; this menu item is not used when working with the System Manager perspective. For more information on this menu, see the workbench documentation.

Component menu: This menu is provided with the System Manager perspective and is useful for working with integration components you create. The items in this menu are documented throughout this guide and others in sections that describe specific tasks.

Tools menu: This menu is provided with the System Manager perspective and is used to launch tools used to create integration components.

For more information, see "Working with components in integration component libraries" on page 51.

Server menu: This menu has the **Register Servers** item, which you can use to register an InterChange Server instance.

For more information, see "Registering an InterChange Server instance" on page 38.

Run menu: This menu has an item to configure external tools to run external programs, batch files, and build scripts. For more information, see the workbench documentation.

Window menu: This menu has items for working with perspectives, views, editors, and preferences.

These items are covered in the workbench documentation and in various sections of this guide.

Help menu: This menu has items that launch the workbench documentation and that provide version information about the workbench and perspectives.

Perspective shortcut bar

Use the perspective shortcut bar to navigate conveniently between different perspectives. You might have the System Manager, Integrated Test Environment, and Java perspectives open; you can click on their workspace icons in the perspective shortcut toolbar to switch between them.

You can also navigate to other perspectives by using the **Window** menu:

- To navigate to a perspective represented by an icon that is higher up in the perspective shortcut bar than the icon for the perspective you are currently viewing use the keyboard shortcut **Alt+Up Arrow**.
- To navigate to a perspective represented by an icon that is lower down in the perspective shortcut bar than the icon for the perspective you are currently viewing, select **Perspective > Next** from the menu bar, or use the keyboard shortcut **Alt+Down Arrow**.

WebSphere Business Integration System Manager view

This view has the **User Projects** and **Integration Component Libraries** nodes, which are types of WebSphere InterChange Server projects.

For more information on working with these types of projects, see “Working with integration component libraries” on page 45 and “Working with user projects” on page 47.

InterChange Server Component Management view

Use this view to work with InterChange Server instances and the components in them. When you register InterChange Server instances, a row and entry are created underneath the **Server Instances** node for each server. You can expand the node to view the components in the repository of that instance, and can use right-click menu items to manipulate those components.

For more information about registering and working with InterChange Server instances, see “Working with InterChange Server instances.”

Editor view

This is the view in which you work with different resources in the framework, such as files and integration component definitions. Different editors open to work with different types of resources. For instance, text files are opened in a text editor whereas collaboration object definitions are opened in a WebSphere InterChange Server-specific editor.

Console view

This view has two tabs: **Console** and **Log Output**. When you compile maps or collaboration templates in System Manager, the **Console** tab displays messages to indicate whether compilation for each component completed successfully and the **Log Output** tab displays any errors or warnings encountered.

Working with InterChange Server instances

This section describes the tasks you perform in System Manager to work with InterChange Server.

Registering an InterChange Server instance

To work with an InterChange Server instance you must register it in System Manager. Do the following:

Important: To connect to an InterChange Server instance, it must be running. For information on starting InterChange Server see the *System Administration Guide*. You can still register a server that is not running, but System Manager will not be connected to it after doing so.

1. Do one of the following to display the "Register Server" dialog:
   - Select **Server** > **Register Servers** from the menu bar.
   - Right-click the **InterChange Server Instances** node in the InterChange Server Component Management view and choose **Register Server** from the context menu.

     Note: If you do not see the InterChange Servers view, enable it by following the instructions in "Showing and closing views" on page 87.
2. Do one of the following to enter the name of the server in the **Server name** field:
   - Type the name of the InterChange Server instance in the **Server name** field.

     Important: The name of an InterChange Server instance is case-sensitive, so you must be accurate when specifying the name.
   - Do the following to browse for the InterChange Server instance on the network:
     a. Click **Browse**.

        System Manager discovers active servers on the network and lists them at the "Find Server" dialog. Depending on the size, speed, and configuration of the network this may take some time.

        Note: Although you can register a server that is not running by typing the name of the server, you will not be able to browse for the server if it is not running.

     b. Select the server instance you want to register and click **OK**.
3. Type the user name to interact with the InterChange Server instance in the **User name** field.

   The default user name is `admin`.
4. Type the password for the user name specified in step 3 in the **Password** field.

   The default password for the default user name `admin` is `null`.
5. If you do not want to have to supply the user name and password each time you have to connect to the InterChange Server instance in System Manager then enable the **Save User ID and Password** checkbox.

   Important: Be sure to consider the security implications of caching the user name and password in this way. Some component definitions require sensitive information such as valid user names and passwords to log in to the applications being integrated. With such information, an individual could gain access to records stored in those applications and the critical information stored in those records, such as credit and payroll information. It is recommended that you only cache the user name and password in this way when the only information that could be accessed through System Manager is not production information.

6. To register the InterChange Server instance as a local test server, enable the **Local Test Server** checkbox and then either type the full path to the WebSphere InterChange Server product directory in the **Test Server Installation Path** field, or use the **Browse** button to navigate to the product directory.

   You should only enable this checkbox when you need to test an interface, and should leave it disabled when developing components or when working with a production server.

   For more information on testing interfaces, see Chapter 12, "Using Integrated Test Environment," on page 215 and Chapter 13, "Using Collaboration Debugger," on page 273.

7. Click **OK**.

   System Manager registers the InterChange Server instance, connects to it (if the name of the InterChange Server instance, the user name, and the password supplied are all accurate and the server and the IBM Java Object Request Broker (ORB) are running), and displays an entry for it in the InterChange Server Component Management view.

   You can subsequently work with the InterChange Server instance by right-clicking it in the InterChange Server Component Management view.

Figure 3 shows the "Register new server" dialog where the user name and password have been typed in and have been cached.

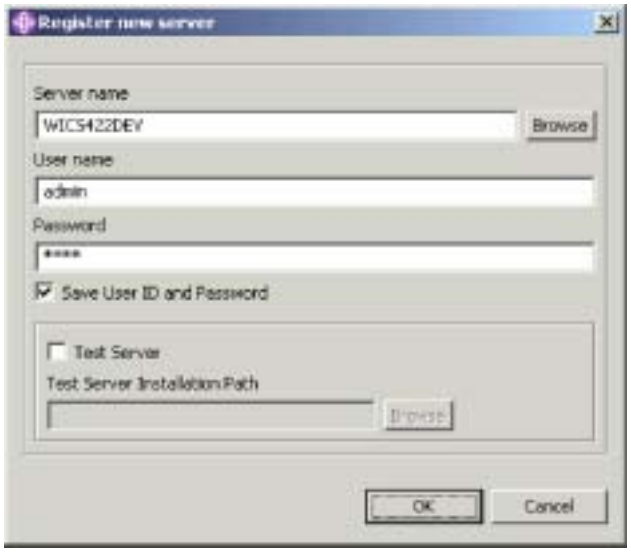

Figure 3. Registering an InterChange Server instance

Connecting to InterChange Server

When you register an InterChange Server instance in System Manager, System Manager automatically connects to the instance if the instance name, the user name, and the password are all accurate and the server and the IBM ORB are running.

If you have to shut down the instance, or exit from System Manager, then you need to reconnect System Manager to the instance. The task is slightly different depending on whether or not you chose to cache the user name and password, so follow the steps in the appropriate following section.

Connecting with a cached user name and password

Do the following if you chose to cache the user name and password when registering the server initially:

1. Right-click the entry for the InterChange Server instance in the InterChange Server Component Management view and choose **Connect** from the context menu.

   The "Login" dialog is displayed with the login information cached. Figure 4 shows the dialog.

Figure 4. Server User ID and Password dialog with cached information

2. Click **OK**.

Note: If System Manager was unable to connect to the server, see "Troubleshooting problems connecting to InterChange Server in System Manager" on page 94.

Connecting when the user name and password have not been cached

If you did not choose to cache the user name and password when initially registering an InterChange Server instance, then you must do the following:

1. Right-click the entry for the InterChange Server instance in the InterChange Server Component Management view and choose **Connect** from the context menu.
2. Type the user name to interact with the InterChange Server instance in the **User name** field.

   The default user name is `admin`.
3. Type the password for the user name supplied in step 2 in the **Password** field.

   The default password for the default user name `admin` is `null`.
4. If you do not want to have to supply the user name and password each time you have to connect to the InterChange Server instance in System Manager then enable the **Save userid/password** checkbox.

   Important: Be sure to consider the security implications of caching the user name and password in this way. Some component definitions require sensitive information such as valid user names and passwords to log in to the applications being integrated. With such information, an individual could gain access to records stored in those applications and the critical information stored in those records, such as credit and payroll information. It is recommended that you only cache the user name and password in this way when

the only information that could be accessed through System Manager is not production information.

5. Click **OK**.

InterChange Server modes

InterChange Server can run in different modes that best suit different stages of the implementation cycle.

Design mode

In design mode, InterChange Server permits the repository to be in an inconsistent state—you can add components to the repository without components they depend upon already existing. For instance, if you try to import a business object definition that has a child object into the repository, but the child object does not exist yet, then an InterChange Server in production mode would cause the import to fail to protect the integrity of the repository. An InterChange Server in design mode, however, would allow you to proceed so that you can assemble your integration components in a way that best suits your development approach.

Furthermore, compiling maps and collaboration templates when deploying a package to a design-mode server is optional. In production mode, the server automatically compiles all maps and collaboration templates.

Design mode is particularly useful when you are importing components from another environment. You may not be aware of all the dependencies yourself, so being able to incrementally import components without the import operations failing due to unresolved dependencies is very helpful.

To start InterChange Server in design mode, use the `-design` parameter when starting the server. Figure 5 on page 42 shows the shortcut for InterChange Server on a Windows system; the **Target** field contains the `-design` parameter. On a Unix system you would use the `-design` parameter when executing the `ics_manager` script.

Figure 5. InterChange Server shortcut specifying design mode

Production mode

In production mode, InterChange Server is designed to guarantee the integrity of the repository. It will not allow you to deploy a package with unresolved dependencies to the repository, and it automatically compiles all maps and collaboration templates in the deployment package. These restrictions guarantee that the server environment is in a state in which its components can execute properly. If there were components with unresolved dependencies or uncompiled components in the server environment at runtime then any transactions that involved those components would fail. Although that is an acceptable situation in a development environment, where it is presumed that you are still creating the required components, it is not considered acceptable in a production environment, so these restrictions enforce safe deployment procedures.

Production mode is the default mode for InterChange Server, so you do not have to take any configuration steps to start it in production mode. If you want to start it in production mode, however, be sure that you have not taken steps to start it in design mode and confirm its mode in the InterChange Server Component Management view of System Manager.

Changing the InterChange Server password

You can change the password for the user account that is used to connect to InterChange Server. Do the following to change the password:

1. Right-click the entry for the InterChange Server instance in the InterChange Servers view and choose **Change Password** from the context menu.

   The “Change InterChange Server Password” dialog displays, as shown in Figure 6 on page 43.

2. Type the current password in the **Old Password** field.
3. Type the new password in the **New Password** field.
4. Type the new password again in the **Confirm Password** field.
5. Click **OK**.

Figure 6. Changing the InterChange Server password

Refreshing InterChange Server

After you have deployed components to an InterChange Server instance you must refresh the instance in System Manager for it to accurately display the components in the server. For instance, if you deploy components to a server and then try to create a new integration library and add components to the library from the server, System Manager does not list the recently deployed components unless you refresh the server.

To refresh a server instance, right-click it in the InterChange Server Component Management view and choose **Refresh** from the context menu.

Disconnecting from InterChange Server

To disconnect System Manager from an InterChange Server instance, right-click the InterChange Server instance from which you want to disconnect in the InterChange Server Component Management view, then select **Disconnect** from the context menu.

Shutting down InterChange Server

To shut down an InterChange Server instance, right-click the InterChange Server instance you want to shut down in the InterChange Server Component Management view, then select **Disconnect** from the context menu and then either select **Gracefully** or **Immediately** from the submenu depending on how you want the instance to shut down.

If you select **Immediately** as the type of shutdown, then InterChange Server shuts down immediately and any flows that it might be processing at the time fail. You can subsequently resolve any failed flows by using Flow Manager. Use this type of shutdown in development and test environments where you do not care about the flows in the system or in production environments where there are no complications presented by submitting failed flows.

If you select **Gracefully** as the type of shut down, then the InterChange Server integration components will finish processing their current flows before the server

shuts down. Use this type of shutdown in production environments where there may be complications resulting from flow failures.

Note: You can also shut down InterChange Server by closing the console window in which it is running. This results in an immediate shutdown, and any flows currently being processed will fail. This may be acceptable in a development environment, if the integrity of the data does not matter, but should not be done in a production environment. A benefit of using System Manager to shut down InterChange Server even in a development environment is that the InterChange Server Component Management view displays the status of the server, but it cannot detect and report changes in the state of the server if it is shut down by some other means than the use of System Manager.

Deleting an InterChange Server instance from the view

You may want to remove an InterChange Server instance from the InterChange Server Component Management view in System Manager. For instance, if you have many servers registered it might be inconvenient to have ones you no longer work with listed. Do the following to remove an InterChange Server instance:

1. Ensure that System Manager is not connected to the InterChange Server instance you want to remove. You can do this by either disconnecting System Manager from the InterChange Server instance or by shutting down the InterChange Server instance.

   For more information on disconnecting System Manager from an InterChange Server instance, see “Disconnecting from InterChange Server” on page 43.

   For more information on shutting down an InterChange Server instance, see “Shutting down InterChange Server” on page 43.
2. In the InterChange Server Component Management view, right-click the InterChange Server instance you want to remove, then select **Delete** from the context menu.
3. At the “Delete server confirmation” prompt click **OK**.

   The server instance is removed from the view.

Other System Manager commands for InterChange Server

The context menu displayed when you right-click an InterChange Server instance has several other menu items, but they are described in other manuals.

Table 2. Other System Manager commands for InterChange Server

InterChange Server menu item	
Edit Configuration	For more information on this menu item, see Chapter 5, “Configuring WebSphere InterChange Server,” on page 97.
Statistics	For more information on this menu item, see the *System Administration Guide*.
System View	For more information on this menu item, see the *System Administration Guide*.
Server object delete	For more information on this menu item, see “Deleting components using the Server Object Delete Wizard” on page 81.
Delete Repository	“Deleting the entire repository” on page 80.

Table 2. Other System Manager commands for InterChange Server (continued)

InterChange Server menu item	
Monitor definition wizard	For more information on this menu item, see the *System Administration Guide.*

Working with integration component libraries

You use integration component libraries to store the components that you develop. This section describes how to create a new integration component library.

Once you have created an integration component library you will typically want to perform the following tasks as well:

- You will want to import components into the library. For information on several ways of doing this, see "Working with components in integration component libraries" on page 51.
- You will want to create shortcuts to the components in user projects. For more information on this, see "Working with user projects" on page 47.
- You will want to deploy components to an InterChange Server instance. For more information, see "Deploying components to a server" on page 65.
- You will want to export components to a package, either to import them into servers or other libraries or to back up your development. For more information, see "Exporting components to a package using System Manager" on page 63.

For conceptual information about integration component libraries, see "Integration component libraries" on page 1.

Creating integration component libraries

Do the following to create a new integration component library in System Manager by using a wizard:

1. Do one of the following to start the "New Integration Component Library" wizard:
   - Select **File > New > Integration Component Library** from the menu bar.
   - In the WebSphere Business Integration System Manager view, right-click the **Integration Component Libraries** folder and select **New Integration Component Library** from the context menu.
   - Click **Open The New Wizard button** in the toolbar and select **New Integration Component Library** from the menu.

   Figure 7 on page 46 shows the "New Integration Component Library" wizard.

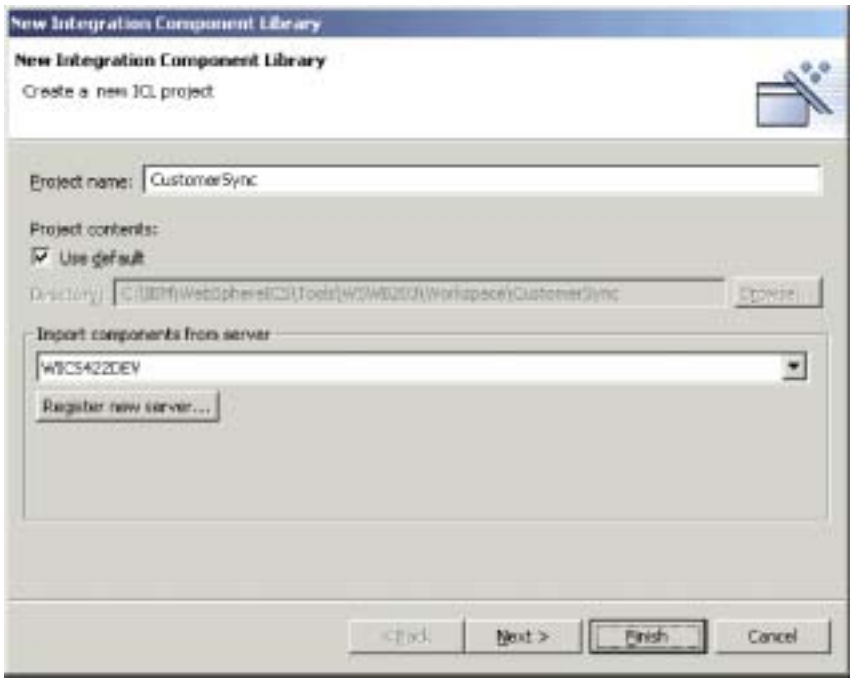

Figure 7. Creating an integration component library

2. Type a name for the integration component library in the **Project name** field.

   Project names can only contain alphanumeric characters and underscores, and must be specified in English.

   For recommendations on naming an integration component library, see "Creating your integration component libraries" on page 18.

3. To have the folder for the library created in the default location (your workspace) and with a name identical to the name specified for the library, leave the **Use default location** checkbox enabled.

   If you want to specify the name and location of the library folder, do the following:

   a. Clear the **Use default location** checkbox.

   b. Type the full path and name of the directory that you want to use for the library in the **Location** field, or click **Browse** to select an existing directory.

   Note: There is no way to create the folder for a library in the path of the workspace other than to let System Manager do it by use of the **Use default location** checkbox.

4. If you do not want to import components into the library from the repository of a server at this time, proceed to step 5 on page 47.

   If you do want to import components into the library from the repository of a running server, do the following:

   a. Do one of the following to specify the server from which you want to import the components:

      - Select the server in the **Import components from server** drop-down menu.

        The server must be running and System Manager must be connected to it for the server to be displayed in the drop-down menu.

- Click **Register new server** and add the server to the list through the wizard. For more information, see "Registering an InterChange Server instance" on page 38.

b. Click **Next**.

Figure 8 shows the screen where you import components from the server:

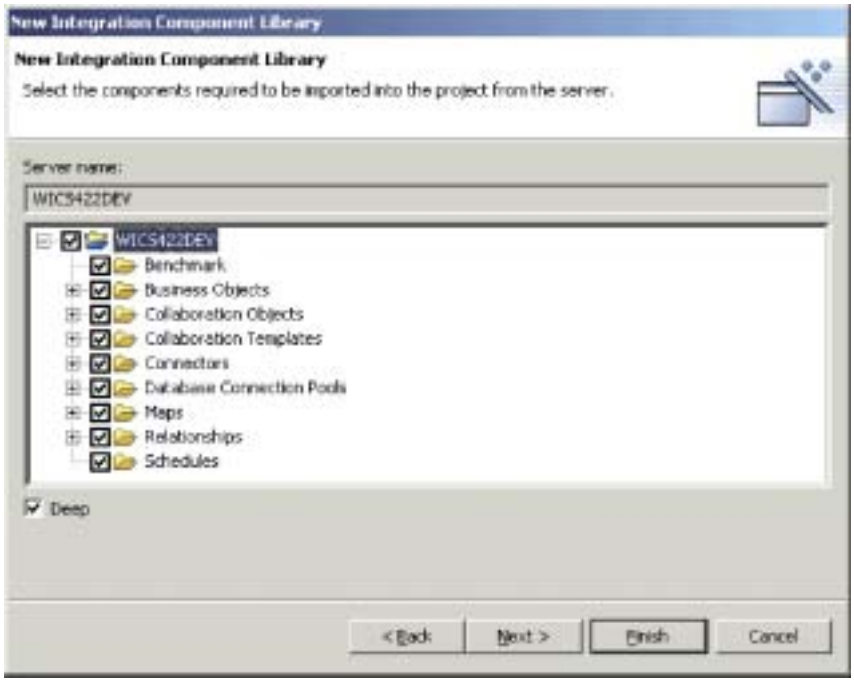

Figure 8. Importing components into a library from a server

c. At the next screen in the wizard, enable the checkbox next to a server instance to add all of the components in its repository or expand the server folder and enable the checkboxes next to component groups, or expand the folders for groups and enable the checkboxes for individual components.

d. Enable the **Deep** checkbox to add all of the dependencies of the selected components. For more information on dependencies, see "Dependencies and references" on page 82.

5. Click **Finish** to complete the wizard.

   System Manager creates a folder with the name you specified under the **Integration Component Libraries** folder.

Working with user projects

You create shortcuts in user projects to the integration components you want to work with in one or more libraries. User projects provide a way for you to organize your view of components as an interface. You must add component shortcuts to a user project to deploy the components to a server from System Manager.

This section contains the following sections:

For conceptual information about user projects, see “User projects” on page 2.

Configuring integration broker preferences for user projects

You can use System Manager to create user projects for multiple integration brokers. Do the following to enable System Manager for different integration brokers:

1. Open the file named `cwtools.cfg` in *ProductDir*`\bin` in a text editor.
2. To enable System Manager for WebSphere InterChange Server projects, set the `Installed` property in the `ICS_PROJECT` section to the value `true`.

   To enable System Manager for WebSphere Application Server projects, set the `Installed` property in the `WAS_PROJECT` section to the value `true`. For more information on working with WebSphere Application Server projects, see *Implementing Adapters with WebSphere Application Server*.

   To enable System Manager for WebSphere MQ Integrator Broker projects, set the `Installed` property in the `WMQI_PROJECT` section to the value `true`. For more information on working with WebSphere MQ Integrator Broker projects, see *Implementing Adapters with WebSphere Message Brokers*.
3. Save and close the file.

Creating user projects

Do the following to create a new user project in System Manager by using a wizard:

1. Do one of the following to start the “New User Project” wizard:
   - Select **File > New > User Project** from the menu bar.
   - In the WebSphere Business Integration System Manager view, right-click the **User Projects** folder, then select **New User Project**, then select **New ICS project** from the context menu.
   - In the WebSphere Business Integration System Manager view, expand the **User Projects** folder, then right-click the **InterChange Server Projects** folder and select **New ICS project** from the context menu.
   - Click **Open The New Wizard button** in the toolbar and select **New User Project** from the menu.
2. Type a name for the user project in the **Project name** field.

   Project names can only contain alphanumeric characters and underscores, and must be specified in English.

   For recommendations on naming user projects, see “Creating your user projects” on page 18.
3. To have the folder for the user project created in the default location (your workspace) and with a name identical to the name specified for the user project, leave the **Use default** checkbox enabled in the “Project contents” pane.

   If you want to specify the name and location of the user project folder, do the following:

   a. Clear the **Use default** checkbox in the “Project contents” pane.

   b. Type the full path and name of the directory that you want to use for the user project in the **Directory** field, or click **Browse** to select an existing directory.

   Note: There is no way to create the folder for a user project in the path of the workspace other than to let System Manager do it by use of the **Use default location** checkbox.

4. If you do not want to create shortcuts to existing integration components at this time, proceed to step 5.

   If you do want to create shortcuts to existing integration components, enable the checkbox next to an integration component library to create shortcuts to all of the components within it or expand an integration component library folder and enable the checkboxes next to component groups, or expand the folders for groups and enable the checkboxes for individual components.

   Note: If you select components with the same names from multiple integration component libraries you do not receive a prompt to inform you that there are duplicate references in your selection. In the event that you do select duplicate components, shortcuts are created for the component in the integration component library that was furthest down in the list of libraries when you made your selections in the wizard.

   Figure 9 shows the "New User Project" wizard.

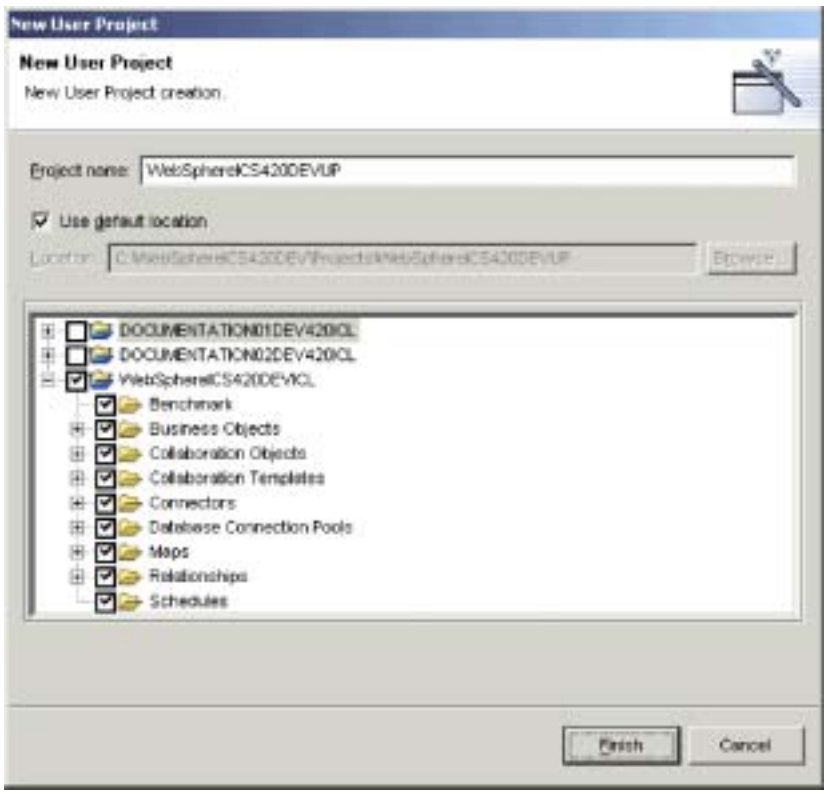

Figure 9. Creating a user project

5. Click **Finish** to complete the wizard.

   System Manager creates a folder with the name you specified in the **InterChange Server Projects** folder in **User Projects** folder.

Adding shortcuts to a user project

You add shortcuts to a user project to give yourself a view to an interface that you are working on. There are several ways to add shortcuts to a user project, described in the following sections:

- "Using the Dependency Tree" on page 50
- "Using the Update Project wizard" on page 51

- “Dragging-and-dropping components” on page 51
- “Importing a solution” on page 62
- “Importing a solution” on page 62

Using the Dependency Tree

The “Dependency Tree” wizard is the most convenient interface for adding component shortcuts to a user project. User projects are primarily designed to represent interfaces, and interfaces are generally centered around a collaboration object. User projects, then, are generally center around a collaboration object as well, and you can typically create the shortcuts you need for a user project by discovering the dependencies of a collaboration object.

For conceptual information about dependencies, see “Dependencies and references” on page 82.

Do the following to add shortcuts to a user project by using the “Dependency Tree” wizard:

1. Right-click an integration component, such as a collaboration object, in a library and choose **Show Dependencies** from the context menu.
2. Select the user project to which you would like shortcuts to be added from the **Add to the project** drop-down menu.
3. Select components for which you would like to create shortcuts in the left-hand pane of the wizard.

   You can use keyboard shortcuts to facilitate the process, such as holding down **Shift** to select ranges of objects and holding down **Ctrl** to select single non-contiguous objects.
4. Click the right-facing arrow to add the components to the right-hand pane of the wizard.

   Figure 10 shows the “Dependency Tree” wizard:

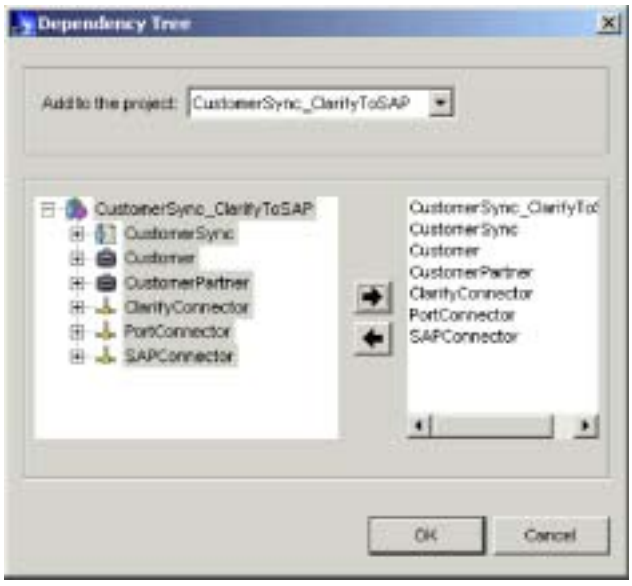

Figure 10. Adding shortcuts to a user project by using the Dependency Tree

5. Click **OK**.

   System Manager creates shortcuts to the selected components in the specified user project.

Using the Update Project wizard

The "Update Project" wizard provides an interface to add shortcuts to a user projects that is similar to the one displayed when you initially create a user project. Do the following to use the "Update Project" wizard:

1. Right-click any user project in the WebSphere Business Integration System Manager view and choose **Update project** from the context menu.
2. Enable the checkbox next to an integration component library to create shortcuts to all of the components within it or expand an integration component library folder and enable the checkboxes next to component groups, or expand the folders for groups and enable the checkboxes for individual components.
3. Click **Finish**.

   If there are shortcuts in the user project to components with the same name as those you selected in the wizard, then the wizard displays a prompt that allows you to do the following:

   - Overwrite the displayed component.
   - Overwrite all duplicate components.
   - Not overwrite the component.
   - Cancel the update operation.

   If you did not select any components with the same name as components for which shortcuts already exist in the user project, then shortcuts are added to the project and the wizard exits.

Dragging-and-dropping components

You can select components from integration component library folders and drag-and-drop them into a user project to add shortcuts to those components to the user project.

If there are no shortcuts in the folder of the user project currently then you must drag-and-drop the components onto the folder itself. Drag-and-drop the components onto the folder and release the mouse button when a square appears beneath the mouse pointer.

If there are already shortcuts in the folder of the user project then you cannot drag-and-drop components onto the folder. Drag-and-drop the components between existing shortcuts in the folder until a line appears and then release the mouse button.

Working with components in integration component libraries

The majority of your time is spent working with integration components when you implement a WebSphere InterChange Server integration system. Although this guide does not go into detail about how to develop individual components, this section does cover how to launch the designer tools, how to start creating new components, how to start modifying existing components, and how to work with the few components that are developed in System Manager.

For more information on how to develop integration components, see the following guides:

- *Collaboration Development Guide*
- *Map Development Guide*
- *Business Object Development Guide*
- *Connector Development Guide for Java or C++*

- *Access Development Guide*

For information on working with components that you have developed and deployed to an InterChange Server instance, see "Working with components in an InterChange Server repository" on page 73.

Launching designer tools

This section describes the different ways you can launch each of the designer tools. You can use the designer tools to create new components or to open and modify existing components.

Note: If you attempt to launch one of the designer tools and experience an error about a class not being found, you must launch System Manager and then try to launch the designer tool again. System Manager does not have to remain running after the tool is initially launched, however.

Business Object Designer

To launch Business Object Designer, do one of the following:

- Right-click the **Business Objects** folder in the WebSphere Business Integration System view and choose **Create New Business Object** from the context menu
- Select any folder in the WebSphere Business Integration System Manager view and do one of the following:
  - Select **Tools > Business Object Designer** from the menu bar
  - Click the **Business Object Designer** toolbar button
  - Use the keyboard shortcut **Ctrl+4**
- Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Development > Business Object Designer**

For more information on Business Object Designer, see the *Business Object Development Guide*.

Connector Configurator

To launch Connector Configurator, do one of the following:

- Right-click the **Connectors** folder in the WebSphere Business Integration System view and choose **Create New Connector** from the context menu
- Select any folder in the WebSphere Business Integration System Manager view and do one of the following:
  - Select **Tools > Connector Configurator** from the menu bar
  - Click the **Connector Configurator** toolbar button
  - Use the keyboard shortcut **Ctrl+1**
- Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Development > Connector Configurator**

For more information on Connector Configurator, see Chapter 7, "Configuring connectors," on page 125 and the *Connector Development Guide for Java or C++*.

Map Designer

To launch Map Designer, do one of the following:

- Right-click the **Maps** folder in the WebSphere Business Integration System view and choose **Create New Map** from the context menu

- Select the folder for any type of integration component in the WebSphere Business Integration System Manager view and do one of the following:
  - Select **Tools > Map Designer** from the menu bar
  - Click the **Map Designer** toolbar button
  - Use the keyboard shortcut **Ctrl+3**
- Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Development > Map Designer**

For more information on Map Designer, see the *Map Development Guide*.

Relationship Designer

To launch Relationship Designer, do one of the following:

- Right-click the **Relationships** folder in the WebSphere Business Integration System Manager view and choose **Relationship Designer** from the context menu
- Select the folder for any type of integration component in the WebSphere Business Integration System view and do one of the following:
  - Select **Tools > Relationship Designer** from the menu bar
  - Click the **Relationship Designer** toolbar button
  - Use the keyboard shortcut **Ctrl+5**
- Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Development > Relationship Designer**

For more information on Relationship Designer, see the *Map Development Guide*.

Process Designer

To launch Process Designer, do one of the following:

- Right-click the **Collaboration Templates** folder in the WebSphere Business Integration System view and choose **Create New Collaboration Template** from the context menu
- Select the folder for any type of integration component in the WebSphere Business Integration System view and do one of the following:
  - Select **Tools > Process Designer** from the menu bar
  - Click the **Process Designer** toolbar button
  - Use the keyboard shortcut **Ctrl+2**
- Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Development > Tools > Process Designer**

For more information on Process Designer, see the *Collaboration Development Guide*.

Creating new components

For the following components, launching their respective designer tools as described in “Launching designer tools” on page 52 allows you to create a new component of that type:

- Business objects
- Connectors
- Maps
- Relationships
- Collaboration templates

Some components do not have dedicated designer tools, so you create them in interfaces displayed by System Manager. To create new benchmarks, collaboration objects, database connection pools, and schedules see Table 3:

Table 3. Techniques for creating new integration components without designer tools

Component	Technique	For more information, see...
Benchmark	Right-click the **Benchmark** folder in the WebSphere Business Integration System view and select one of the following from the Benchmark Setup context menu: • **Configure new Benchmark** • **Edit existing Benchmark** • **Generate work load to a file**	*Benchmarking Guide*
Collaboration object	Right-click the **Collaboration Objects** folder in the WebSphere Business Integration System view and select **Create New Collaboration Object** from the context menu	Chapter 9, "Configuring collaboration objects," on page 165
Database connection pool	Right-click the **Database Connection Pools** folder in the WebSphere Business Integration System view and select **Create New Database Connection** from the context menu	Chapter 8, "Configuring database connection pools," on page 155
Schedule	• Right-click the **Schedules** folder in the WebSphere Business Integration System view and select **Edit Components' schedule** from the context menu • Right-click any component that can be scheduled in a library or the shortcut to any component that can be scheduled in a user project and select **Edit Components' schedule** from the context menu	*System Administration Guide*

Modifying existing components

To modify business objects, connectors, maps, relationships, and collaboration templates you can do the following:

- Double-click the component in a library or the shortcut to a component in a user project
- Select the component in a library or the shortcut to a component in a user project and do one of the following:
  - Launch its designer tool as described in "Launching designer tools" on page 52
  - Press **Enter**
  - Press **Ctrl+E**
  - Select **Component > Edit Definitions** from the menu bar
- Right-click the component in a library or the shortcut to a component in a user project and choose **Edit definition** from the context menu
- Launch the component's designer tool as described in "Launching designer tools" on page 52 and then open the component after the tool has started

See Table 4 for information on how to modify components benchmarks, collaboration objects, database connection pools, and schedules:

Table 4. Techniques for modifying integration components without designer tools

Component	Technique	For more information, see...
Benchmark	Right-click the **Benchmark** folder in the WebSphere Business Integration System view and select one of the following from the Benchmark Setup context menu: • **Configure new Benchmark** • **Edit existing Benchmark** • **Generate work load to a file**	*Benchmarking Guide*
Collaboration object	Double-click the collaboration object in the **Collaboration Objects** folder in the WebSphere Business Integration System view	Chapter 9, "Configuring collaboration objects," on page 165
Database connection pool	You cannot modify a database connection pool. You may modify some properties of the pool component, but cannot change definition elements such as the database to which the pool connects.	Chapter 8, "Configuring database connection pools," on page 155
Schedule	• Right-click the **Schedules** folder in the WebSphere Business Integration System view and select **Edit Components' schedule** from the context menu • Right-click any component that can be scheduled in a library or the shortcut to any component that can be scheduled in a user project and select **Edit Components' schedule** from the context menu The "Schedule" interface appears and allows you to modify schedules that have been defined.	*System Administration Guide*

Importing components into a library from a server using the import wizard

You can import integration components into a library from an InterChange Server repository. This is most helpful when you join a development team and have to update your individual development environment with existing components.

You can import components from the libraries of other developers, but if they have not been maintaining their components properly then you may want to use only components of known quality. Test and production environments are typically maintained properly and can generally be trusted more than individual development environments.

Do the following to import components into an integration component library from an InterChange Server repository:

Note: To import components into a library from an InterChange Server instance, the server must be running. If System Manager is not connected to the server when you start the "Import components" wizard then no components will be displayed for importing.

1. Connect System Manager to InterChange Server as described in "Connecting to InterChange Server" on page 39.
2. In the WebSphere Business Integration System Manager view, right-click the library into which you want to import components, select **Import components from server** from the context menu, and then do one of the following to start the "Import components" wizard:
   - Select **Components** from the submenu to display all types of components in the server.
   - Select a component type from the submenu to only display that type of component.

   System Manager displays the "Import the additional components from the server" wizard, as shown in Figure 11.

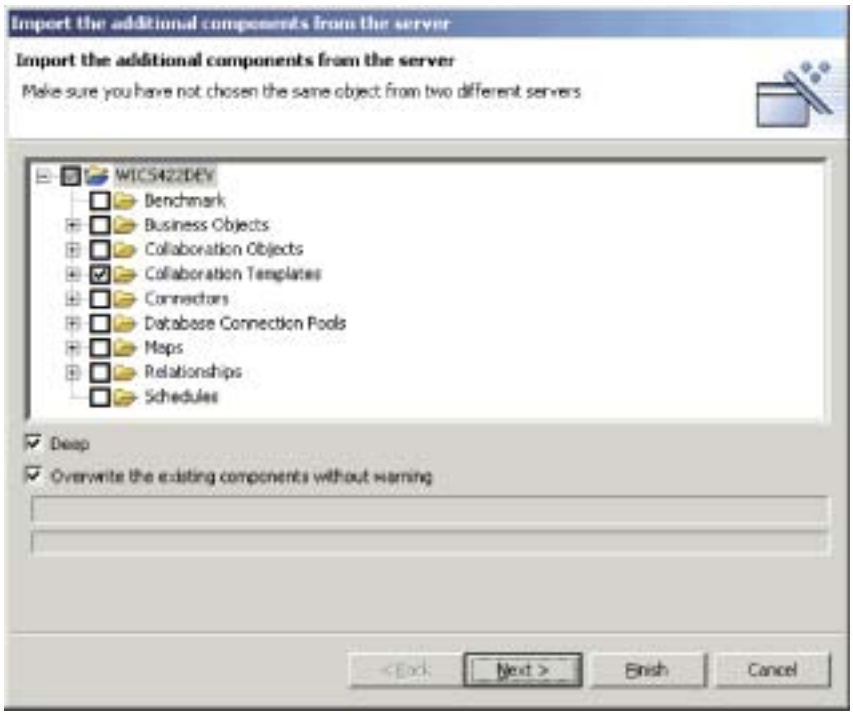

Figure 11. Importing components from the server

3. Enable checkboxes next to the servers, component groups, or individual components that you want to import from the server.
4. Enable the **Deep** checkbox if you also want to import the dependencies of the component. For more information about dependencies, see "Dependencies and references" on page 82.
5. Enable the **Overwrite the existing components without warning** checkbox if you want to overwrite any components you currently have in the library that have the same name as those you selected to import from the server.

   If you did not select any maps or collaboration templates then proceed to step 7 on page 57.

If you selected any maps or collaboration templates to import then do one of the following:

- If you are importing the maps and collaboration templates from a server instance that is prior to 4.2.2 and want to upgrade them to 4.2.2 format at this time, proceed to step 6 to advance to the next screen of the wizard.
- If the maps and collaboration templates you are importing are already in 4.2.2 format, or if you do not want to upgrade them at this time, proceed to step 7 to complete the wizard.

6. At the second screen of the "Import components" wizard, enable the **Maps** and **Collaboration Templates** checkboxes to have System Manager convert any maps or collaboration templates among the components you selected to the 4.2.2 format.

   If you used repos_copy to deploy components from a release prior to 4.2.2 to a release 4.2.2 server repository then the maps and collaboration templates stored there are not upgraded. You can only upgrade these components to the 4.2.2 format using System Manager or the Map Designer and Process Designer tools.

   At the second screen of the "Import components" wizard you can leave the **Maps** and **Collaboration Templates** checkboxes enabled to have them converted at this time, or you can disable the checkboxes to defer the conversion until later, when you can do so in the designer tools. You may want to defer the conversion because it can be a time-consuming process, particularly with a large number of those types of components.

   Figure 12 shows the second screen of the "Import components" wizard:

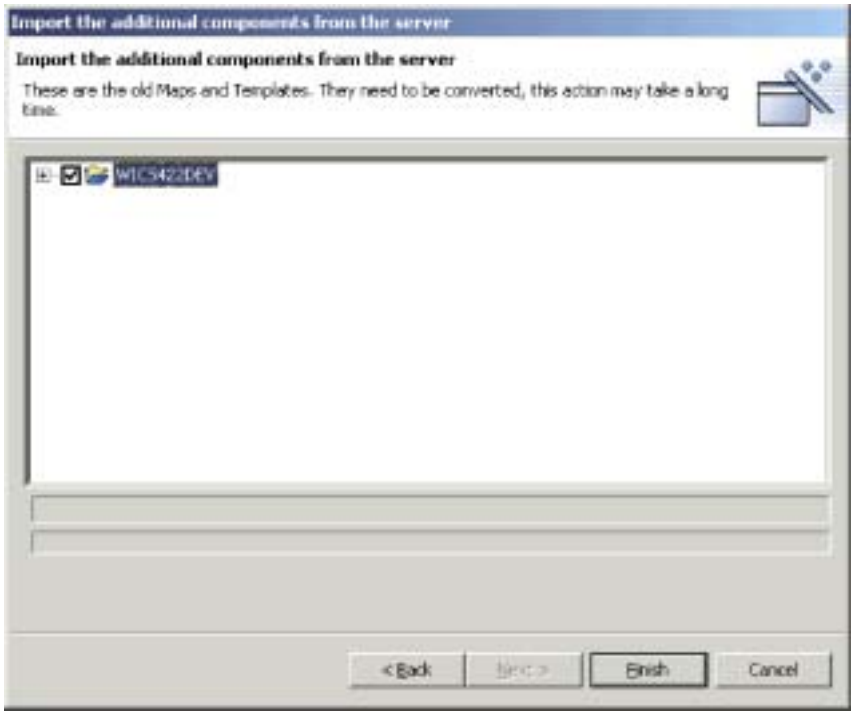

Figure 12. Converting maps and collaboration templates when importing from the server

7. Click **Finish**.

Importing components into a library from the server using drag-and-drop

Do the following to import components into an integration component library by using drag-and-drop techniques among the System Manager views:

1. Do the following in the InterChange Server Component Management view to select the components you want to deploy:
   - Choose all components of one type by selecting its folder in a user project or integration component library.
   - Expand the component folders and select individual components to choose them. You can use standard Windows selection techniques to select multiple user projects for deployment at the same time, such as the following:
     – Hold down **Shift** to select contiguous items
     – Hold down **Ctrl** to select non-contiguous items
2. Drag-and-drop the selected resources onto the integration component library into which you want to import the components in the WebSphere Business Integration System Manager view.

   System Manager attempts to import the components to the specified integration component library. Messages and errors are displayed in the Console view.

Importing components into a library from a package

You can export integration components to a `.jar` file package, as described in "Exporting components to a package using System Manager" on page 63. This makes it easy to migrate components between environments, share them with other developers, and submit them to technical support.

Do the following to import components into an integration component library from a package:

Important: If there are components with the same name as those in the package you are importing, System Manager overwrites the existing components without a warning.

1. Right-click an integration component library and choose **Import from Repository File** from the context menu.

   System Manager displays the "Import Repository File" wizard, as shown in Figure 13 on page 59.

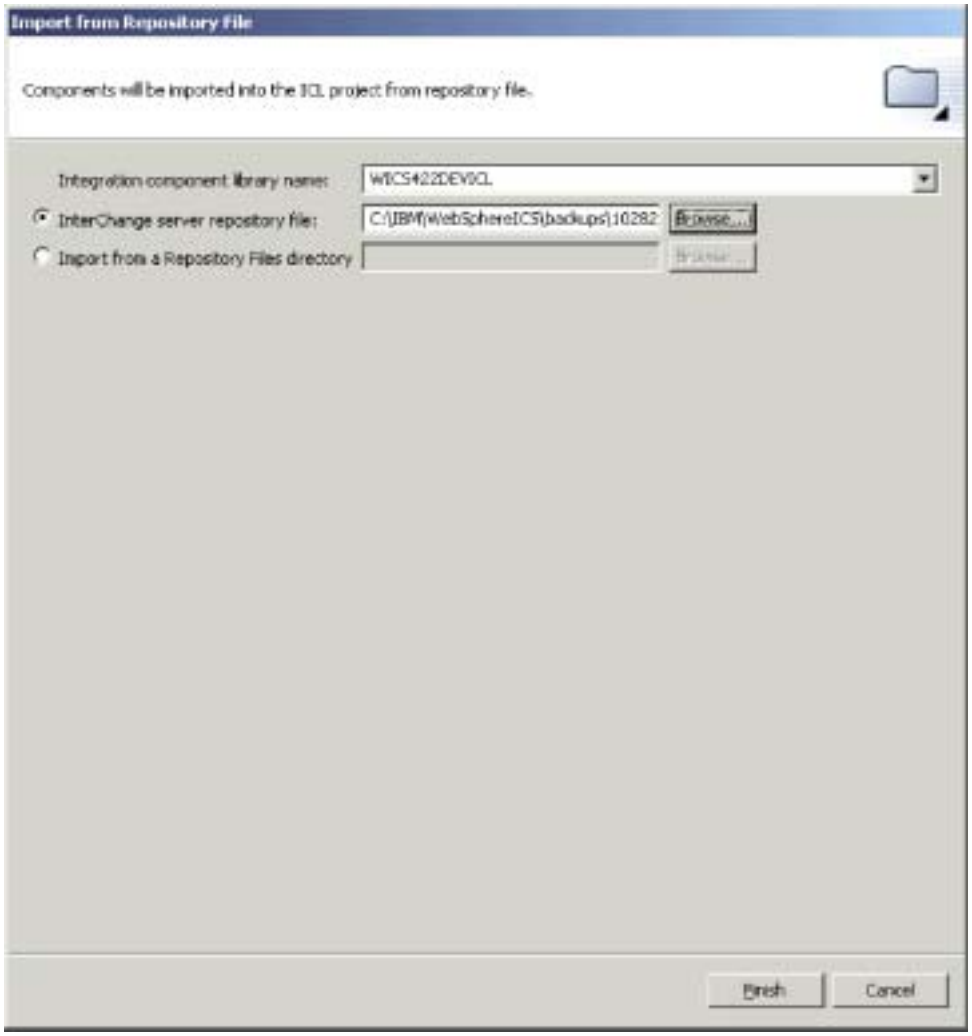

Figure 13. Importing a package

2. At the “Import Repository File” screen, ensure that the Integration Component Library Name drop-down menu contains the name of the library into which you want to import the components.

   If you launched the “Import from Repository File” wizard from a library other than the one into which you want to import the components, you can change the destination this way instead of closing the wizard and launching it again.
3. Do one of the following to specify the components you want imported:
   - To import a single package file, either type the full path and name of the `.jar` file that you want to import in the **InterChange server repository file** field or click **Browse** to select the file.
   - To import an entire directory of package files, either type the full path to the directory in the **Import from a Repository Files directory** field or click **Browse** to select the file.
4. Click **Finish**.

Note: Do not use the **File > Import** menu item in the workbench to import a package file. Although the “Zip file” wizard works with archives with the `.jar` extension, and InterChange Server package files have a `.jar` extension, the “Zip file” wizard does not work properly with package files.

Modifying map and collaboration object properties in integration component libraries

Maps and collaboration objects have properties that you can modify to change the behavior of those components. To modify the properties of a component, right-click either the component in an integration component library or a shortcut to it in a user project and select **Properties** from the context menu. System Manager displays a dialog to configure the properties of the component.

For more information about collaboration object properties, see Chapter 9, "Configuring collaboration objects," on page 165.

For more information about map properties, see the *Map Development Guide*.

Validating database connection pools

There is a context menu item named **Validate Connection** for database connection pool components. For more information about this menu item, see "Validating database connection pools" on page 161..

Compiling maps and collaboration templates

You can compile maps and collaboration templates in System Manager, which is very convenient because components must be compiled to be deployed or exported to a package.

If the component you are compiling uses libraries that are not provided by IBM—for example, libraries you have created yourself, or are provided by another vendor—then you must configure System Manager to reference those libraries. For more information, see 4 on page 91 for information about the "Compiler" preferences interface.

Do one of the following to compile a map or collaboration template:

- Select the map or collaboration template in an integration component library or the shortcut to one in a user project and do one of the following:
  - Press **Ctrl+F7**
  - Select **Component > Compile** from the menu bar
- Right-click the map or collaboration template in an integration component library or the shortcut to one in a user project and select **Compile** from the context menu

Do one of the following to compile all the maps or collaboration templates in an integration component library:

- Select the **Maps** or **Collaboration Templates** folder in an integration component library and do one of the following:
  - Press **F7**
  - Select **Component > Compile all** from the menu bar
- Right-click the **Maps** or **Collaboration Templates** folder in an integration component library and select **Compile all** from the context menu

To compile a map with its submaps, right-click the map in an integration component library or the shortcut to a map in a user project and select **Compile with submap(s)** from the context menu.

Working with solutions

You can export a user project as a solution. This action copies the shortcuts from the user project as well as the component definitions that the shortcuts reference in the integration component libraries. This makes it easy to migrate an entire interface or business integration system from one environment to another.

Exporting a solution

Do the following to export a user project and the integration components it references as a solution:

1. In the WebSphere Business Integration System Manager view, expand the **User Projects** folder, then right-click the **InterChange Server Projects** folder and choose **Export Solution** from the context menu.

   System Manager displays the "Export Solution" wizard, as shown in Figure 14.

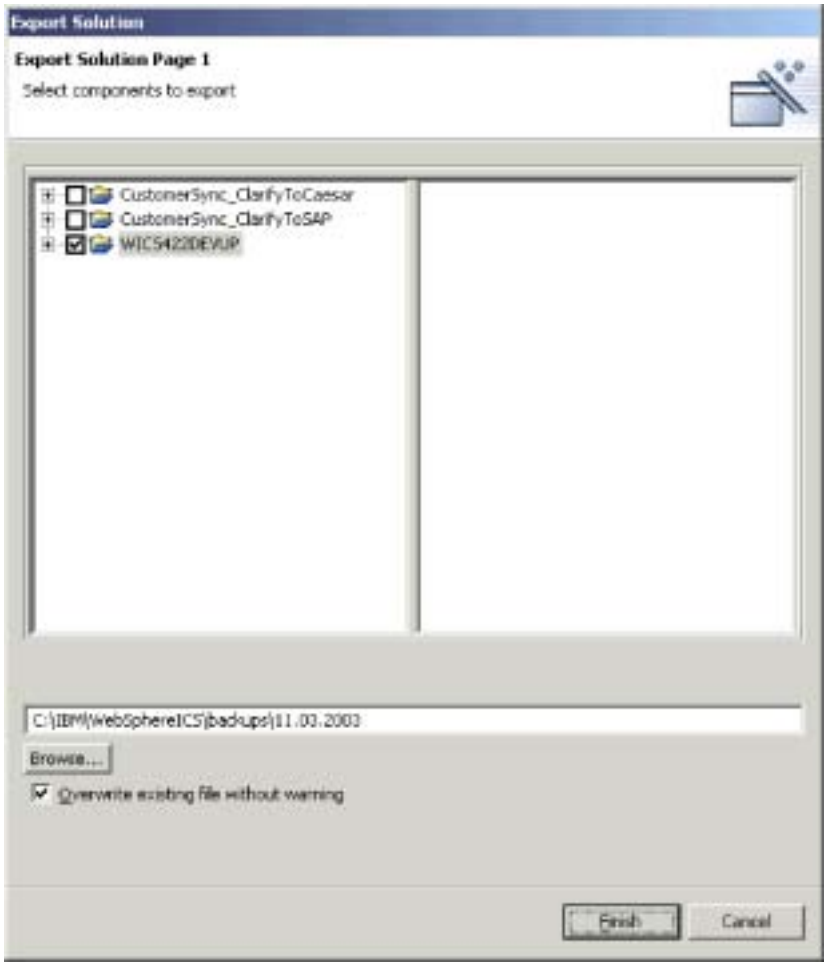

Figure 14. Exporting a solution

2. Use the following techniques to select the components you want to export:
   - Enable the checkbox next to user projects to select all of the components in the projects.
   - Enable the checkbox next to a component group to select all of the components in the group.

- Highlight a component group and then enable checkboxes next to individual components in the right-hand pane to select those components.

3. Either type the full path and name of the directory into which the solution should be exported in the text field at the bottom of the wizard screen, or click **Browse** to navigate to the desired directory.
4. Click **Finish**.

   System Manager does the following to export the solution in the directory specified in step 3:

   - Creates a `User` directory that contains the shortcuts in the user projects selected during the export of the solution.
   - Creates a `System` directory that contains the directories of the integration component library referenced by the shortcuts in the user projects selected during the export of the solution.
5. When prompted that the export operation completed successfully, click **OK**.

Importing a solution

Do the following to import a solution:

1. In the WebSphere Business Integration System Manager view, expand the **User Projects** folder, then right-click the **InterChange Server Projects** folder and choose **Import Solution** from the context menu.

   System Manager displays the “Import Solution” wizard, as shown in Figure 15.

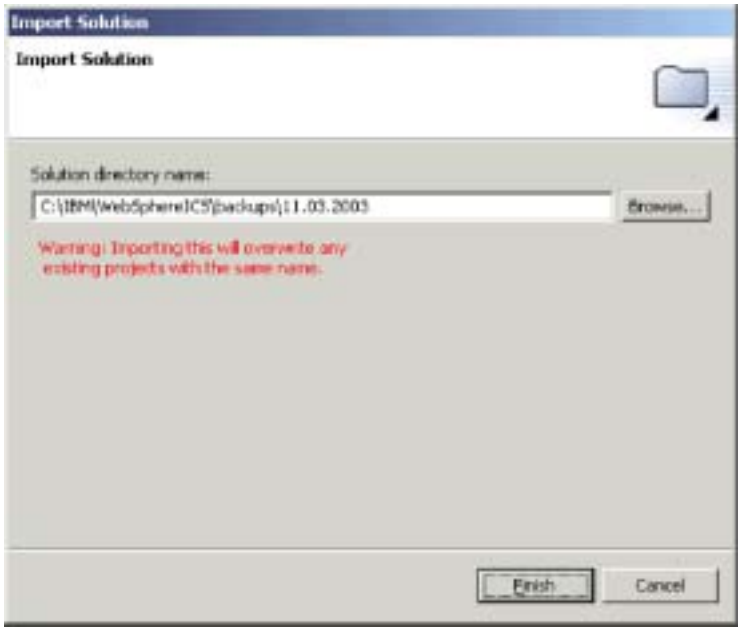

Figure 15. Importing a solution

2. Either type the full path and name of the directory in which the exported solution exists into the **Solution directory name** field, or click **Browse** to navigate to the desired directory.
3. Click **Finish**.

   System Manager creates the integration component libraries and user projects defined in the exported solution in your environment.

Exporting components to a package using System Manager

You can export integration components to a package file. Integration components are resources, which are ultimately files stored in the file system as described in “Resources” on page 32. When System Manager exports components to a package, it compresses the following resources into a `.jar` (Java archive) file:

- Definition files (stored in XML format, with different extensions depending on the component type)
- Java source files for maps and collaboration templates
- Message files

Do the following to export components to a package:

Important: In order to export maps or collaboration templates they must be compiled. If you try to export uncompiled maps or collaboration templates System Manager prompts you to compile them first.

1. Right-click either an integration component library or a user project that contains the components you want to export and choose **Export as Repository File** from the context menu.

   System Manager displays the “Export Repository File” wizard, as shown in Figure 16 on page 64.

Figure 16. Exporting a package

2. Use the following techniques to select the components you want to export:
   - Enable the checkbox next to the integration component library or user project to select all of the components in the library or project.
   - Enable the checkbox next to a component group to select all of the components in the group.
   - Highlight a component group and then enable checkboxes next to individual components in the right-hand pane to select those components.
3. If you plan to specify an existing `.jar` file to export the components to and you want to overwrite it without receiving a prompt, enable the **Overwrite existing file without warning** checkbox.

   Note: To benefit from the **Overwrite existing file without warning** checkbox you must enable it before specifying the file to use as described in step 4. System Manager prompts you to overwrite a file as soon as it detects that an existing file has been specified and does not wait until you have finished the wizard, so you must enable this option beforehand to benefit from it.
4. Type the name and path of the `.jar` file to which the components should be exported into the **InterChange server repository jar file path** field, or click **Browse** to select a file to overwrite, or navigate to a directory and specify a file name.

If you specified the name and path to an existing file and did not enable the **Overwrite existing file without warning** checkbox and you want to overwrite the existing file then click **Yes** when prompted.

Note: If you type the name and path of the file into the field you must include the `.jar` extension in order for the **Finish** button to be enabled.

5. Click **Finish** to complete the wizard.

Deploying components to a server

As described in "The life cycle of an interface" on page 3, you create integration components in a library in your local file system and deploy them to an InterChange Server instance to make them executable.

You can deploy a package of integration components using either the System Manager graphical interface or the repos_copy command-line interface. For information on using System Manager, see "Deploying components using the deployment wizard" on page 68. For information on using repos_copy, see Chapter 6, "Using repos_copy," on page 113. For information on the advantages and disadvantages of each interface, see "Deciding to use System Manager or repos_copy for deployment."

Deciding to use System Manager or repos_copy for deployment

Table 5 describes the advantages and disadvantages of using System Manager or repos_copy for deployment. Evaluate the two interfaces and use whichever one is best suited to your needs.

Table 5. Advantages and disadvantages of System Manager and repos_copy in deployment

Interface	Advantages	Disadvantages
System Manager	• System Manager uses a graphical interface. • You can selectively choose the components in the user project that you want to deploy. • You can choose to create relationship schemas in the server repository during the deployment, or can choose for them to not be created. • You can deploy multiple user projects at the same time. • With drag-and-drop deployment, you can deploy user projects, integration component libraries, or even individual components easily.	• System Manager runs only on Windows.

Table 5. Advantages and disadvantages of System Manager and repos_copy in deployment (continued)

Interface	Advantages	Disadvantages
repos_copy	• There is no requirement to have components grouped together, as they must be in a user project when deploying through System Manager. • Repos_copy is platform-independent.	• Repos_copy uses a command line interface. • You must export the components you want to deploy to a package file first, using either System Manager or repos_copy. • You cannot selectively deploy components in the package file. • You can only deploy a single package file at a time.

Setting the initial states of components for deployment

For components with states (such as connectors, collaboration objects, maps, relationships, and database connection pools), you can set the state in which the component initializes when rebooting the server after deployment. Do the following to set the initial post-deployment state of components:

1. Select the shortcuts for the components whose initial states you want to set in a user project.

   You can use standard Windows selection techniques to choose multiple components, such as the following:

   - Hold down **Shift** to select contiguous items
   - Hold down **Ctrl** to select non-contiguous items

2. Right-click the selection made in step and do one of the following:
   - Choose **Start** or **Start All** to have the selected components initialize in a "Running" state.
   - Choose **Stop** or **Stop All** to have the selected components initialize in a "Stopped" state.

Validating a package using System Manager

You can validate a package comprised of the components you want to deploy before performing the deployment to ensure that the deployment will succeed. Do the following to validate a package of components:

1. Add shortcuts for the components you want to validate to user projects.

   For information on creating a user project see "Creating user projects" on page 48, and for information on adding shortcuts to a user project see "Adding shortcuts to a user project" on page 49.

   Note: You can validate multiple user projects at once, so add shortcuts for all of the desired components to the appropriate user projects.

2. In the WebSphere Business Integration System Manager view, select the user projects you want to validate, right-click one of the selected user projects, and choose **Validate user project** from the context menu.

   System Manager displays the "Validate Project(s)" wizard, as shown in Figure 17 on page 67.

Figure 17. Selecting components for validation

3. Select the server against which you want to validate the components from the **Please select the server to validate with** drop-down menu.
4. Use the following techniques to select the components you want to validate:
   - Enable the checkbox next to the integration component library or user project to select all of the components in the library or project.
   - Enable the checkbox next to a component group to select all of the components in the group.
   - Expand a component group and then enable checkboxes next to individual components to select those components.

   Note: If you select components with the same name from multiple user projects you will be required to select a single one from among the duplicates at the next screen of the wizard.

   If you have not selected any duplicate components, proceed to step 6 on page 68.

   If you have selected duplicate components, proceed to step 5.
5. Click **Next**.

   System Manager displays the “Local Duplicates” screen, as shown in Figure 18 on page 68.

Figure 18. Selecting distinct components from user projects to validate

Select distinct components among the duplicates.

6. Click **Finish**.

   System Manager creates a package containing the selected components and validates it against the server repository. A message is displayed to indicate whether validation was successful or not.

Deploying components using System Manager

You can deploy components to an InterChange Server instance in one of two ways described in the following sections:

- "Deploying components using the deployment wizard"
- "Deploying components using drag-and-drop" on page 72

Deploying components using the deployment wizard

Do the following to use the deployment wizard to deploy a package of components to an InterChange Server instance:

1. If any of the components you want to deploy already exist in the server repository you must stop them prior to deploying them.

   Use the InterChange Server Component Management view or System Monitor to stop the duplicate components in the server before deploying a package. For more information, see "Managing component states in the repository" on page 74.

2. It is recommended that you set the initial state for the components you deploy. For more information, see “Setting the initial states of components for deployment” on page 66.
3. Add shortcuts for the components you want to deploy to a user project.

   For information on creating a user project see “Creating user projects” on page 48, and for information on adding shortcuts to a user project see “Adding shortcuts to a user project” on page 49.

   Note: You can deploy multiple user projects at once, so add shortcuts for all of the desired components to the appropriate user projects.
4. In the “WebSphere Business Integration System Manager” view, select each user project you want to deploy to the server instance. You can use standard Windows selection techniques to select multiple user projects for deployment at the same time, such as the following:
   - Hold down **Shift** to select contiguous items.
   - Hold down **Ctrl** to select non-contiguous items.
5. Right-click a selected user project and choose **Deploy user project** from the context menu.

   System Manager displays the “Deploy wizard page 1 screen”, as shown in Figure 19.

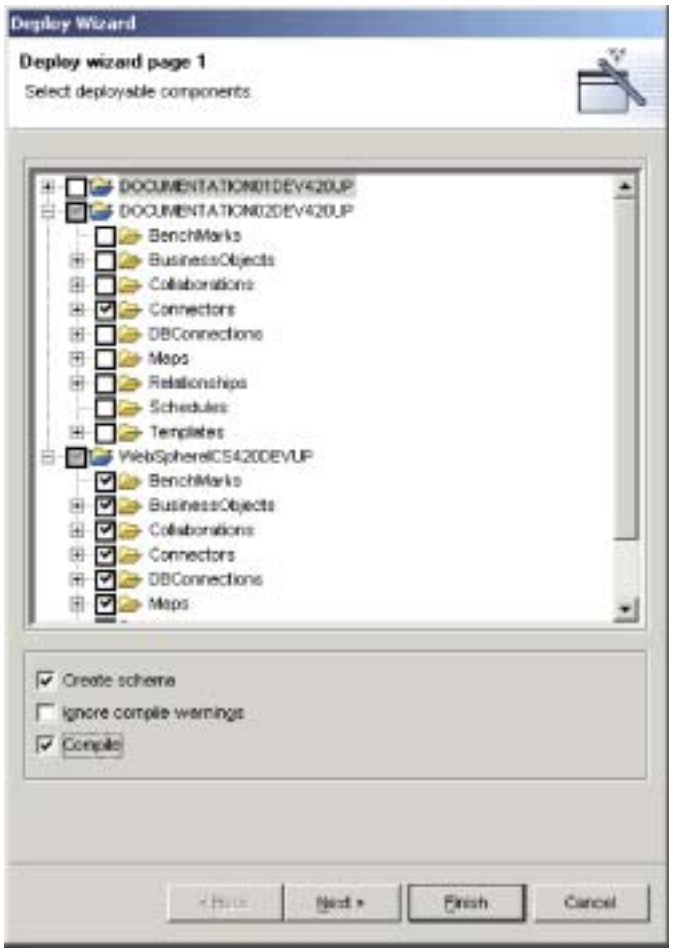

Figure 19. Selecting deployable components and deployment options

6. At the “Deployable Components” screen, do the following:

a. Select the server instance to which you want to deploy the components from the **Please select the destination server** drop-down menu.

b. Use the following techniques to select the components to deploy:

- Enable the checkbox next to a user project to deploy all of the components in the project.
- Expand the user project node and enable the checkbox next to entire component groups to deploy all of the components of that type.
- Expand the user project node and a component group node and enable the checkbox next to individual components to deploy just those components.

Note: If you select components with the same name in different user projects you will be prompted in the next screen of the wizard to choose which component among the duplicates should be deployed.

c. Enable the **Create schema** checkbox if you want to create the database connection pool and relationship schemas during the deployment process.

The schemas for relationships and database connection pools must be created for interfaces that reference them to work. If you are deploying an interface for the purpose of running it then you must enable this option for any components the interface uses.

Note: You should only enable this option if the settings for the database connection pools and relationships reference the appropriate connection information for the server into which you are deploying the components. If you are migrating components from one environment where the database settings were correct to another environment where they will no longer be valid, you may not want to create the schemas for these components during deployment.

d. Enable the **Ignore compile warnings** checkbox if you do not want to receive a prompt about warnings generated during compiling, such as about deprecated methods.

e. Enable the **Compile** checkbox if you want to compile the map and collaboration templates you selected for deployment.

Maps and collaboration templates be compiled for interfaces that reference them to work.

Note: Compiling can take a long time depending on the number of components and you may want to deploy them to the server first and compile them afterwards.

f. Do one of the following depending on your selections:

- If you have not selected any components that are duplicates across the selected user projects and have not selected any components that already exist in the server repository then you may start the deployment at this time; proceed to step 9 on page 72.
- If you have not selected any components that are duplicates across the selected user projects, but have selected components that already exist in the server repository, then the “Server Duplicate Objects” screen appears; proceed to step 8 on page 71.
- If you have selected duplicate components within the user projects then proceed to step 7.

7. Click **Next**.

If you have selected duplicate components within the user projects then the "Deploy wizard page 2" screen appears, as shown in Figure 20:

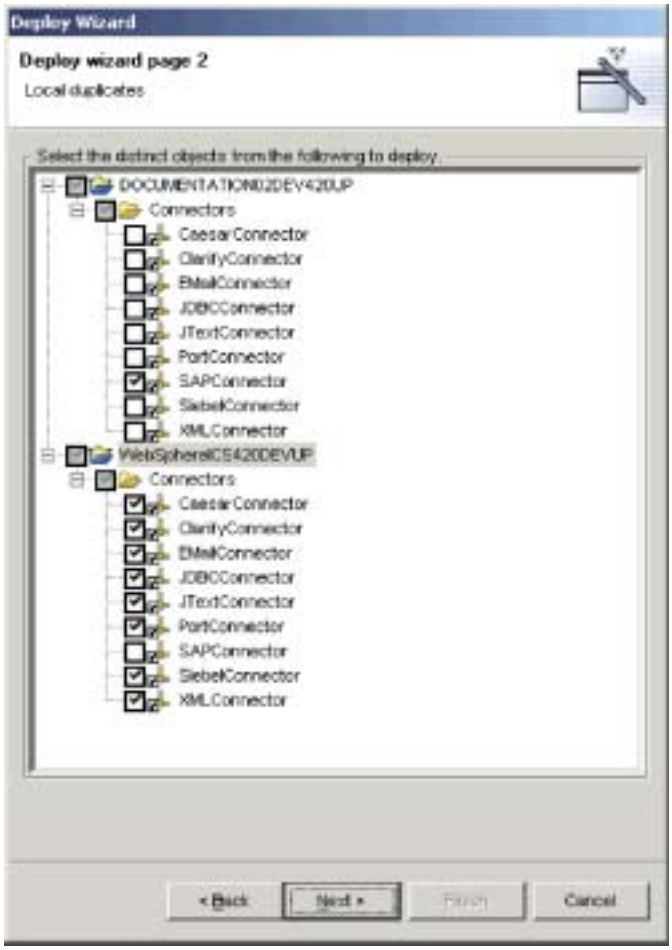

Figure 20. Selecting among local duplicates when deploying

Expand the folders of the displayed user projects and enable checkboxes for the particular components you want deployed between the local duplicates.

Do one of the following depending on your selections:

- If you have not chosen to deploy any components that already exist on the server then you may start the deployment at this time; proceed to step 9 on page 72.
- If you have chosen to deploy components that already exist on the server then proceed to step 8.

8. Click **Next**.

   If you have chosen to deploy components that already exist on the server then the "Server Duplicate Objects" screen appears, as shown in Figure 21 on page 72:

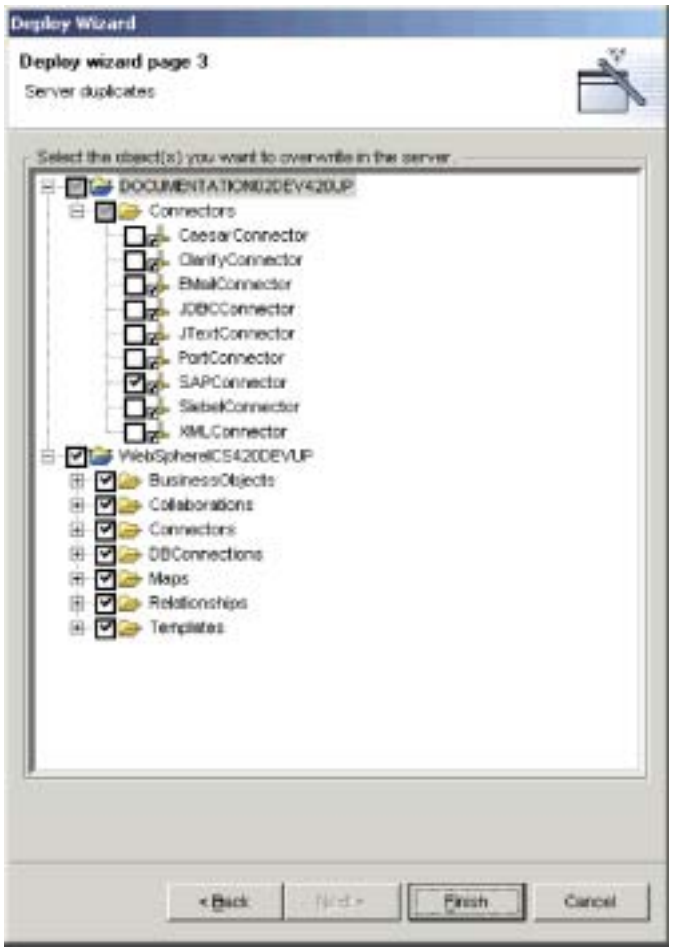

Figure 21. Selecting duplicate objects to overwrite in the server

Expand the folders of the displayed user projects and enable checkboxes for the duplicate components you want to overwrite in the server repository.

9. Click **Finish**.

   System Manager attempts to deploy the selected components into the server repository. If you view the server logging output you will see messages logged as the deployment session is started and components are added to the repository.

   When the deployment session finishes System Manager either presents an informational prompt that the deployment succeeded or an error prompt that the deployment failed.

10. Take note of any significant information displayed in the prompt. For instance, the prompt might tell you to reboot the server to activate some deployed components, or might tell you that the deployment failed because active components cannot be overwritten. Click **OK** and take whatever actions were specified by the prompt.

Deploying components using drag-and-drop

Do the following to deploy components by using drag-and-drop techniques among the System Manager views:

1. If any of the components you want to deploy already exist in the server repository you must stop them prior to deploying them.

Use the InterChange Server Component Management view or System Monitor to stop the duplicate components in the server before deploying a package. For more information, see “Managing component states in the repository” on page 74.

2. Do the following in the WebSphere Business Integration System Manager view to select the components you want to deploy:
   - Select user projects or integration component libraries to choose all of the components in them.
   - Choose all components of one type by selecting its folder in a user project or integration component library.
   - Expand the component folders and select individual components to choose them. You can use standard Windows selection techniques to select multiple user projects for deployment at the same time, such as the following:
     – Hold down **Shift** to select contiguous items.
     – Hold down **Ctrl** to select non-contiguous items.
3. Drag-and-drop the selected resources onto the InterChange Server instance to which you want to deploy the components in the InterChange Server Component Management view.

   System Manager attempts to deploy the components to the specified InterChange Server instance. Messages and errors are displayed in the Console view.

   Note: If you selected duplicate components in the WebSphere Business Integration System Manager view then System Manager generates an error when you try to deploy the components. Refine your selection to include only unique components and then attempt the deployment again.

Working with components in an InterChange Server repository

The InterChange Server Component Management view allows you to manage the repositories of the InterChange Servers you have registered, and to manage the components in those repositories.

Validating the components in the repository

As described in “InterChange Server modes” on page 41, InterChange Server has different restrictions regarding the integrity of the repository depending on the mode you start it in. Although a server running in design mode allows the repository to be inconsistent so that you can add components as they become available, a server running in production mode requires that all references and dependencies among the components be resolved. When you migrate from a design-mode server to a production-mode server, then, you should validate the repository beforehand to make sure that the repository will be in a consistent state.

Do the following to validate the repository using System Manager:

1. In the InterChange Server Component Management view, right-click the server instance whose repository you want to validate and choose **Validate Repository** from the context menu.
2. If the repository is in a consistent state, an informational prompt is displayed.

   Click **OK** to close the “Validate Repository” prompt.

If there are unresolved dependencies in the repository then the “Missing Dependencies” dialog is displayed, as shown in Figure 22. Click **Finish** to close the “Missing Dependencies” dialog.

Note: You must resolve any missing dependencies before the server can start in production mode. For more information about production mode, see “Production mode” on page 42.

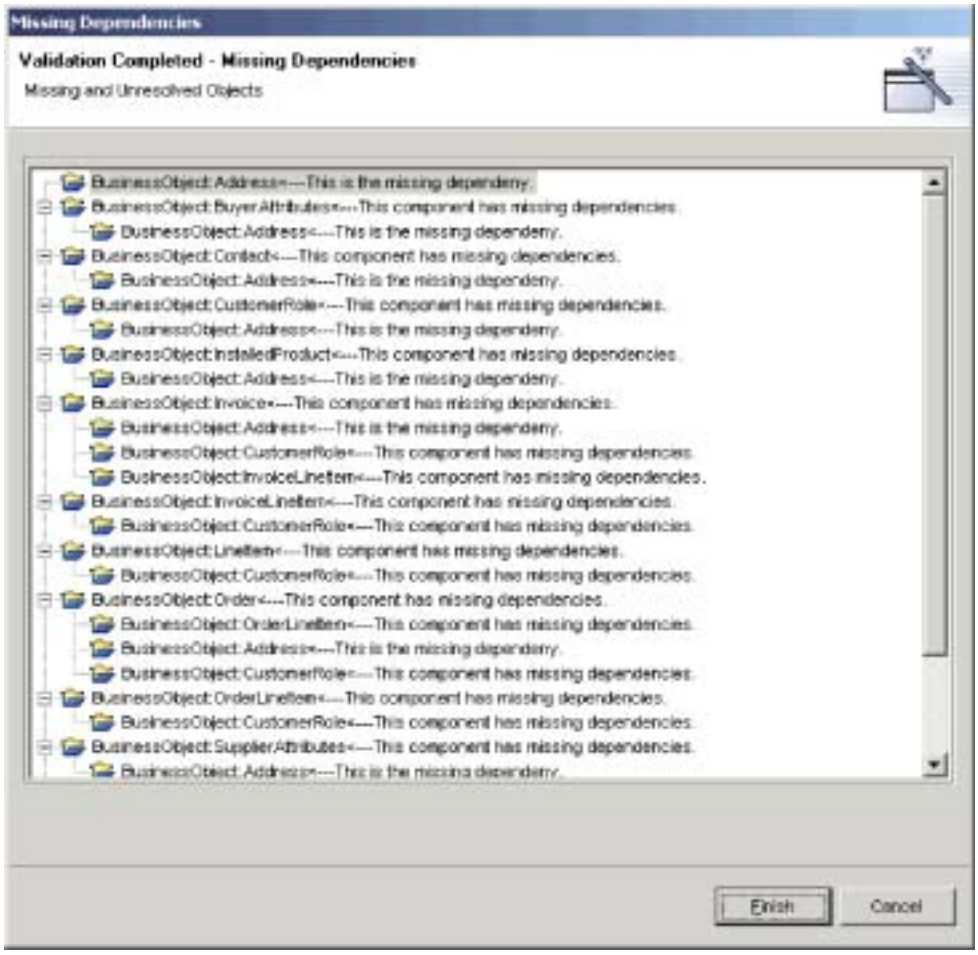

Figure 22. Unresolved dependencies in the repository

Validating database connection pools

To validate a database connection pool, right-click the pool in the InterChange Server Component Management view and choose **Validate database connection** from the context menu.

For more information about database connection pools, see Chapter 8, “Configuring database connection pools,” on page 155.

Managing component states in the repository

You can use the InterChange Server Component Management view to manage the states of components in the repository of an InterChange Server instance.

This is very useful when you are developing an interface because you have to test the components you are developing. You will frequently test a component and find

that you must change it. Then you will deploy the modified component to the server and test it again. To deploy a component to the server, however, it must be in an inactive state, so you must stop the component prior to deployment. Since you can develop, deploy, and test an interface all within the workbench, it is very convenient to also be able to manage the component states from it as well.

To change the state of a component in the InterChange Server Component Management view, right-click the component in the navigation tree and choose the desired state from the context menu. Table 6 on page 76 lists the state operations that you can apply for each of the component types that have runtime states. For more information on the behavior of the system when components are in different states, see the *System Administration Guide*.

Table 6. State operations available for components in System Manager

Component type	State operation
Collaboration object	• Start • Stop • Pause • Shut Down
Connector	• Start • Stop • Pause • Shut Down • Boot
Map	• Start • Stop
Relationship	• Start • Stop

You can also select multiple components, right-click one of the components, and apply state changes to all of them at the same time. This is very useful when you must deploy a number of components at the same time, because you do not have to perform a separate state-management operation for each one. You can use standard Windows selection techniques to select multiple user projects for deployment at the same time, such as the following:

- Hold down **Shift** to select contiguous items.
- Hold down **Ctrl** to select non-contiguous items.

Note: If you have any components that do not have states selected when you try to perform a state management operation on a group of selected components, then the state management options will not be displayed in the context-sensitive menu. For instance, if you want to stop several connectors, but accidentally select a business object definition as well, then the state management operations will not be exposed in the context-sensitive menu.

To modify the state all of the components of a particular type, right-click the folder for the component type in the InterChange Server Component Management view and select **Start All** or **Stop All** from the context menu.

It is recommended that you use System Monitor to manage components in a production installation. System Monitor offers greater flexibility and is a dedicated administration tool. For more information on using System Monitor, see the *System Administration Guide*.

Modifying component properties in the repository

Maps, connectors, and collaboration objects have runtime properties that you can modify to change the behavior of those components. Follow the instructions in the following sections to modify the properties for particular components:

- "Modifying collaboration object properties" on page 77
- "Modifying map properties" on page 77
- "Modifying connector properties" on page 78

Modifying collaboration object properties

Do the following to modify the properties of a collaboration object:

1. Right-click the collaboration object whose properties you want to modify in the InterChange Server Component Management view and select **Properties** from the context menu.
2. Set the properties on the "Collaboration General Properties" and "Properties" tabs to the desired values.

   For more information about collaboration object properties, see Chapter 9, "Configuring collaboration objects," on page 165.

   Figure 23 shows the "Properties" dialog for collaboration objects:

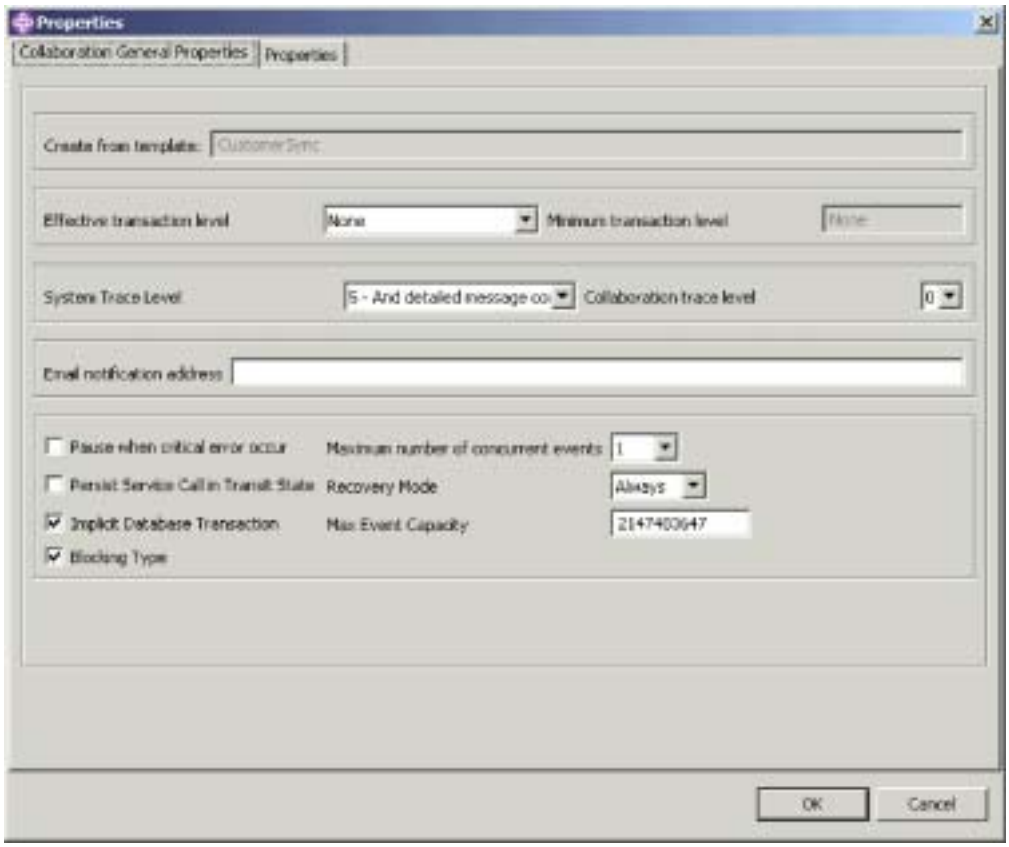

Figure 23. Modifying collaboration object properties

3. Click **OK**.

Modifying map properties

Do the following to modify the properties of a map:

1. Right-click the map whose properties you want to modify in the InterChange Server Component Management view and select **Properties** from the context menu.
2. Set the properties on the "Maps Property Page" dialog to the desired values.

   For more information about map properties, see the *Map Development Guide.*

   Figure 24 on page 78 shows the "Maps Property Page" dialog:

Figure 24. Modifying map properties

3. Click **OK**.

Modifying connector properties

Do the following to modify the properties of a connector:

1. Right-click the connector whose properties you want to modify in the InterChange Server Component Management view and select **Properties** from the context menu.
2. Set the properties on the "Connector Standard Properties", "Associated Maps", and "Resources" tabs to the desired values. For more information about connector properties, see Chapter 7, "Configuring connectors," on page 125.

   Figure 25 on page 79 shows the "Properties" dialog for connectors:

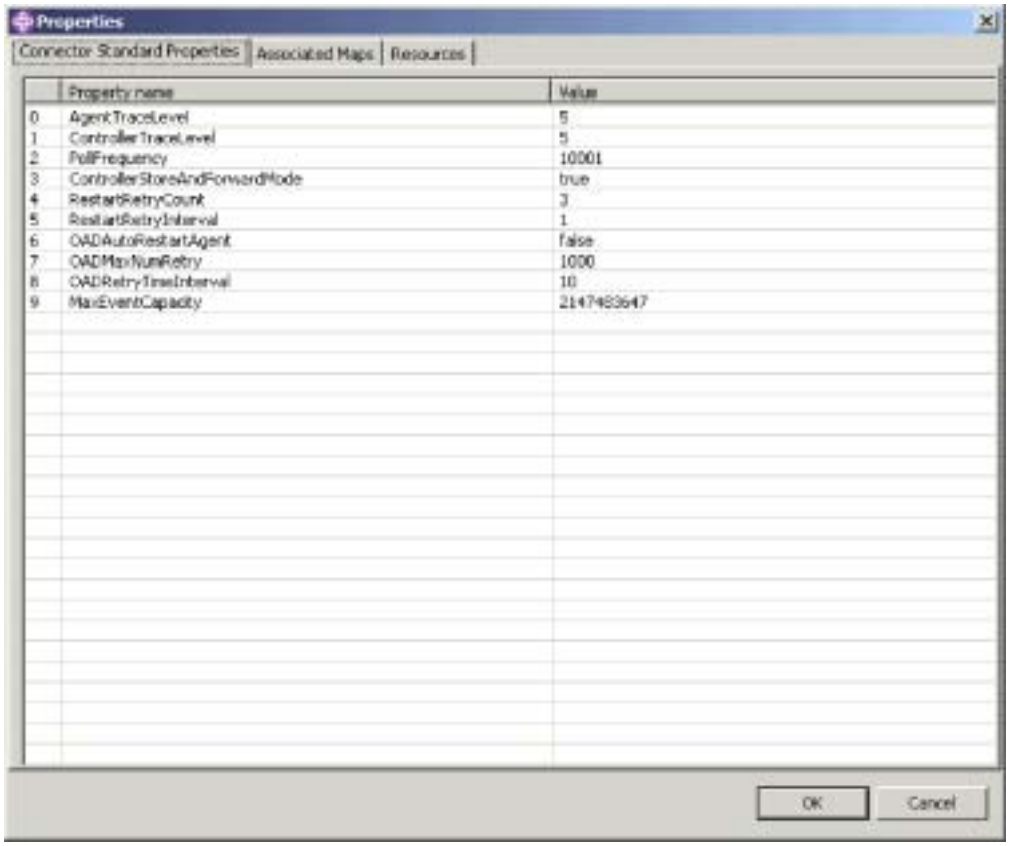

Figure 25. Modifying connector properties

3. Click **OK**.

Compiling components in the repository

Collaboration templates must be compiled for the collaboration objects that are based on them to run, and maps must be compiled to run as well. You can deploy collaboration templates and maps to an InterChange Server running in design mode without compiling them, but must ensure that they are compiled to test them.

To compile a collaboration template in the server, right-click the template in the InterChange Server Component Management view and choose **Compile** from the context menu.

To compile all of the collaboration templates in the server, right-click the **Collaboration Templates** folder in the InterChange Server Component Management view and choose **Compile All** from the context menu.

To compile a map in the server, right-click the map in the InterChange Server Component Management view and choose **Compile** from the context menu.

To compile all of the maps in the server, right-click the **Maps** folder in the InterChange Server Component Management view and choose **Compile All** from the context menu.

To compile a map along with all of its submaps, right-click the map in the InterChange Server Component Management view and choose **Compile with submaps** from the context menu.

The "Console" view displays messages about the success or failure of compilation attempts.

Deleting components from the repository

As you develop a business integration system, the design of the system frequently changes. New requirements and environmental differences commonly result in the redesign of an interface. You will want to make sure that the repository contains only the components it needs to solve business problems according to the current design, and that will sometimes mean that you have to delete components that belong to designs that have changed.

Deleting the entire repository

To delete all of the components in a server repository using System Manager do the following:

1. In the InterChange Server Component Management view, right-click the server instance whose repository you want to delete and choose **Delete Repository** from the context menu.
2. At the "Delete repository confirmation" prompt click **OK**.
3. At the next "Delete repository confirmation" prompt click **OK**.

Deleting components using the component browser

Do the following to delete components from an InterChange Server repository by using the component browser:

1. In the InterChange Server Component Management view, select the components you want to delete in the browser.

   You can use standard Windows techniques for selecting multiple items, such as holding **Shift** to select multiple contiguous items, and holding **Ctrl** to select multiple non-contiguous items.
2. To delete a component you must first make sure it is in an inactive state. Before attempting to delete any components, ensure that they are in an inactive state by stopping them.

   To stop a component using the InterChange Server Component Management view in System Manager, see "Managing component states in the repository" on page 74.

   To stop a component using System Monitor, see the *System Administration Guide*.
3. Do one of the following to delete the selected components:
   - Press **Delete**
   - Right-click one of the selected components and choose **Delete** from the context menu
   - Select **Edit > Delete** from the menu bar

   Figure 26 on page 81 shows System Manager when the component browser is used to delete components.

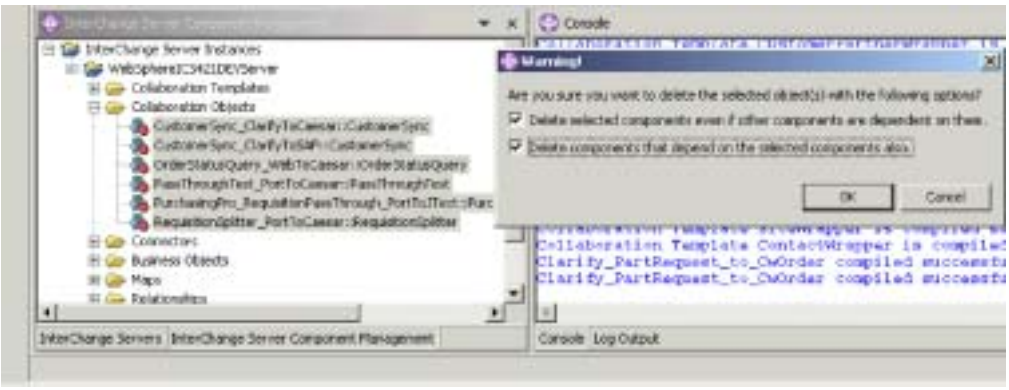

Figure 26. Deleting components using the component browser

4. Do the following when presented with the dialog:
   - Enable the **Delete selected components even if other components are dependent on them** checkbox to delete the component even if other components reference it.

     For instance, a business object cannot be deleted if it contains other business objects because they depend on it, even if those dependent components are also selected for deletion, unless this option is enabled.
   - Enable the **Delete components that depend on the selected components also** checkbox to also delete the dependencies of the selected components. For more information about dependencies, see "Dependencies and references" on page 82.
5. Click **Finish**.

   If the delete operation is successful then the wizard exits.

   If the delete operation failed, click **OK** to close the error prompt, then resolve the problem and try the operation again. The error information in the prompt can be very difficult to use in troubleshooting, so it is recommended that you use the server logging output to determine the cause of the failure.

Deleting components using the Server Object Delete Wizard

Do the following to use System Manager to delete components from an InterChange Server repository by using the Server Object Delete Wizard:

1. In the InterChange Server Component Management view, right-click the server instance from whose repository you want to delete components and choose **Server object delete** from the context menu.
2. To delete a component you must first make sure it is in an inactive state. Before attempting to delete any components, ensure that they are in an inactive state by stopping them.

   To stop a component using the InterChange Server Component Management view in System Manager, see "Managing component states in the repository" on page 74.

   To stop a component using System Monitor, see the *System Administration Guide*.
3. Do the following to select which components you want to delete:
   - Enable the checkbox for the server instance to delete all of the components in the repository.
   - Expand the folder for the server instance and enable the checkbox for a component type to delete all objects of that type.

- Select the folder for a component type to populate the right-hand pane with a list of the components of that type in the repository and then enable the checkbox for individual components.

4. Enable the **Force remove even if other objects depend on this object** checkbox to delete the component even if other components reference it.

   For instance, a business object cannot be deleted if it contains other business objects because they depend on it, even if those dependent components are also selected for deletion, unless this option is enabled.

5. Enable the **Delete dependent objects also** checkbox to also delete the dependencies of the selected components. For more information about dependencies, see "Dependencies and references."

   Figure 27 shows the "Server Objects Delete Wizard".

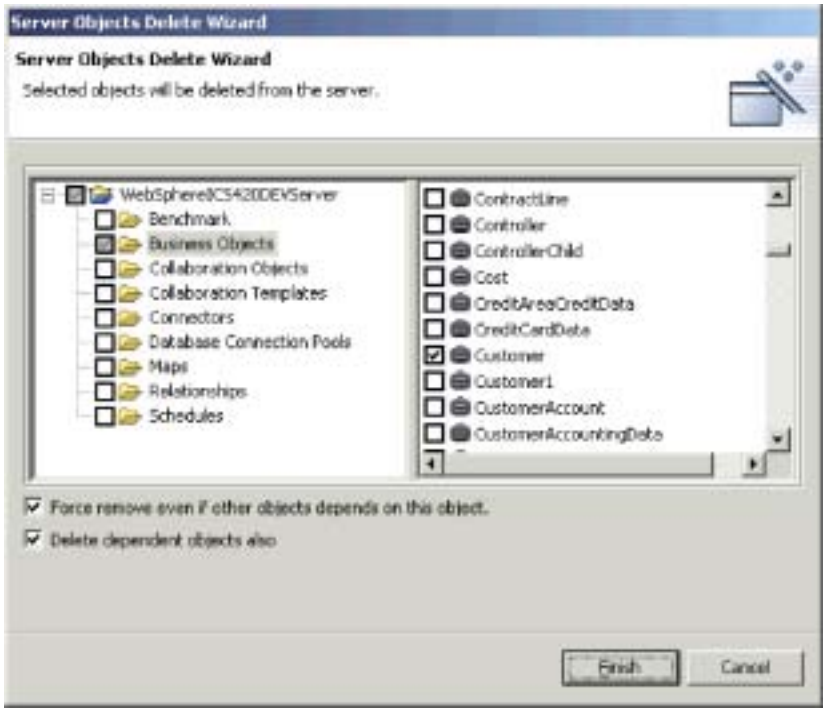

Figure 27. Deleting objects from the server

6. Click **Finish**.

   If the delete operation is successful then the wizard exits.

   If the delete operation failed, click **OK** to close the error prompt, then resolve the problem and try the operation again. The error information in the prompt can be very difficult to use in troubleshooting, so it is recommended that you use the server logging output to determine the cause of the failure.

Dependencies and references

Integration components depend upon one another to perform their roles in the business integration system. For instance, business object definitions can contain other business object definitions as children, collaboration templates have business object definitions associated with their ports, and connectors have maps associated with their supported business objects. These dependencies must be satisfied for the system to function properly, and when InterChange Server is started in production

mode it checks to make sure that all dependencies and references are resolved. If it detects any unresolved dependencies or references then InterChange Server fails to start in production mode.

The terms **dependencies** and **references** are used to describe the relationship between components, depending on the context. For example, a connector definition requires the business object definitions it supports and the maps associated with those business object definitions to exist so that it can exchange data with InterChange Server. In this context, the business object definitions and maps are **dependencies** of the connector definition. If you view the same relationship between the business object definition and the connector definition, but in the context of the business object, the connector is one of the **references** of the business object definition—along with any other business objects that contain it as a child, any maps that transform it, any collaboration templates that support it for their port definitions, and so forth.

Table 7 specifies the components that can be dependencies and references for each component type.

Table 7. Integration component dependencies and references

Component	Dependencies	References
Business objects	• Business objects	• Business objects • Maps • Connectors • Collaboration templates • Collaboration objects
Relationships	• Business objects	None
Connectors	• Business objects • Maps	• Collaboration objects
Maps	• Business objects • Maps	• Connectors • Collaboration objects
Collaboration templates	• Business objects	None
Collaboration objects	• Business objects • Maps • Connectors • Collaboration templates • Collaboration objects	• Collaboration objects

Showing dependencies and references

You can use System Manager to show the dependencies and references for an integration component.

To show the dependencies for a component, right-click it in System Manager and select **Show Dependencies** from the context menu. The “Dependency Tree” wizard appears. For more information, see “Using the Dependency Tree” on page 50.

To show the references for a component, right-click it in System Manager and select **Show References** from the context menu. The “Object References” window appears.

Dependencies that cannot be detected by the system

There are some dependency and reference relationships that cannot be automatically detected and enforced by the system. Table 8 describes the components for which the system cannot determine dependencies and references.

Although the system is unable to detect these dependencies and references, they must still be satisfied for the system to operate properly. You must maintain accurate project documentation to make sure that when you deploy an interface from one environment to another you include any of these extra dependencies.

Table 8. Components for which the system cannot determine dependencies and references

Component	Explanation
Relationships	Maps and collaboration templates can use relationships, but they do so through APIs in Java code. The system cannot parse all the code in all maps and collaboration templates to detect the use of relationships.
Database connection pools	Maps and collaboration templates can use database connection pools, but they do so through APIs in Java code. The system cannot parse all the code in all maps and collaboration templates to detect the use of database connection pools.
Data handlers	Connectors, as well as other components, can use data handlers to format data. Data handlers are not managed by the system at all, and it therefore cannot detect them.
any component used exclusively in code	Although the system can determine that a map depends upon a business object when you have drag-and-dropped the business object definition into the map as a source or destination, it cannot detect any use of components that occurs exclusively through custom Java code. If you instantiate a new business object or manually invoke a map in a map or collaboration, the system is not able to detect that dependency.
Benchmarks	The system does not currently support the detection of dependencies and references for benchmarks.
Schedules	The system does not currently support the detection of dependencies and references for schedules.

Standard operations available for multiple workbench resources

Although many of the tasks you perform in the workbench are sensitive to the particular resource you are working with or the context in which you are working, there are many operations that affect all resources the same way. This section describes the tasks you can perform in the workbench that work similarly for all resources.

Adding projects to the workspace from source code control

You can add integration component libraries and user projects to your workspace from a source code control system such as Rational ClearCase.

For more information about how to do so, see the documentation for the source code control system plug-in.

For an example of how to do so using ClearCase, see “Adding a project to the workspace” on page 21.

Cutting, copying, and pasting resources

You can cut, copy, and paste resources both in System Manager and in the file system.

To cut, copy, or paste a user project, integration component library, integration component, shortcut, or folder in System Manager, right-click the resource and choose the desired menu item. If you copy integration components, you cannot paste them into the same library (for instance, to copy a business object definition and paste it into the same library to specify a different name and use it as a template). You can, however, open the component definition in its designer tool and do a "Save as" operation to save it with a new name to the same library.

To cut, copy, and paste integration components or shortcuts in the file system, launch Windows Explorer and navigate to the appropriate subdirectory within the project directory, copy the file that shares the name of the component, and paste it into the appropriate subdirectory within the destination project directory.

If you use cut, copy, and paste operations in the file system, you must refresh the integration component library or user project in System Manager in order to see the newly added resources. For more information, see "Refreshing resources."

You cannot just cut, copy, and paste whole user projects or integration component libraries, because there are meta-data references maintained in the workbench that specify which user projects and libraries exist. Copying a folder into the workspace directory does not update those meta-data references. You can, however, create a new user project or library in System Manager to satisfy the meta-data entries, and then paste the folders of component definitions into the directory for the new library or user project.

Refreshing resources

If you add component definitions to a library or shortcuts to a user project by cutting, copying, and pasting files in the file system (as described in "Modifying map and collaboration object properties in integration component libraries" on page 60) then you must refresh the library in System Manager so that the changes are reflected.

To refresh an integration component library or user project, right-click it in System Manager and choose **Refresh View** from the context menu.

Deleting resources

Do the following to delete a workbench resource:

1. Either right-click the resource in System Manager and choose **Delete** from the context menu or select the resource in System Manager and press the **Delete** key.
2. When presented with the "Delete Component" dialog, click **OK**.

Note: If a component has dependencies then you will not be allowed to delete it.

Using Eclipse-based workbenches

This section describes some of the optional tasks you may want to perform in your tooling framework to make your experience with the tools more efficient.

Opening and closing perspectives

This section describes how to open and close perspectives.

Opening perspectives

Do the following to open a perspective in the workbench:

1. Select **Window > Open Perspective > Other** from the menu bar of your workbench.
2. At the "Select Perspective" dialog, choose the perspective you want to open and click **OK**.

Besides the System Manager, Integrated Test Environment, and Collaboration Debugger perspectives that you use to work with WebSphere InterChange Server integration components, you might also want to explore the perspectives described in the following sections:

Resource perspective: This perspective allows you to work directly with the meta-data files that represent project materials, such as integration component libraries. Figure 28 shows the Resource perspective open where the .cwt file that represents the ContactSync collaboration template has been opened from the Navigator view to display its contents in the Editor view.

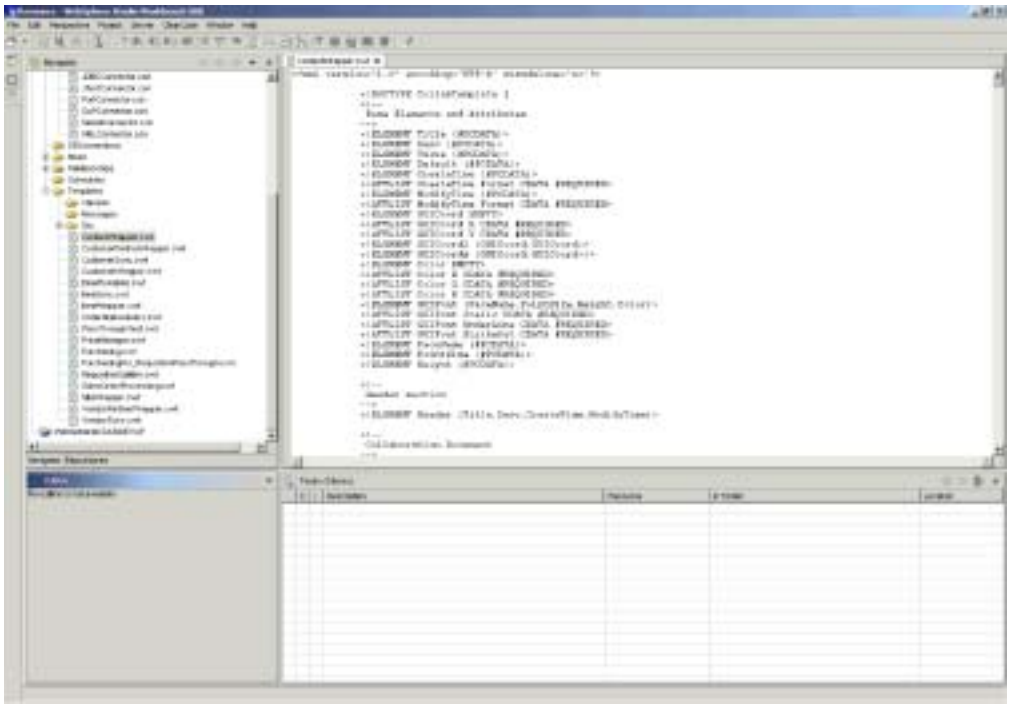

Figure 28. Resource perspective

Important: The meta-data files exposed through the Resource perspective define the WebSphere InterChange Server integration components. The System Manager perspective provides a way to work with those components safely through interfaces. If you manipulate meta-data files directly you run the risk of ruining the component definition. It is recommended that you only work with component meta-data files if you understand their structures very well, or in situations where you are interacting with Technical Support to troubleshoot a definition and are asked to.

Java Perspective: This perspective provides editors and views that assist with authoring Java files. Although you perform most Java programming in the designer tools, you occasionally have to write external components such as data handlers, or write utility classes. In these cases, this Java perspective can be very useful. shows the Java perspective.

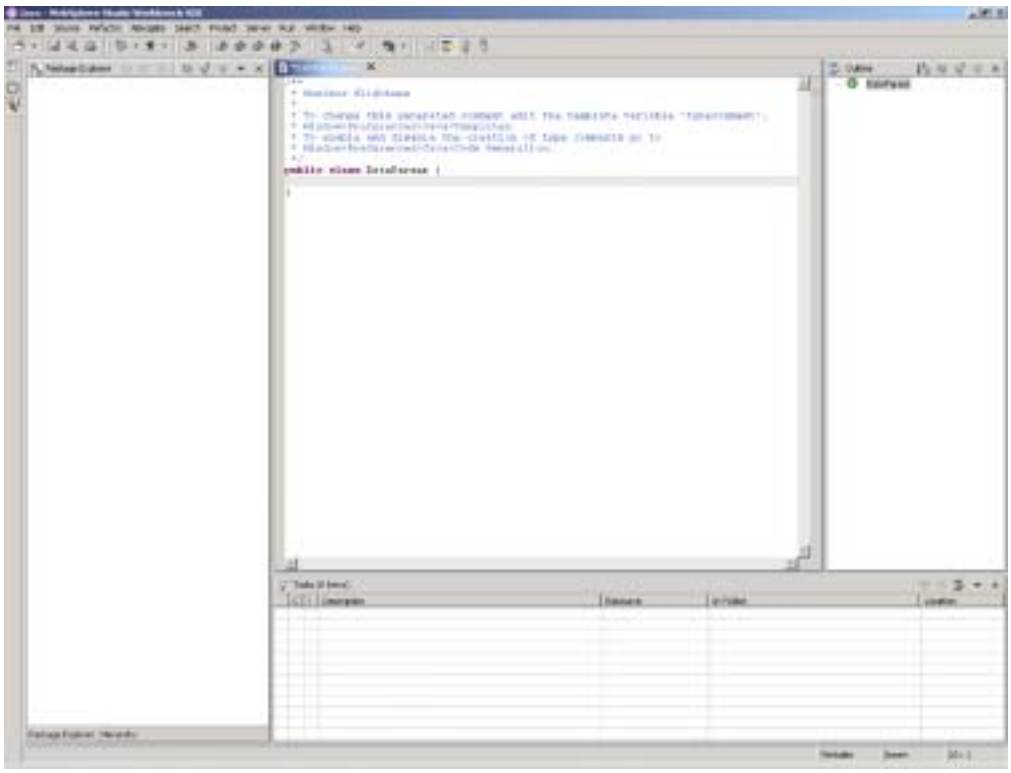

Figure 29. Java perspective

Closing perspectives

You can do the following to close perspectives:

- Select **Window > Close Perspective** to close the currently active perspective.
- Select **Window > Close All Perspectives** to close all currently open perspectives
- Right-click the icon for a perspective in the perspective shortcut bar and choose **Close** from the context menu to close that perspective.
- Right-click the icon for a perspective in the perspective shortcut bar and choose **Close All** from the context menu to close all open perspectives.

Showing and closing views

You can control the panes that are displayed in the WebSphere WorkBench and WebSphere Studio Application Developer Integration Edition perspectives.

Showing views

Do the following to show a view:

1. Select **Window > Show View > Other**.
2. Expand the folder for the view group, such as **ICS Control View**.
3. Select the particular view, such as **InterChange Server Component Management**.
4. Click **OK**.

Closing views

To close a view, do one of the following:

- Right-click the title bar of the view and choose **Close** from the context menu.
- Click the close button in the title bar of the view.

Customizing perspectives

You can customize a perspective to include the perspectives, views, wizards, and plug-in interfaces that you want so that you can minimize the number of times you have to open elements you use frequently and close elements you do not need. Do the following to customize the currently active perspective:

1. Select **Window > Customize Perspective** from the menu bar.
2. Click on the node you want to customize to expand it.
3. Enable and disable checkboxes for the node elements. Table 9 lists the customizable perspective nodes and the effect that enabling them has.

Table 9. Customizable perspective nodes

Customize perspective node	**Result**
File > New	Adds or subtracts items from the **File > New** menu.
Window > Open Perspective	Adds or subtracts perspectives from the **Window > Open Perspective** menu.
Window > Show View	Adds or subtracts views from the **Perspective > Show View** menu.
Other	Adds or subtracts from the menu bar and toolbar. For instance, you must enable the **ClearCase** checkbox to display the **ClearCase** menu.

Saving perspectives

You can save a perspective configuration to preserve customizations you have made. Do the following to save a perspective:

1. Select **Window > Save Perspective As** from the tooling framework menu bar.
2. Type a name for the perspective in the **Name** field.
3. Click **OK**.

Setting the default perspective

By default your tooling framework opens to the Resource perspective. If you primarily use the tooling framework to work with integration components, you may want to make one of the IBM WebSphere InterChange Server perspectives the default. Do the following to do so:

1. Select **Window > Preferences** from the menu bar of the tooling framework.
2. Expand the **Workbench** node.
3. Select the **Perspectives** node under the **Workbench** node.
4. Select the desired perspective (such as System Manager) from the **Available Perspectives** list.
5. Click **Make Default**.
6. Click **OK**.

Configuring System Manager preferences

To set your System Manager preferences, do the following:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Select **System Manager Preferences** and do the following to configure the available preference options:
   - If you do not want to be prompted when you delete a component from an integration component library, enable the checkbox for the component type in the "Do not confirm object deletion" pane.

     Note: Enabling the Unresolved Flows checkbox does not have any effect on whether or not you receive a prompt when you delete a flow from Flow Manager. For information on Flow Manager and how to configure preferences for it, see the *System Administration Guide*.
   - Enable the **Deep Copy** checkbox if you want to copy a component's dependencies along with the component when you perform a copy operation on it.

     If **Deep Copy** is enabled and you copy a business object definition from one library to another, then all of the child business objects it contains are copied as well, for example. If **Deep Copy** is not enabled, however, and you copy a business object definition from one library to another, then only the business object itself is copied.

     For more information on dependencies, see "Dependencies and references" on page 82.
   - Type the name and path of a file in the **Log File** field, or click **Browse** to select a file. When errors occur in System Manager, the error information is written to the file specified. Type a number in the **Max Size** field to specify the maximum size of the log file in megabytes.
   - Click **Restore Defaults** to set the preferences elements to their default values.

   Figure 30 on page 90 shows the System Manager preferences interface.

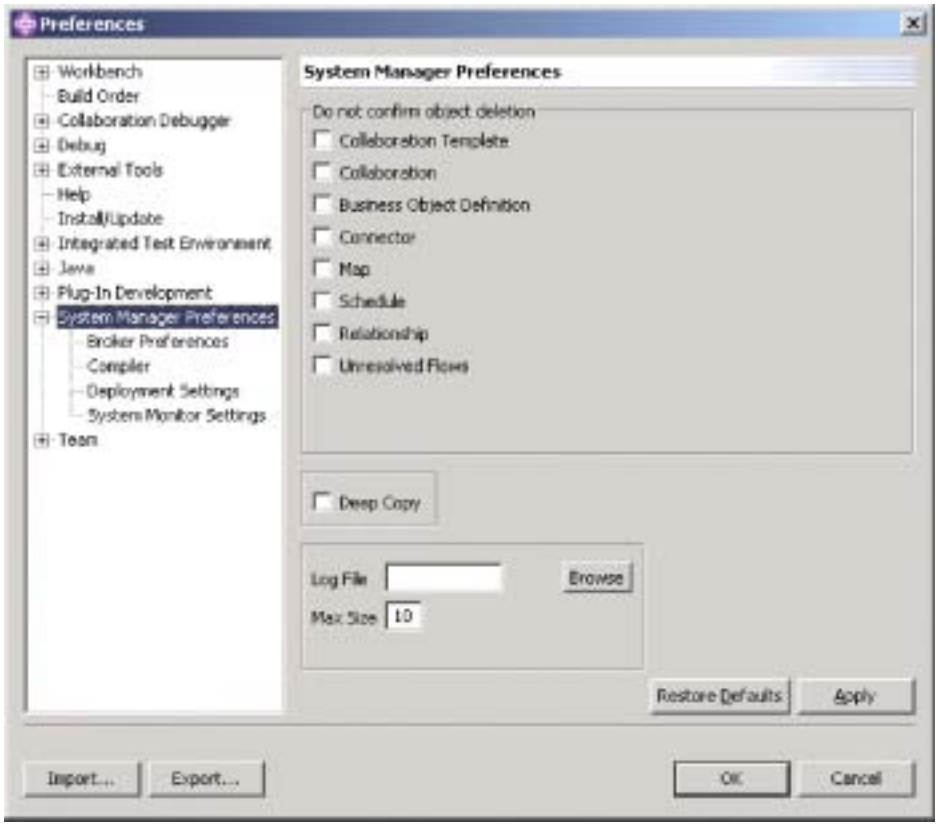

Figure 30. System Manager preferences

3. The “Broker Preferences” interface allows you to configure System Manager to work with the supported WebSphere message brokers.

   Do the following to set your broker preferences:

   a. Expand **System Manager Preferences** and then select **Broker Preferences**.

   b. Either type the full path to the WebSphere MQ Integrator Broker importer utility in the **Specify the Integrator broker importer path** field or click **Browse** to select the directory.

   c. Either type the full path to the WebSphere BI Message Broker importer utility in the **Specify the Message broker importer path** field or click **Browse** to select the directory.

   d. Either type the full path to the Eclipse workspace directory for WebSphere BI Message Broker in the **Specify the Message broker workspace directory** field or click **Browse** to select the directory.

   Figure 31 on page 91 shows the “Broker Preferences” interface:

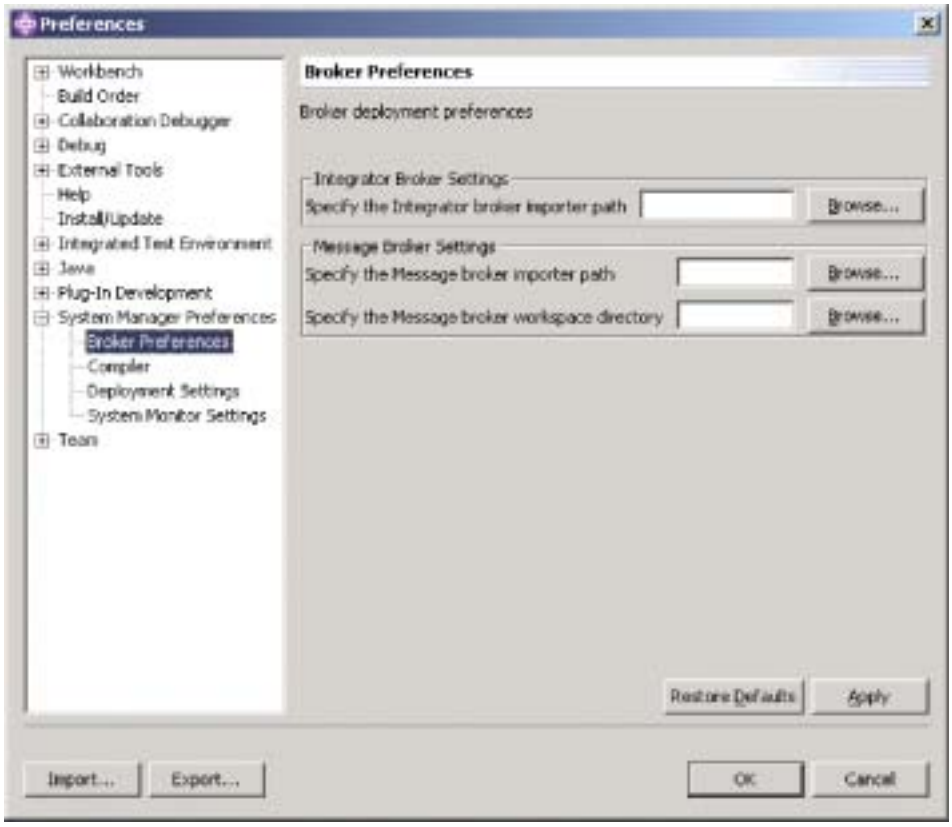

Figure 31. Broker preferences

4. The “Compiler” preferences interface allows you to specify the location of libraries required by maps or collaborations that you develop so that the compiler can locate them.

   Do the following to set your compiler preferences:

   a. Expand **System Manager Preferences** and then select **Compiler**.

   b. Click **New**.

   c. At the “Add Classpath” dialog, navigate to a library that you want to add to the compiler classpath, select the library, and click **Open**.

   Figure 32 on page 92 shows the “Compiler” preferences interface:

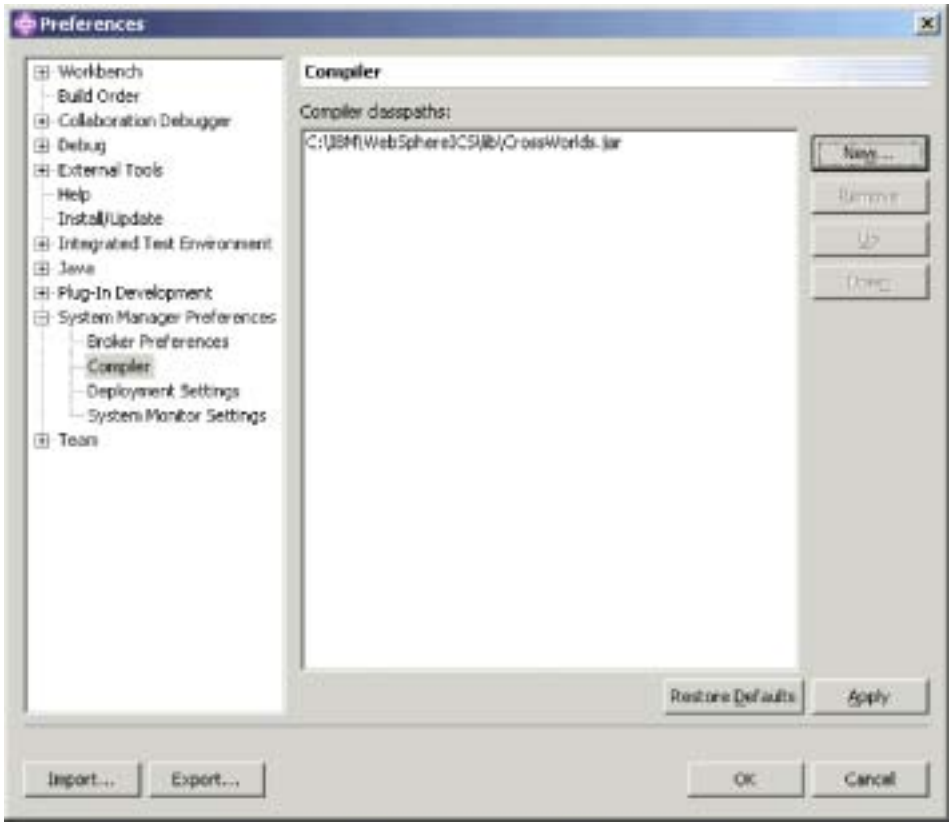

Figure 32. Compiler preferences

5. The “Deployment Settings” interface allows you to set options that are relevant when you deploy components to a server from System Manager.

   Do the following to set your deployment preferences:

   a. Expand **System Manager Preferences** and then select **Deployment Settings**.

   b. Enable the **Overwrite Components in Server during Drag and Drop deployment** checkbox.

   Figure 33 on page 93 shows the “Deployment Settings” interface:

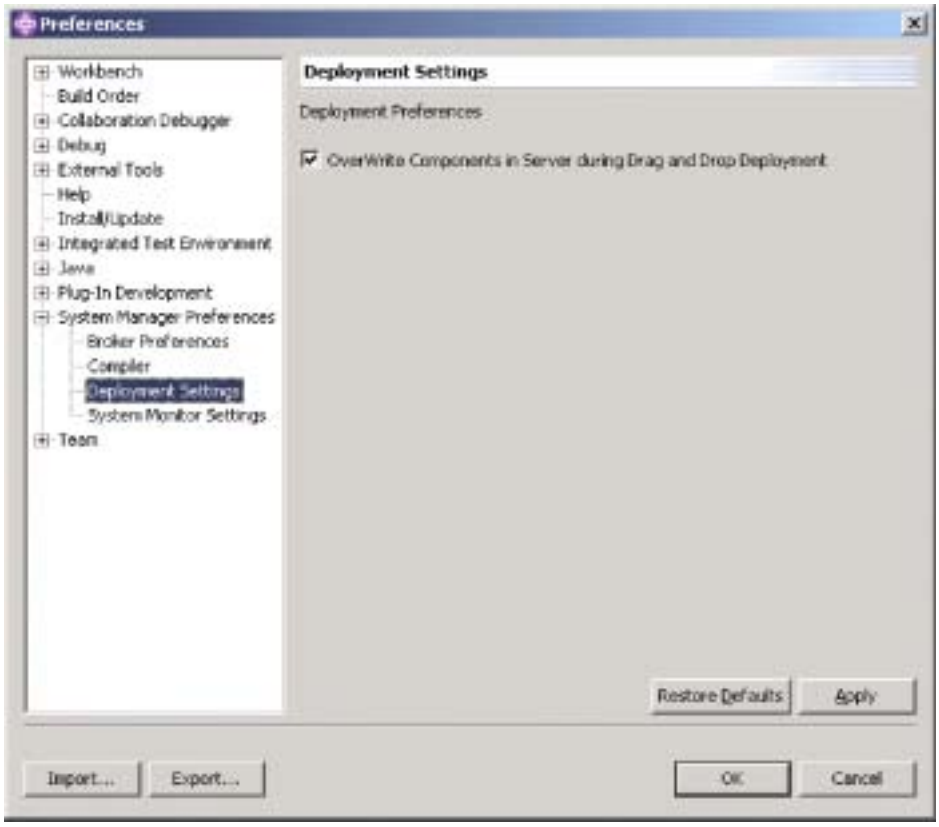

Figure 33. Deployment Settings interface

6. The “System Monitor Settings” interface allows you to configure the behavior of the monitoring features of the InterChange Server Component Management view.

   Do the following to set your System Monitor preferences:

   a. Expand **System Manager Preferences** and then select **System Monitor Settings**.

   b. Type the number of seconds in the **Poll Interval** field that you want System Manager to wait in between checking the system state to display information in the System View view.

   Figure 34 on page 94 shows the “System Monitor Settings” interface:

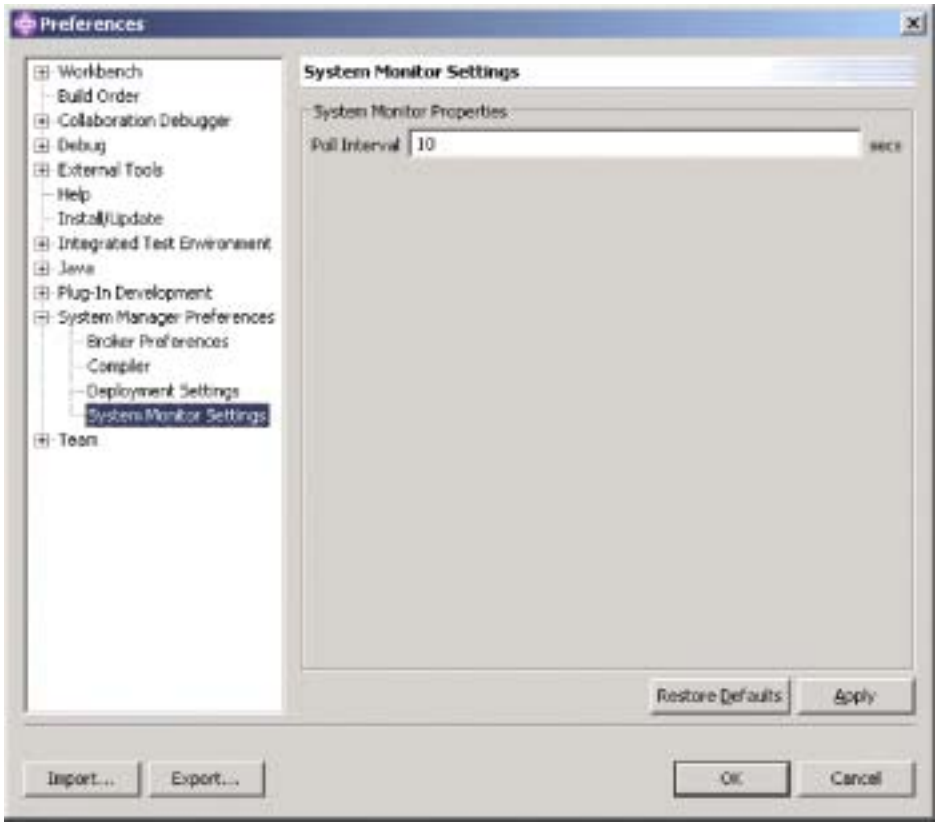

Figure 34. System Monitor Settings

For more information about the System View view, see the *System Administration Guide*.

7. Click **Apply** to save your preferences and continue working in the "Preferences" dialog, or click **OK** to save your preferences and exit the dialog.

Troubleshooting problems connecting to InterChange Server in System Manager

There are several common problems that result in System Manager being unable to connect to InterChange Server.

Investigate these possibilities to troubleshoot connection problems:

- InterChange Server must be running in order for System Manager to connect to it. Examine the logging output for InterChange Server to ensure that it has a logging statement which reads "<*server name*>" is ready", where <*server name*> is the name of your InterChange Server instance.
- The IBM Java Object Request Broker (ORB) must be running for clients such as the tools to communicate with it.
- You must specify the name of InterChange Server exactly as it exists. If you use the wrong case or leave out a single character when trying to register a server instance then System Manager will be unable to connect. You can be sure of the exact name of the server with the following techniques:
  - Examine the name exactly as it appears in whatever interface is used to start the server. On Windows, the server name is commonly supplied as a parameter to the `start_server.bat` batch file. The value is in the **Target** field

of the shortcut that invokes the batch file. On Unix platforms you manually specify the name of the InterChange Server instance.

  - Examine the name as it is displayed in the InterChange Server logging output. The log entry "<*server name*>" is ready" indicates that the server has started and by what name it can be identified.

- You must specify the correct user name, which is `admin`.
- You must specify the correct password. The default password is `null`, but it can be changed as described in "Changing the InterChange Server password" on page 42. If you cannot connect when specifying the password `null`, make sure that no other developers who work with the server instance have changed the password.

  If you have cached the user name and password, the password value sometimes gets corrupted. It will still appear as four asterisks, so you will not intuitively think that its value has changed. Delete the cached value in the **Password** field and type the password again.

Chapter 5. Configuring WebSphere InterChange Server

You must configure InterChange Server with environment-specific information for it to run. The configuration information is stored in XML format in a file named `InterchangeSystem.cfg`, which is stored in the product directory. The two tools you can use to configure InterChange Server are:

- InterChange Server Configuration Wizard

  This tool does not expose all of the configuration options for InterChange Server, but it can be used when the server is not running.

- System Manager

  This tool can configure many more options than the InterChange Server Configuration Wizard, but the server must be running and System Manager must be connected to it.

This chapter documents how to use InterChange Server Configuration Wizard and System Manager to configure a server instance. It contains the following sections:

Using the InterChange Server Configuration Wizard

Do the following to start the InterChange Server Configuration Wizard and update the configuration file:

1. Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere InterChange Server >IBM WebSphere InterChange Server Configuration Wizard**.
2. Set the fields of the wizard tabs to the desired values as described in the following sections:
   - "Configuring server properties using the wizard"
   - "Configuring WebSphere MQ properties using the wizard" on page 98
   - "Configuring database properties using the wizard" on page 99
3. Click **Apply**.
4. When presented with the "Changes Complete" dialog, click **OK**.
5. Click **Exit**.

Configuring server properties using the wizard

The "InterChange Server" tab of the configuration wizard allows you to change some basic properties of the server itself. Do the following to configure the InterChange Server properties:

1. Click the "InterChange Server" tab.

   Figure 35 on page 98 shows the "InterChange Server" tab:

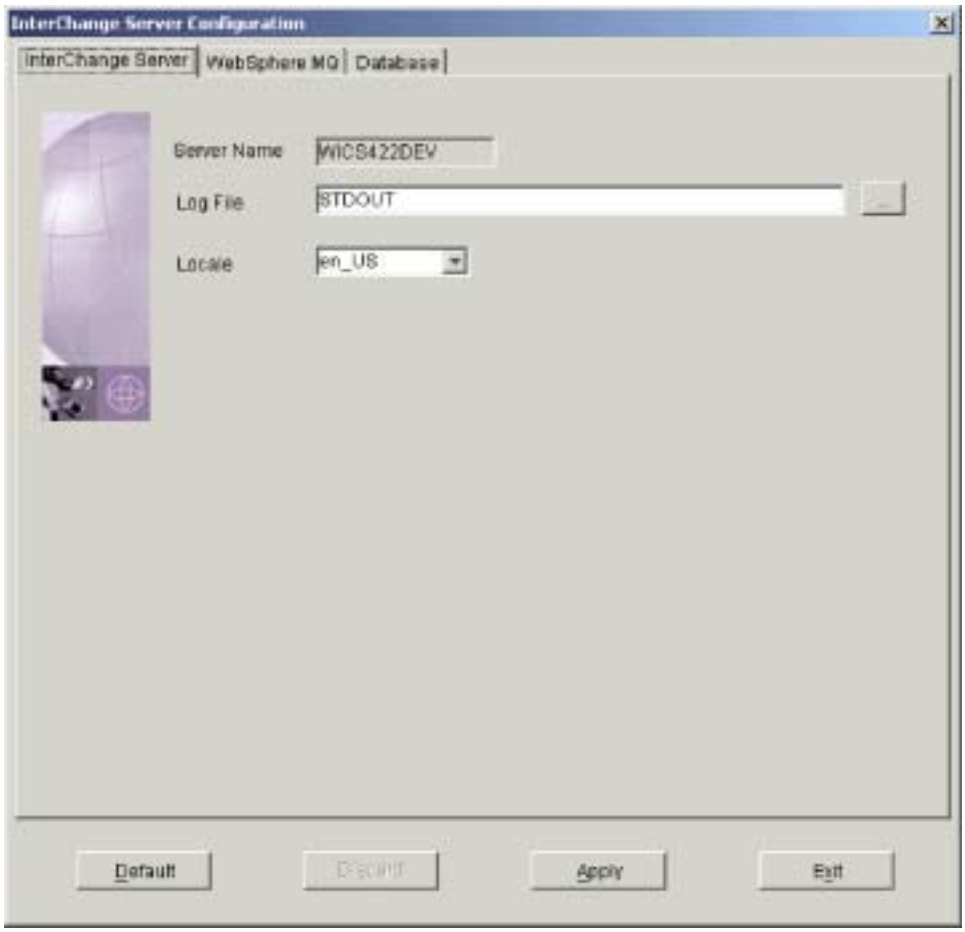

Figure 35. InterChange Server configuration tab

2. Type the name of the InterChange Server instance in the **Server Name** field.
3. Do one of the following to set the value of the **Log File** field:
   - Type the value `STDOUT` in the field to direct InterChange Server logging to the console.
   - Type the name and path of a file in the field or click **Browse** to select a file to which to InterChange Server logging should be directed.

   Note: Consider the information in "Configuring logging in a development environment" on page 16 when setting this property.
4. Select the desired locale from the **Locale** drop-down menu.

Configuring WebSphere MQ properties using the wizard

The "WebSphere MQ" tab of the configuration wizard allows you to change properties that specify how InterChange Server communicates with the WebSphere MQ persistent messaging server. Do the following to configure the WebSphere MQ properties:

1. Click the **WebSphere MQ** tab.

   Figure 36 on page 99 shows the "WebSphere MQ" tab:

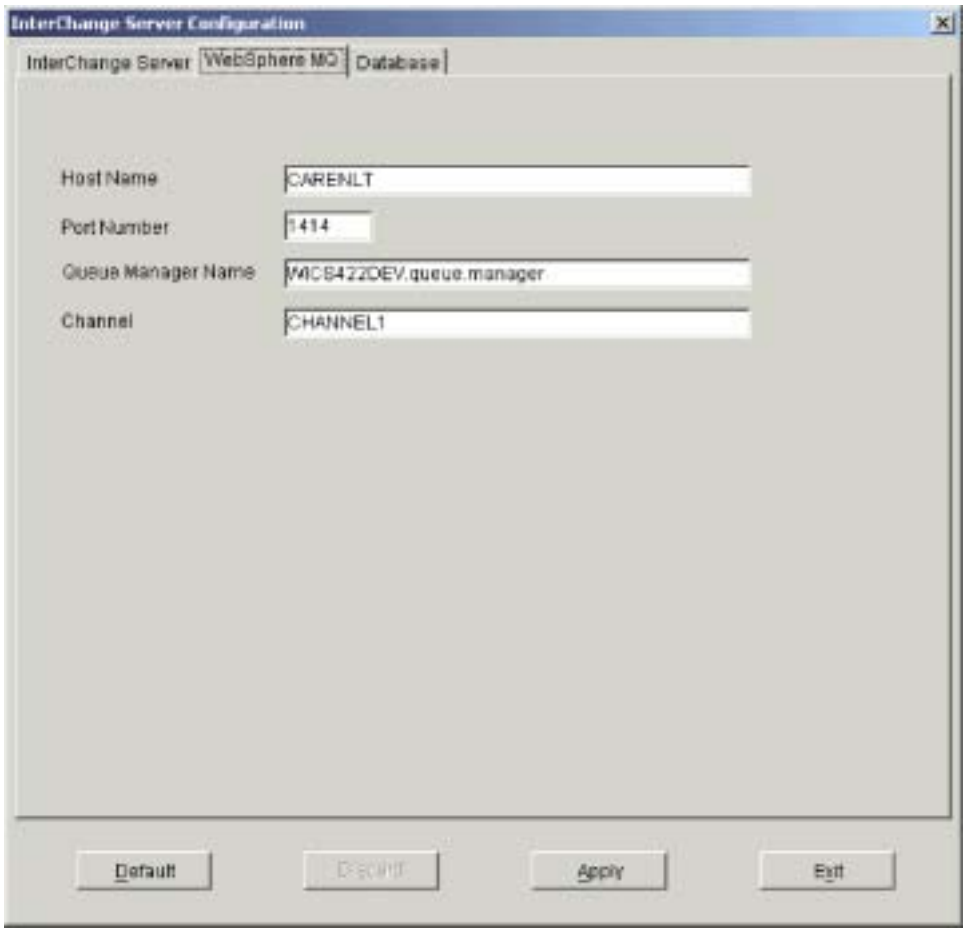

Figure 36. WebSphere MQ configuration tab

2. Type the name of the computer on which the WebSphere MQ server is installed in the **Host Name** field.
3. Type in the **Port Number** field the port over which WebSphere MQ allows clients to communicate with the server.

   The default port is 1414.
4. Type in the **Queue Manager Name** field the name of the queue manager that has been created to contain the queues on which InterChange Server will persistently store messages for the flows it processes.
5. Type in the **Channel** field the name of the channel over which clients of WebSphere MQ communicate with the WebSphere MQ server. The default value is `CHANNEL1`. Consult with the WebSphere MQ administrator at the site to determine if `CHANNEL1` can be used and if not, which channel can be used.

Configuring database properties using the wizard

The "Database" tab of the configuration wizard allows you to change properties that specify the databases InterChange Server uses to store repository, event management, and transaction management data. Do the following to configure the database properties:

1. Click the **Database** tab.

   Figure 37 on page 100 shows the "Database" configuration tab:

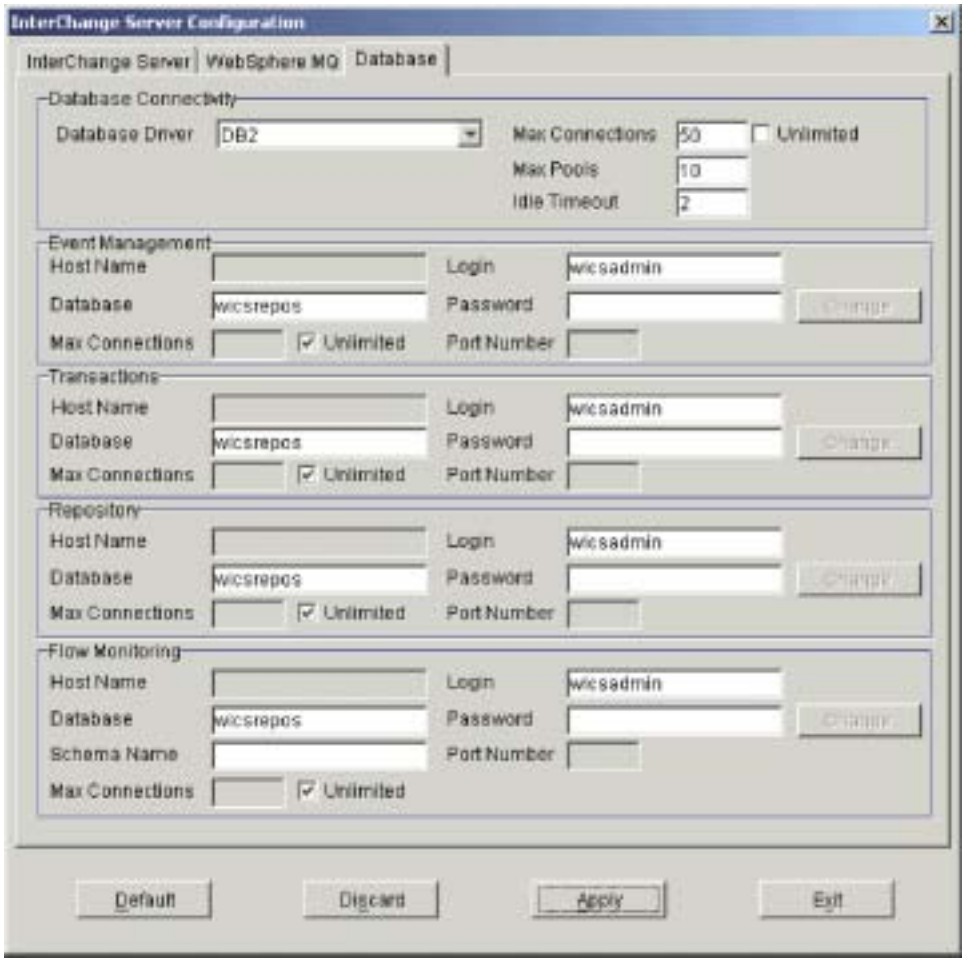

Figure 37. Database connectivity configuration tab

2. Select the appropriate value in the **Database Driver** drop-down menu—either `DB2`, `MQ SQL Server(Type 4)`, or `Oracle(Type 4)`—depending on the database vendor.
3. Type the maximum number of connections you want InterChange Server to make with the database server in the **Max Connections** field, or enable the **Unlimited** checkbox to allow InterChange Server an unlimited number of connections.
4. Type the maximum number of pools InterChange Server should establish to contain the database connections it caches in the **Max Pools** field.
5. Type the number of minutes you want a connection object to remain idle before being returned to the database connection pool for reuse in the **Idle Timeout** field.
6. Do the following in the "Event Management" pane to configure the database connectivity for the event management service:
   a. If you chose `MQ SQL Server(Type 4)`, or `Oracle(Type 4)` in the **Database Driver** drop-down menu, then type the name of the computer on which the database server resides in the **Host Name** field.
   b. Type the name of the database in the **Database** field.

c. Type the maximum number of connections you want InterChange Server to make with the specific database server in the **Max Connections** field, or enable the **Unlimited** checkbox to allow InterChange Server an unlimited number of connections.

d. Type the user name that should be used by InterChange Server to log in to the specified database in the **Login** field.

e. Type the password for the user name specified in step 6d in the **Password** field.

f. If you chose `MQ SQL Server(Type 4)`, or `Oracle(Type 4)` in the **Database Driver** drop-down menu, then you must type the port number through which clients communicate with the database server in the **Port Number** field.

7. Repeat step 6 on page 100 in the "Transactions" pane to configure the database connectivity for the transaction management service.
8. Repeat step 6 on page 100 in the "Repository" pane to configure the database connectivity for the repository service.
9. Do the following in the "Flow Monitoring" pane to configure the database connectivity for the flow monitoring service:

   a. If you chose `Oracle(Type 4)` in the **Database Driver** drop-down menu, then type the name of the computer on which the database server resides in the **Host Name** field.

      Note: Microsoft SQL Server is not a supported database for WebSphere Business Integration Monitor. If SQL Server is the database you are using for the other databases used by WebSphere InterChange Server, then you cannot use Flow Monitoring.

   b. Type the name of the database in the **Database** field.

   c. Type the name of the schema used by WebSphere Business Integration Monitor in the **Schema** field.

      If you are using Oracle for the flow monitoring database then the name of the schema is identical to the name of the user account. If you are using IBM DB2 for the flow monitoring database, however, the schema name can be different from the name of the user account, so you must specify the correct schema name.

   d. Type the maximum number of connections you want InterChange Server to make with the specific database server in the **Max Connections** field, or enable the **Unlimited** checkbox to allow InterChange Server an unlimited number of connections.

   e. Type the user name that should be used by InterChange Server to log in to the specified database in the **Login** field.

   f. Type the password for the user name specified in step 9e in the **Password** field.

   g. If you chose `Oracle(Type 4)` in the **Database Driver** drop-down menu, then you must type the port number through which clients communicate with the database server in the **Port Number** field.

   For more information about Flow Monitoring, see the *System Administration Guide*.

Changing database passwords

Do the following to change the password for the user name that InterChange Server uses to access one of the databases:

1. Click **Change** next to the **Password** field in the “Event Management”, “Transactions”, “Repository”, or “Flow Monitoring” panes.
2. Type the new password in the **New Password** field.
3. Type the new password again in the **Confirm Password** field.
4. Click **OK**.

Using System Manager

System Manager provides more flexibility when configuring InterChange Server, but the server must be running and System Manager must be connected to it.

Do the following to open the InterChange Server configuration editor in System Manager:

1. Start System Manager as described in “Starting System Manager” on page 34.
2. Connect System Manager to the InterChange Server instance as described in “Connecting to InterChange Server” on page 39.
3. Right-click the InterChange Server instance and choose **Edit Configuration** from the context menu.
4. Make the desired changes in the configuration tabs as described in the following sections:
   - “Viewing general properties using System Manager”
   - “Configuring database properties using System Manager” on page 103
   - “Configuring tracing levels using System Manager” on page 106
   - “Configuring logging and tracing properties using System Manager” on page 106
   - “Configuring e-mail notification properties using System Manager” on page 108
   - “Configuring miscellaneous properties using System Manager” on page 109
   - “Configuring environment variables using System Manager” on page 110
   - “Configuring WebSphere MQ properties using System Manager” on page 111
5. When you are done use the keyboard shortcut **Ctrl+S** to save the file.

Viewing general properties using System Manager

The “General” tab of the System Manager configuration file editor has the following read-only fields:

- The **Server name** field, which identifies the name of the InterChange Server instance.
- The **Server version** field, which identifies the version of IBM WebSphere InterChange Server software the server is running.
- The **Total memory** field, which identifies the amount of RAM on the computer running InterChange Server.
- The **Free memory** field, which identifies the amount of RAM that is available on the computer running InterChange Server.
- The **Startup time** field, which identifies the time at which the server instance started.
- The **Local Time Zone** field, which identifies the time zone in which the computer running System Manager resides.
- The **Server Time Zone** field, which identifies the time zone in which the computer running InterChange Server resides.

- The **Server locale** field, which identifies the locale of the computer on which InterChange Server is running.

Figure 38 shows the “General” tab of the System Manager configuration file editor:

Figure 38. Configuring InterChange Server general properties

Configuring database properties using System Manager

The “Database” tab of the configuration file editor allows you to change properties that specify the databases InterChange Server uses to store repository, event management, and transaction management data. Do the following to configure the database properties:

1. Click the **Database** tab.

   Figure 39 on page 104 shows the “Database” tab in the System Manager configuration file editor:

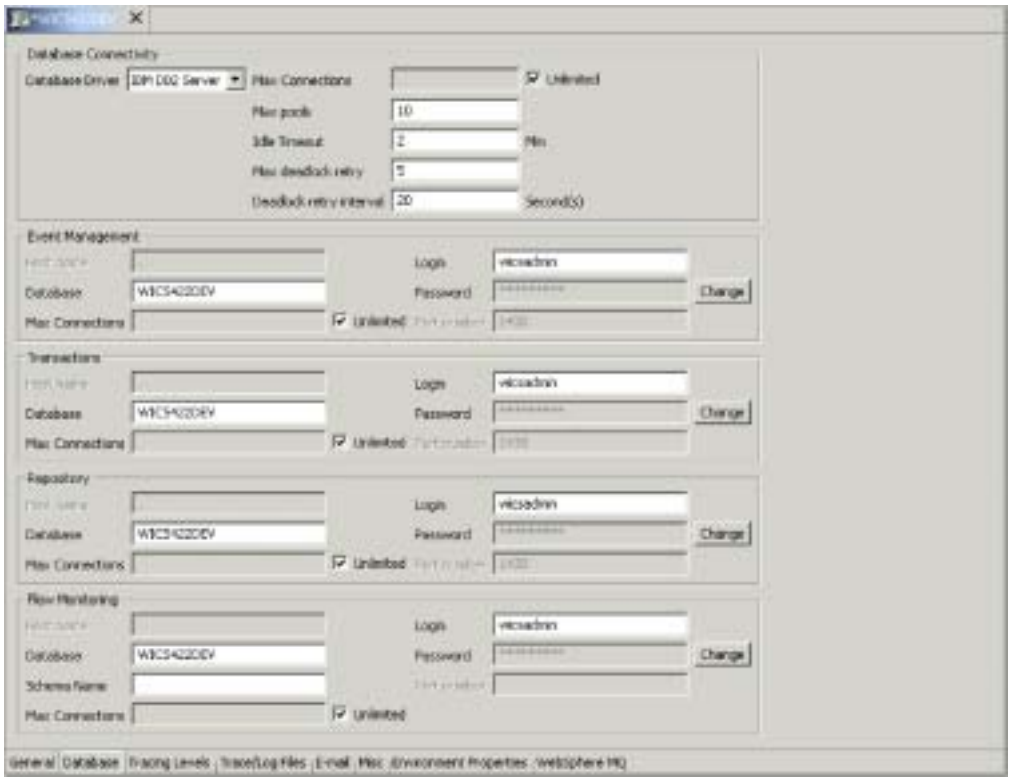

Figure 39. Configuring InterChange Server database properties

2. Select the appropriate value in the **Database Driver** drop-down menu—either `IBM DB2 Server`, `MQ SQL Server`, or `Oracle Server`—depending on the database vendor.
3. Type the maximum number of connections you want InterChange Server to make with the database server in the **Max Connections** field, or enable the **Unlimited** checkbox to allow InterChange Server an unlimited number of connections.
4. Type the maximum number of pools InterChange Server should establish to contain the database connections it caches in the **Max pools** field.
5. Type the number of minutes you want a connection object to remain idle before being returned to the database connection pool for reuse in the **Idle Timeout** field.
6. Type the number of times that the server should attempt to execute a deadlocked database transaction in the **Max deadlock retry** field.

   This field is related to the **Deadlock retry interval** field documented in step 7.

   For more information on the behavior of the server in the case of a database deadlock, see the *System Administration Guide*.
7. Type the number of seconds that the server should wait in between attempts to execute a deadlocked database transaction in the **Deadlock retry interval** field.

   This field is related to the **Max deadlock retry** field documented in step 6.

   For more information on the behavior of the server in the case of a database deadlock, see the *System Administration Guide*.
8. Do the following in the “Event Management” pane to configure the database connectivity for the event management service:

a. If you chose MQ SQL Server(Type 4), or Oracle(Type 4) in the **Database Driver** drop-down menu, then type the name of the computer on which the database server resides in the **Host name** field.

b. Type the name of the database in the **Database** field.

c. Type the maximum number of connections you want InterChange Server to make with the specific database server in the **Max Connections** field, or enable the **Unlimited** checkbox to allow InterChange Server an unlimited number of connections.

d. Type the user name that should be used by InterChange Server to log in to the specified database in the **Login** field.

e. Type the password for the user name specified in step 6d on page 101 in the **Password** field.

f. If you chose MQ SQL Server(Type 4), or Oracle(Type 4) in the **Database Driver** drop-down menu, then you must type the port number through which clients communicate with the database server in the **Port Number** field.

9. Repeat step 6 on page 100 in the "Transactions" pane to configure the database connectivity for the transaction management service.

10. Repeat step 6 on page 100 in the "Repository" pane to configure the database connectivity for the repository service.

11. Do the following in the "Flow Monitoring" pane to configure the database connectivity for the flow monitoring service:

    a. If you chose Oracle(Type 4) in the **Database Driver** drop-down menu, then type the name of the computer on which the database server resides in the **Host Name** field.

    Note: Microsoft SQL Server is not a supported database for WebSphere Business Integration Monitor. If SQL Server is the database you are using for the other databases used by WebSphere InterChange Server, then you cannot use Flow Monitoring.

    b. Type the name of the database in the **Database** field.

    c. Type the name of the schema used by WebSphere Business Integration Monitor in the **Schema** field.

    If you are using Oracle for the flow monitoring database then the name of the schema is identical to the name of the user account. If you are using IBM DB2 for the flow monitoring database, however, the schema name can be different from the name of the user account, so you must specify the correct schema name.

    d. Type the maximum number of connections you want InterChange Server to make with the specific database server in the **Max Connections** field, or enable the **Unlimited** checkbox to allow InterChange Server an unlimited number of connections.

    e. Type the user name that should be used by InterChange Server to log in to the specified database in the **Login** field.

    f. Type the password for the user name specified in step 11e in the **Password** field.

    g. If you chose Oracle(Type 4) in the **Database Driver** drop-down menu, then you must type the port number through which clients communicate with the database server in the **Port Number** field.

    For more information about Flow Monitoring, see the *System Administration Guide*.

Changing database passwords

Do the following to change the password for the user name that InterChange Server uses to access one of the databases:

1. Click **Change** next to the **Password** field in the "Event Management", "Transactions", or "Repository" panes.
2. Type the new password in the **New Password** field.
3. Type the new password again in the **Confirm Password** field.
4. Click **OK**.

Configuring tracing levels using System Manager

Do the following to set properties for flow tracing and subsystem tracing at the "Tracing Levels" tab:

1. Click the **Tracing Levels** tab.

   Figure 40 shows the "Tracing Levels" tab in the System Manager configuration file editor:

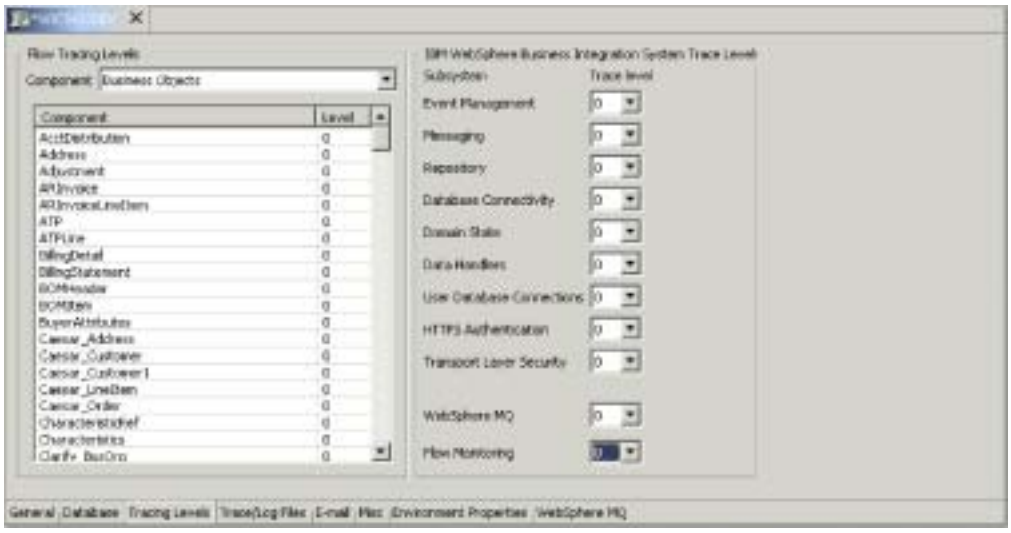

Figure 40. Configuring InterChange Server tracing levels

2. To set the flow trace level for a business object, select the desired value in the **Level** column for the particular business object in the "Flow Tracing Levels" pane.

   For more information on flow tracing, see the *System Administration Guide.*
3. To set the trace level for an InterChange Server subsystem, select the desired trace level from the drop-down menu associated with the subsystem.

   For information on what information is reported at each level for each subsystem, see the "Configuration Parameters" appendix of the *System Installation Guide for Windows or Unix*.

Configuring logging and tracing properties using System Manager

Do the following to configure the locations to which InterChange Server logs and traces runtime information:

1. Click the **Trace/Log Files** tab.

Figure 41 shows the "Trace/Log Files" tab in the System Manager configuration file editor:

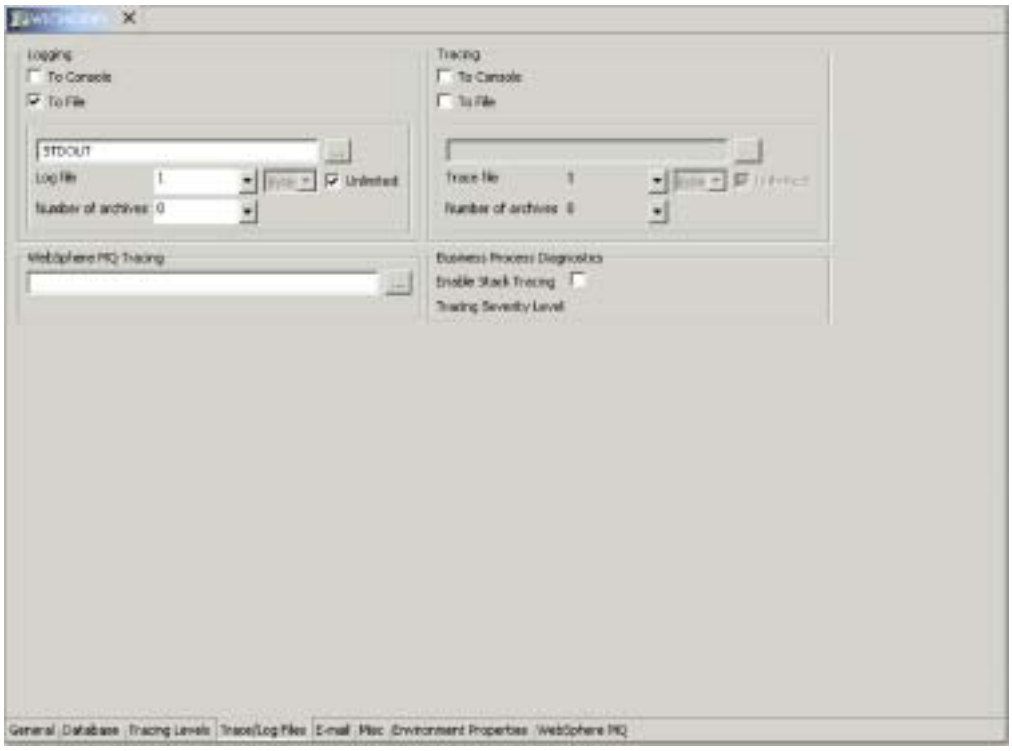

Figure 41. Configuring InterChange Server logging and tracing properties

2. Do the following in the Logging pane to configure how InterChange Server logs runtime information:
   - Enable the **To Console** checkbox if you want InterChange Server to output logging information to the console.
   - Enable the **To File** checkbox if you want InterChange Server to output logging information to a file. Do the following steps as well if you enable this checkbox:
     a. Type the name and path of the file to which output should be logged in the unlabeled text field, or click the browse button to select a file.
     b. Do one of the following to specify how the log files should be managed:
        – Click the **Unlimited** checkbox to have InterChange Server write information to the file specified in step 2a without a limit on the size of the file.

          Note: Be sure to manage the file well if you make this selection, because InterChange Server could potentially fill up the disk and cause a system failure.

        – Do the following to configure a log file archiving approach:
          1) Select **Byte**, **KB**, **MB**, or **GB** from the unlabeled drop-down menu to indicate whether you want to store the log and archive files in increments of bytes, kilobytes, megabytes, or gigabytes.
          2) Select a number from the **Log file** drop-down menu to indicate the number of bytes, kilobytes, megabytes, or gigabytes you want the log file to grow to before it is archived.

3) Select the number of archive files that should be kept before they are deleted in the **Number of archives** drop-down menu.

3. Repeat step 2 on page 107 for the “Tracing” pane.
4. To enable tracing for WebSphere MQ, type the name and path of the file to which output should be logged in the unlabeled text field in the “WebSphere MQ Tracing” pane, or click the browse button to select a file.
5. To use stack tracing, select **Enable Stack Tracing** and then select the desired severity level from the **Tracing Severity Level** drop-down menu.

   For more information about stack tracing, see *System Administration Guide*.

Configuring e-mail notification properties using System Manager

InterChange Server can send e-mail notifications when errors occur in the business integration system. By default the server uses the Java e-mail APIs, and can send e-mail notifications for a number of different subsystems. To enable the server to send e-mail notifications for errors experienced by individual collaborations, you must configure the system to use the WebSphere Business Integration Adapter for e-Mail. For detailed information on configuring the system for e-mail notification, see the *System Administration Guide*. To make the necessary modifications to the configuration file (which is just one step in the total number of tasks described in the *System Administration Guide*), do the following:

1. Click the **E-mail** tab.

   Figure 42 shows the “E-mail” tab in the System Manager configuration file editor:

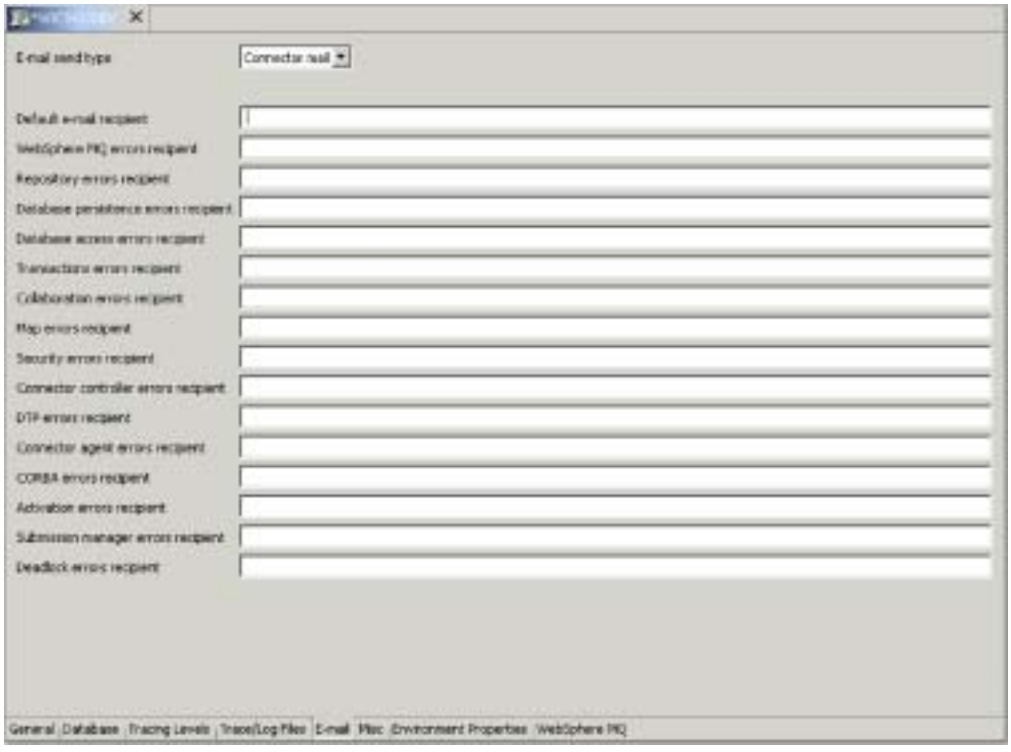

Figure 42. Configuring InterChange Server e-mail notification properties

2. Select `Java mail` or `Connector mail` from the **E-mail send type** drop-down menu depending on which e-mail notification mechanism you want to use.

Note: Keep in mind that collaborations will only be able to send e-mail notifications if you choose `Connector mail`.

3. If you chose `Java mail` in step 2 on page 108, type the name of the computer on which the SMTP server is installed in the **SMTP mail host** field.
4. Type a valid e-mail address or comma-separated series of e-mail addresses in the text fields for each subsystem as desired.

Configuring miscellaneous properties using System Manager

The "Misc" tab in the System Manager configuration file editor has panes for configuring features such as persistent monitoring, flow control, and long-lived business processes. Do the following to configure settings for these features:

1. Click the **Misc** tab.

   Figure 43 shows the "Misc" tab in the System Manager configuration file editor:

Figure 43. Configuring InterChange Server miscellaneous properties

2. Do the following in the "Persistent Monitoring" pane:
   a. Select `Continue` in the **Action on error** drop-down menu if you want InterChange Server to continue running in the event of errors experienced by the persistent monitoring subsystem.

      Select `Shutdown` in the **Action on error** drop-down menu if you want InterChange Server to shut down in response to errors with the subsystem.
   b. Select the desired tracing level in the **Persistent monitoring service** drop-down menu to specify the tracing level for the subsystem.
3. For information on the interface elements in the "Flow Control" pane, see the *System Administration Guide.*
4. For information on the interface elements in the "Flow Monitoring" pane, see the *System Administration Guide.*
5. In the "Workflow Management" pane, type a number in the **Business object consumer processor** to indicate how many threads you want spawned to manage business objects participating in a long-lived business process.

These threads deliver business objects from the collaboration runtime to the appropriate connector controllers when the business objects are retrieved from persistent storage to resume processing in a long-lived business process. For more information on long-lived business processes, see the *Collaboration Development Guide*.

6. For information on the "Server Memory" pane, see "Using the memory checker thread" on page 298.

Configuring environment variables using System Manager

At the "Environment Properties" tab, you can specify any Java user environment properties that might be required in the business integration system. Some environments may require such a variable to be specified; instead of adding them to the script that starts InterChange Server, you can add them to the configuration file by using the "Environment Properties" tab. Do the following to add a new environment property:

1. Click the **Environment Properties** tab.

   Figure 44 shows the "Environment Properties" tab in the System Manager configuration file editor:

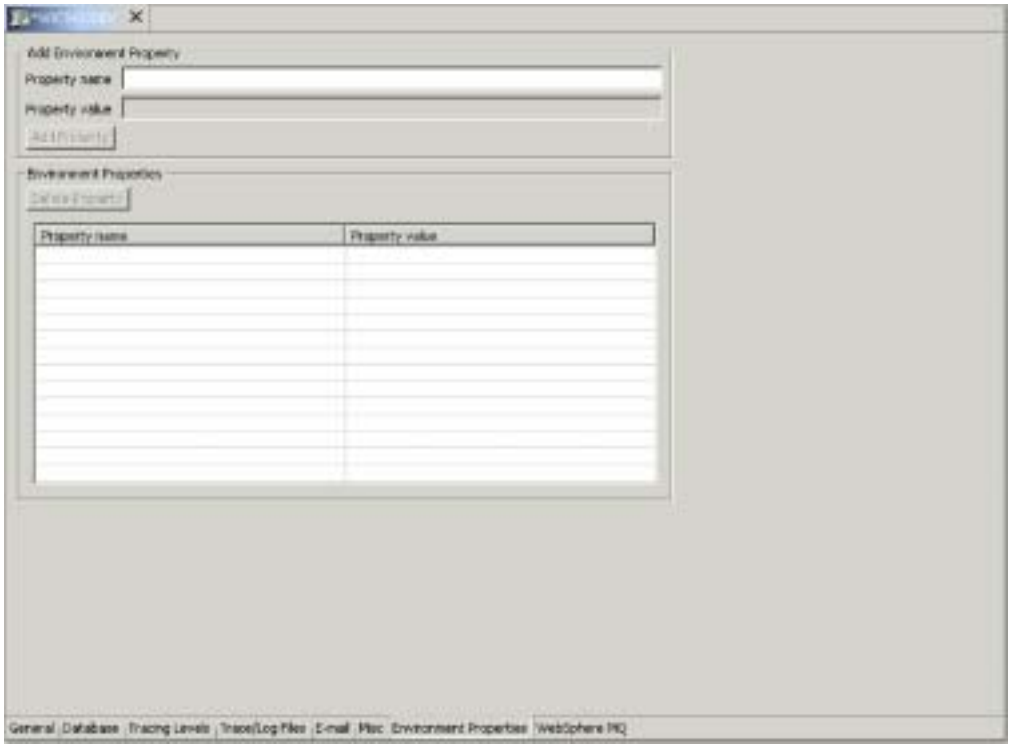

Figure 44. Configuring InterChange Server environment variables

2. Type the name for the property in the **Property name** field.
3. Type the value for the property in the **Property value** field.
4. Click **Add Property**.

To delete a property, select the property and click **Delete Property**.

To edit a property's value, select the property and edit the text in the **Property value** column for the property.

Configuring WebSphere MQ properties using System Manager

Do the following to configure the connectivity between InterChange Server and the WebSphere MQ persistent messaging software:

1. Click the **WebSphere MQ** tab.

   Figure 45 shows the "WebSphere MQ" tab in the System Manager configuration file editor:

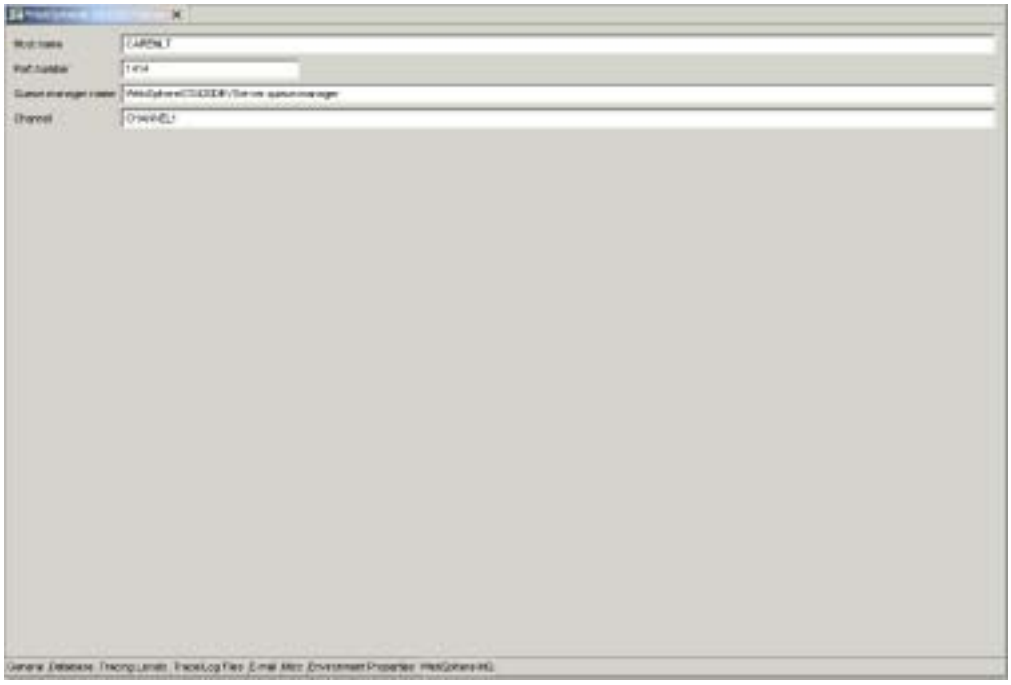

Figure 45. Configuring InterChange Server WebSphere MQ properties

2. Type the name of the computer on which the WebSphere MQ server is installed in the **Host name** field.
3. Type in the **Port number** field the port over which WebSphere MQ allows clients to communicate with the server.

   The default port is 1414.
4. Type in the **Queue manager name** field the name of the queue manager that has been created to contain the queues on which InterChange Server will persistently store messages for the flows it processes.
5. Type in the **Channel** field the name of the channel over which clients of WebSphere MQ communicate with the WebSphere MQ server. The default value is CHANNEL1. Consult with the WebSphere MQ administrator at the site to determine if CHANNEL1 can be used and if not, which channel can be used.

Chapter 6. Using repos_copy

Repos_copy is a command line interface for working with integration components and InterChange Server repositories. It allows you to deploy a package—a collection of integration components—to a server repository, or to export components from the repository to a package.

To run repos_copy, enter the command at a shell prompt (UNIX) or in an MS-DOS command prompt window (Windows). The *ProductDir*/bin directory, where the utility resides, should be in your path as a result of installation.

Note: The repos_copy output file contains encrypted passwords for relationships and connector applications. If you try to edit the output file and change these passwords, repos_copy will not work.

Important: When repos_copy deploys components to the repository, it deploys them to the repository *only*. It does *not* deploy them to any in-memory tables of business object definitions. For instance, connectors load business object definitions from the repository into their memory space when they start. If you deploy a business object definition to the repository to update it, you must restart the connector agent so that it loads the modified business object definition into memory. You must therefore stop and restart InterChange Server and components that load definitions into memory for them to load recently deployed components.

This chapter has the following sections:

For more information about backing up the system, see the *System Administration Guide*.

Repos_copy syntax

Table 10 on page 114 describes the options of repos_copy and their arguments, and shows the correct case usage for the options and the lack of spacing between the option and its argument. The syntax shows that the options between curly braces ({}) represent a set of options that are required. If you do not specify the -u, -p, -i, -o, or -s options at the command line, then repos_copy prompts you for them. If you do not specify them when prompted, repos_copy does not execute. Options enclosed in brackets ([]) are optional.

Note: Some new arguments have been added in release 4.2, and some arguments from the previous release have been removed. For a list of these arguments, see "New arguments in release 4.2" on page 118and "Arguments removed in release 4.2" on page 118.

```
repos_copy [-sserverName][-uusername][-ppassword]
{-i[filename1][-rrelationshipName[relationshipName2]][[-k][-ai|-ar|-arp]
[-xcompilePackage][-vp|-vr]}
{-o[outfilename[[-fEntityFile][-eEntityType:Entity1[+EntityType:Entity2][+...]]
```

```
[-deep][-summary]}
{[-d]|[-doEntityType:Entity[+EntityType:Entity2][+...]|
[-dfoEntityType:Entity[+EntityType:Entity2][+...]}
{-v}
{-vr}
{[-xCompileAll]|[-xCompileAllCollabs]|[-xCompileAllMaps]|
[-xCompileCollab:collabTemplateName[+collabTemplateName][+...]]|
[-xCompileMap:nativeMapName[+nativeMapName][+...]]}
```

Table 10. Repos_copy command options

Option	Description
`-ai`	Ignore and do not load any duplicate objects (business objects, maps, relationships, collaboration templates and objects, and connectors) that are found when deploying a package.
`-ar`	Replace any duplicate objects (business objects, maps, relationships, collaboration templates and objects, and connectors) that are found when deploying a package. **Note:** The `-ar` option only works with release 4.2.0 or later.
`-arp`	This is an interactive version of the `-ar` option. If the components in the package being deployed already exist in the repository then repos_copy displays a prompt asking if you want to ignore or replace the component. **Note:** The `-arp` option only works with release 4.2.0 or later.
`-d`	Deletes the components in the repository, except the state data. Use this option to delete all of the components from the repository.
`-deep`	Used with the `-e` option when you want to include all the dependent components. If you omit the `-deep` option, only the component that is specified with the `-e` option will be included.
`-dfo`*`EntityType:Entity[+EntityType:Entity2]`*	This option is the same as the `-do` option except that it will forcefully delete the component even if the component has referents that depend on it. This option only works with the repository of a server that is running in design mode. A server that is running in production mode does not permit unresolved dependencies and references.
`-do`*`EntityType:Entity[+EntityType:Entity2]`*	Specifies the entities to be deleted from the repository. See Table 11 on page 117for the list of entity types and keywords. If the object has no referents—other components that depend on it—then the deletion takes place. If the object has referents, then the deletion fails and a message is displayed. The behavior is the same in both design mode and production mode. For more information about starting the server in design mode or production mode, see the *System Implementation Guide*.

Table 10. Repos_copy command options (continued)

Option	Description
-e*EntityType:Entity1[+EntityType:Entity2...]*	Exports one or more referenced first-class entities. A first-class entity is a business object, collaboration object, collaboration template, connector, database connection pool, map, or relationship. You identify the entity to load or unload by specifying one of the keywords in Table 11 on page 117. Follow the *EntityType* keyword with a colon (:) and the name of the entity. Use the "+" to specify more than one entity. When combined with the -o option, the -e option unloads the data to an output file.
-f*entityFile*	This option is similar to the -e option except that the names of the entities to be imported are stored in a file. The file should contain references to the entities, with the following conditions: • The entity names must follow after the proper entity type keyword. The entity types and their keywords are listed in Table 11 on page 117. • A colon must separate the entity type from the entity name. • There must be a new line separating each entity reference. When combined with the -o option, this option exports the components to a package.
-i*filename*	Deploys the specified package file to the repository. If you omit the input file name value, the command interactively prompts you to enter the name of the input file. The file can be either a .jar file containing objects in XML format, or a file in text format from a release prior to 4.2.0. The .jar files created by repos_copy or System Manager have a particular structure which must be maintained for any subsequent imports of such a file to be successful. You should not, therefore, ever modify an input file manually.
-k	Overrides the default behavior of repos_copy when it finds a Mercator map in the package file it is loading. By default, repos_copy exits if it encounters a Mercator map. If you use the -k option, repos_copy skips over any Mercator maps in the package file and proceeds with the deployment process.
-mode	Returns the mode of the server. For more information about InterChange Server modes, see the *Implementation Guide for WebSphere InterChange Server*.

Table 10. Repos_copy command options (continued)

Option	Description
-n*encoding*	Specifies the character encoding when importing a text-based repository file fromreleases prior to 4.2.0. For a list of valid character encodings, see the Java documentation about the String class.
-o*outfilename*	Exports the components in the repository to the specified package file. You must specify the name of the package file. If the file already exists then repos_copy prompts you to overwrite it or not. The output file is in `.jar` format, and contains the component definitions in XML format, as well as `.java` source files for components that have them. This option cannot be combined with the `-i` or `-d` options, nor can it export components in text format as it did in previous releases. Repos_copy does not append the `.jar` extension, so you must specify it when specifying the name of the output file.
-p*password*	Specifies the password for the user name supplied with the -u option. The password case-sensitive. If you do not specify this option then repos_copy prompts you for the password.
-r*	This option is similar to the `-r` option; it allows you to import relationship definitions and not create the runtime schemas for any of them.
-r*relationshipName1[:relationshipName2]*	Loads the named relationship definition(s) into the repository without creating its runtime schema.
-s*serverName*	Specifies the name of the InterChange Server instance with which repos_copy should interface. The name is case-sensitive. If the server name is not specified, the tool prompts for a server name.
-summary	This option prints a list of components in the server repository (they are identified as “artifacts” rather than as components in the output). The output is in XML format. You can combine this option with the `-o` option to print the output to a file rather than the console.
-u*username*	Specifies the user name to log in to InterChange Server. If no user name is specified, repos_copy prompts for a user name.
-v	Prints the version number of the program that the repos_copy utility executes.
-vp	This option validates a package file. The server validates packages against the repository and makes sure that the dependencies among the components in the package are resolved. If the validation is not successful, repos_copy prints a list of the missing dependencies. This option does not make any changes to the repository; it just validates the package file. When using the `-vp` option you must also use the `-i` option to specify the package file to be validated.

Table 10. Repos_copy command options (continued)

Option	Description
-vr	This option validates the repository. The output message indicates whether the validation is successful or not. If the validation is not successful, repos_copy prints a list of the missing dependencies.
-wi	When this option is specified, repos_copy does not display any warnings that occur during the compilation of collaboration templates or maps. Only errors that occur during compilation are displayed. This allows the user to ignore warnings about deprecated methods, for instance.
-xCompileAll	Compiles all collaboration templates and maps in the repository. Valid only for collaboration templates and maps created using release 4.2 or later.
-xCompileAllCollabs	Compiles all collaboration templates in the repository. Valid only for templates created using release 4.2 or later.
-xCompileAllMaps	Compiles all maps in the repository. Valid only for maps created using release 4.2 or later.
-xCompileCollab:*collabTemplateName[+collabTemplateName]*	Compiles the specified collaboration templates in the repository. Valid only for templates created using release 4.2 or later.
-xCompileMap:*nativeMapName[+nativeMapName]*	Compiles the specified maps in the repository. Valid only for maps created using release 4.2 or later.
-xCompilePackage	This option automatically compiles the package being deployed to the server. Since the production-mode server automatically compiles all packages, this option applies only to design-mode servers. For a full description of InterChange Server modes, see the System Implementation Guide. **Note:** This option works only if you are deploying components from release 4.2. If the components are from a prior release, this option will be ignored.

Table 11. Keywords for different entity types

Entity type	Keyword
Business object	BusObj
Collaboration object	Collaboration
Collaboration template	CollabTemplate
Database connection pool	ConnectionPool
Connector	Connector
Map	Map
Relationship	Relationship

New arguments in release 4.2

The following list contains all of the new options provided with repos_copy in release 4.2:

- -dfoEntityType:Entity[+EntityType:Entity2]
- -doEntityType:Entity[+EntityType:Entity2]
- -mode
- -r*
- -summary
- -wi
- -xCompilePackage

Arguments removed in release 4.2

The following table lists the repos copy arguments that were removed from the

Table 12. Arguments removed from repos copy

Arguments removed	Reason for removal
[-xCompileUpdated] [-xCompileUpdatedCollabs] [-xCompileUpdatedMaps]	All compile update options have been removed, because the server does not support maps or templates prior to release 4.x.
[-xUncompress]	In the new package format, all definitions are stored in the Java Archive (JAR) format, instead of with a proprietary compression algorithm. The -xUncompress argument is therefore no longer necessary.
[-eProject][-w]	Project type is no longer supported for the -e option. Projects are now maintained in the local file by System Manager, instead of being maintained by the server.

Repos_copy usage scenarios

This section describes many of the common situations in which you will use repos_copy. It contains the following sections:

Printing the repos_copy command

You can run repos_copy without any arguments to have the command and its arguments printed out. The example below shows repos_copy when executed without any arguments, and the resulting output:

```
C:\>repos_copy
No Command line arguments to ReposCopy were specified
Usage: repos_copy {-o[outputFile] | -i[inputFile]}
    [-sserverName] [-uuserName] [-ppassword]
    [-ai] [-ar] [-arp] [-d] [-k] [-v]
    [-eentityType:entityName1[+entityType:entityName2] -deep]
```

```
[-fentityFileName]
[-rrelationshipName1[:relationshipName2] ]
[-xCompileAll] [-xCompileAllCollabs] [-xCompileAllMaps]
[-xCompileCollab:collabTemplateName[+collabTemplateName]]
[-xCompileMap:nativeMapName[+nativeMapName]]
[-xcompilepackage]
[-mode]
[-doentityType:entityName1[+entityType:entityName2] -deep]
[-dfoentityType:entityName1[+entityType:entityName2] -deep]
[-summary]
[-vp]
[-vr]
```

Validating a package

You can validate a package of components before deploying the package to a server. This is very useful because if you deploy a package to a production-mode server all the dependencies must be resolved or the deployment will fail. You cannot validate a user project or integration component library in System Manager to make sure that the dependencies are satisfied, so the only way to find out if a package is valid when deploying with System Manager is to attempt the deployment and use the error information when it fails to resolve the dependencies. If there are many components in the package, this can be a very time-consuming process.

Although you cannot validate an integration component library, you can export it to a package file and then validate the package file using repos_copy.

To validate a package file using repos_copy, use the `-i` option to specify the name of the package file to be validated and the `-vp` argument to validate it rather than deploy it.

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-iWebSphereICS420DEVServer.jar -vp
```

Repos_copy validates the contents of the package and displays a message to indicate whether or not the dependencies are resolved.

Deploying a package to the repository

The `-i` option allows you to deploy a package of components to the repository. If you do not specify the name of the package file then you are prompted to enter it.

The following example shows a a file named `WebSphereICS420DEVServer.jar` being deployed to a repository:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-iWebSphereICS420DEVServer.jar
```

Working with duplicate components during deployment

Commonly there will be components with the same name in the package file as there are in the repository. In this case you must decide whether or not you want to replace the components in the repository with those in the package file. The `-ai` option specifies that duplicate components should not be loaded into the repository:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-iCustomer.jar -ai
```

If you want to replace all the duplicate components in the repository, use the `-ar` option as in the following example:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-iCustomerSyncInterface.jar -ar
```

You can use the -arp option to interactively replace duplicate components in the repository. This lets you decide for each individual duplicate component whether it should be replaced or not.

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-iCustomerSyncInterface.jar -arp
```

Note: The -ar and -arp options only work with release 4.2.0 or later.

Compiling and creating schemas

For maps and collaboration to execute at runtime, the maps and collaboration templates defined in the repository must be compiled. For relationships to function properly at runtime, their schemas must be created.

When you deploy components to a server running in production mode, all templates are automatically compiled and all relationship schemas are created. For the deployment to succeed, then, the code of the map and collaboration templates must be valid and InterChange Server must be able to communicate with the databases specified in the settings of the relationship definitions.

When you deploy components to a server running in design mode, the templates are not automatically compiled; relationship schemas are automatically created. There are options you can use to compile the templates, however, and there are options to not create relationship schemas.

The following example uses the -xCompilePackage option and does not use any form of the -r option. The result is that when the package specified by the -i option is deployed, the maps and collaboration templates are compiled and schemas are created for the relationships:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-iWebSphereICS420DEVServer.jar -xCompilePackage
```

You may not want relationship schemas created when you do a deployment. For instance, if you are deploying a package from one environment to another and did not change the properties of the relationships to use the database resources in the new environment then you will not want the schemas created until after you have changed the relevant properties. The following example uses the -r* option to not create schemas for all of the relationships in the package being deployed:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-iWebSphereICS420DEVServer.jar -xCompilePackage -r*
```

Note: You can use the -r option without the asterisk to specify the names of individual relationships whose schemas should not be created. For instance, -rCustomer:Order would not create schemas for the Customer and Order relationships, but would still create schemas for any other relationships in the package being deployed.

Important: Although there are options to compile maps and collaboration templates after deployment, there is no way to either through repos_copy or System Manager to create the schema for a relationship other than during deployment. So, if you chose not to create the schema for a relationship during deployment because you needed to

change the database settings, then you need to re-deploy the relationship afterwards and allow repos_copy to create the schema for the relationship.

Validating the repository

The repository must be in a valid state for a server instance to start in production mode. The reason for this is that ultimately the repository must be valid for the server to process flows successfully. Use the -vr option to validate a server repository, as in the example below:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull -vr
```

If the server is valid then repos_copy writes the following output to the console:

```
Validation Succeeded.All Dependencies Resolved.
```

If the repository is not valid then repos_copy prints a list of the dependencies that must be resolved.

Compiling components in the repository

If you deployed maps or collaboration templates to the repository and did not compile them during deployment, you can use repos_copy to compile them afterwards. This can be useful in situations where there are many components to deploy because deployment can take a long time and compiling can make the operation take even longer. Waiting until after the deployment has succeeded to do the compilation task can reduce the risk of spending an even greater amount of time migrating the environment if an error occurs.

The following example shows the use of the -xCompileAll option to compile all maps and collaboration templates in the the repository:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-xCompileAll
```

There are options to compile all of either type of component as well. Use -xCompileAllCollabs to compile all the collaboration templates, and -xCompileAllMaps to compile all the maps. The example below shows the use of -xCompileAllMaps:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-xCompileAllMaps
```

Just as you can compile all of one type of component, you can also compile an individual component. Use the -xCompileCollab or -xCompileMap option followed by a colon and the name of the collaboration template or map to compile a single component. The example below would compile a collaboration template named CustomerSync:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-xCompileCollab:CustomerSync
```

Note: The compilation options only work with WebSphere InterChange Server version 4.2.0 and later.

Deleting components from the repository

There are several options provided by repos_copy for deleting components in the repository. You can delete the entire repository, individual components, and individual components as well as any components that reference them.

Note: Components must be inactive for you to delete them. If you delete a single component then you must deactivate it first or the delete operation will fail. If you want to delete a component and all the components that reference it, you must deactivate not only the single component, but all those that reference it as well. You can delete the entire repository while the components are in an active state. Use System Monitor or web-based System Monitor to manage the states of components. System Monitor and web-based System Monitor are described in the *System Administration Guide*.

Deleting the entire repository

Use the `-d` option to delete all of the components in the repository. The following example shows the syntax:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin
-pnull -d
```

Repos_copy presents a prompt asking if you want to delete the entire repository or not.

Deleting components without referents

If a component does not have any referents—other components that reference it and require it to exist in order to perform their function in the system—then you can delete the individual component.

Use the `-do` option followed by the entity type, a colon, and the name of the component. The entity types are listed in Table 11 on page 117.The following example deletes the relationship named **Customer**:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin
-pnull -doRelationship:Customer
```

Deleting components with referents

If a component does have referents—other components that reference it and require it to exist in order to perform their function in the system—then you can only delete the component if the server is running in design-mode, and by using certain options.

Forcing a delete in spite of references: If a component has referents, repos_copy will not let you delete it with the `-do` option. You must use the `-dfo` option to force deletion of a component with referents. Forcing deletion of a component that has referents will leave the repository in an inconsistent state, and a server running in production mode does not permit that, so this option only works with a design-mode server. The following example shows the use of the `-dfo` option to delete the `Order` business object in spite of the fact that other components in the system (such as maps and relationships) have references to it:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-dfoBusObj:Order
```

Deleting the referents as well: Another way you can delete a component that has referents is to use the `-deep` option to delete the referents as well. This deletes the component and all of the components that have references to it. The following example shows the use of the `-deep` option when using the `-do` option to delete the `Customer` business object:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-doBusObj:Customer -deep
```

This option, unlike the `-dfo` option, is supported with servers running in production mode because the deletion of the referents along with the component

guarantees that the repository remains valid. Keep in mind, however, that it can result in many components being deleted; you should be aware of the implications of this action prior to taking it.

Exporting components to a package

The -o option allows you to export components from the repository to a package. You must specify the name of the package file. When the -o option is used alone the entire repository is exported to a file, as in the following example:

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-oWebSphereICS420DEVServer.jar
```

You can specify individual components to be exported by using the -e option. You must use the -e option with the appropriate EntityType keyword listed in Table 11 on page 117,and must follow the keyword with the name of the component. You can specify multiple components by concatenating them with the plus (+) sign. In the following example, the Customer business object and CustomerSync collaboration template are exported to a package named CustomerSyncInterface.jar.

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-eBusObj:Customer+CollabTemplate:CustomerSync -oCustomerSyncInterface.jar
```

You can use the -deep option to export the dependencies of a component as well. In the previous example, the Customer business object was exported, but none of its child business objects were. The following example uses the -deep option to export the CustomerSync_ClarifyToSAP collaboration object and all of its dependencies.

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-eCollaboration:CustomerSync_ClarifyToSAP -oCustomerSyncInterface.jar -deep
```

If you want to export specific components, but do not want to have to enter the entity type keyword and component names, you can store them in a text file and use the -f option. This is very convenient when you want to frequently export the same components. The following example uses the -f option to load the components listed in a text file named Components.txt :

```
C:\WebSphereICS420DEV>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull
-fComponents.txt -oCustomerSyncInterface.jar -deep
```

The contents of the file Components.txt are shown below; a paragraph return follows each entity type keyword and name combination:

```
BusObj:Customer
Relationship:Customer
CollabTemplate:CustomerSync
```

Note: Repos_copy and System Manager are unfortunately inconsistent with respect to what they identify as “dependencies”. If you attempt to delete a component using repos_copy but there are components that depend upon it then repos_copy lists those referring components as dependencies. However, if you right-click the component in System Manager and select **Show Dependencies** from the context menu the tool lists the components that the selected component depends on.

Printing a list of components in the repository

You can use the -summary argument when executing repos_copy to print a list of the components in the repository. The output is presented in XML format. Although it is not particularly useful to view at the command line, you can

combine the -summary argument with the the -o argument to redirect the output to a file and then view the file in a browser or XML editor. The command usage in this case would be the following:

```
C:\>repos_copy -sWebSphereICS420DEVServer -uadmin -pnull -summary -oRepository.xml
```

Locale for repos_copy files

The repos_copy utility reads metadata from the repository and writes the data out to files in Unicode (UTF-8 format). It also reads such files and loads them into the repository in Unicode (UTF-8 or UCS-2, as the underlying repository database dictates).

Repos_copy files created with IBM WebSphere ICS version levels earlier than 4.1.1 can be loaded ino the repository correctly only if the dates and times for the component schedules are in full US format. (This is usually not an issue. Repos_copy saves all schedule dates in full US format only. The incompatibility could typically arise if the repos_copy files have been manually edited.)

Chapter 7. Configuring connectors

This chapter describes the procedures for configuring WebSphere Business Integration Adapters and contains the following sections:

- "Using Connector Configurator"
- "Connector standard properties" on page 139

For information about developing WebSphere Business Integration Adapters, see the *Connector Development Guide*.

Using Connector Configurator

This appendix describes how to use Connector Configurator to set configuration property values for your adapter.

You use Connector Configurator to:

- Create a connector-specific property template for configuring your connector
- Create a configuration file
- Set properties in a configuration file

Note:

In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes and follow the conventions for each operating system.

The topics covered in this appendix are:

- "Overview of Connector Configurator" on page 125
- "Starting Connector Configurator" on page 126
- "Creating a connector-specific property template" on page 127
- "Creating a new configuration file" on page 129
- "Setting the configuration file properties" on page 132
- "Using Connector Configurator in a globalized environment" on page 138

Overview of Connector Configurator

Connector Configurator allows you to configure the connector component of your adapter for use with these integration brokers:

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator, WebSphere MQ Integrator Broker, and WebSphere Business Integration Message Broker, collectively referred to as the WebSphere Message Brokers (WMQI)
- WebSphere Application Server (WAS)

You use Connector Configurator to:

- Create a **connector-specific property template** for configuring your connector.
- Create a **connector configuration file**; you must create one configuration file for each connector you install.
- Set properties in a configuration file.
  You may need to modify the default values that are set for properties in the

connector templates. You must also designate supported business object definitions and, with ICS, maps for use with collaborations as well as specify messaging, logging and tracing, and data handler parameters, as required.

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 126).

Connector configuration properties include both standard configuration properties (the properties that all connectors have) and connector-specific properties (properties that are needed by the connector for a specific application or technology).

Because **standard properties** are used by all connectors, you do not need to define those properties from scratch; Connector Configurator incorporates them into your configuration file as soon as you create the file. However, you do need to set the value of each standard property in Connector Configurator.

The range of standard properties may not be the same for all brokers and all configurations. Some properties are available only if other properties are given a specific value. The Standard Properties window in Connector Configurator will show the properties available for your particular configuration.

For **connector-specific properties**, however, you need first to define the properties and then set their values. You do this by creating a connector-specific property template for your particular adapter. There may already be a template set up in your system, in which case, you simply use that. If not, follow the steps in "Creating a new template" on page 127 to set up a new one.

Note: Connector Configurator runs only in a Windows environment. If you are running the connector in a UNIX environment, use Connector Configurator in Windows to modify the configuration file and then copy the file to your UNIX environment.

Starting Connector Configurator

You can start and run Connector Configurator in either of two modes:

- Independently, in stand-alone mode
- From System Manager

Running Configurator in stand-alone mode

You can run Connector Configurator independently and work with connector configuration files, irrespective of your broker.

To do so:

- From **Start>Programs**, click **IBM WebSphere InterChange Server>IBM WebSphere Business Integration Toolset>Development>Connector Configurator**.
- Select **File>New>Configuration File**.
- When you click the pull-down menu next to **System Connectivity Integration Broker**, you can select ICS, WebSphere Message Brokers or WAS, depending on your broker.

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see “Completing a configuration file” on page 131.)

Running Configurator from System Manager

You can run Connector Configurator from System Manager.

To run Connector Configurator:

1. Open the System Manager.
2. In the System Manager window, expand the **Integration Component Libraries** icon and highlight **Connectors**.
3. From the System Manager menu bar, click **Tools>Connector Configurator**. The Connector Configurator window opens and displays a **New Connector** dialog box.
4. When you click the pull-down menu next to **System Connectivity Integration Broker**, you can select ICS, WebSphere Message Brokers or WAS, depending on your broker.

To edit an existing configuration file:

1. In the System Manager window, select any of the configuration files listed in the Connector folder and right-click on it. Connector Configurator opens and displays the configuration file with the integration broker type and file name at the top.
2. Click the Standard Properties tab to see which properties are included in this configuration file.

Creating a connector-specific property template

To create a configuration file for your connector, you need a connector-specific property template as well as the system-supplied standard properties.

You can create a brand-new template for the connector-specific properties of your connector, or you can use an existing file as the template.

- To create a new template, see “Creating a new template” on page 127.
- To use an existing file, simply modify an existing template and save it under the new name.

Creating a new template

This section describes how you create properties in the template, define general characteristics and values for those properties, and specify any dependencies between the properties. Then you save the template and use it as the base for creating a new connector configuration file.

To create a template:

1. Click **File>New>Connector-Specific Property Template**.
2. The **Connector-Specific Property Template** dialog box appears, with the following fields:
   - **Template**, and **Name**

     Enter a unique name that identifies the connector, or type of connector, for which this template will be used. You will see this name again when you open the dialog box for creating a new configuration file from a template.
   - **Old Template**, and **Select the Existing Template to Modify**

The names of all currently available templates are displayed in the Template Name display.

- To see the connector-specific property definitions in any template, select that template's name in the **Template Name** display. A list of the property definitions contained in that template will appear in the **Template Preview** display. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template.

3. Select a template from the **Template Name** display, enter that template name in the **Find Name** field (or highlight your selection in **Template Name**), and click **Next**.

If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one.

Specifying general characteristics: When you click **Next** to select a template, the **Properties - Connector-Specific Property Template** dialog box appears. The dialog box has tabs for General characteristics of the defined properties and for Value restrictions. The General display has the following fields:

- **General:**
  Property Type
  Updated Method
  Description
- **Flags**
  Standard flags
- **Custom Flag**
  Flag

After you have made selections for the general characteristics of the property, click the **Value** tab.

Specifying values: The **Value** tab enables you to set the maximum length, the maximum multiple values, a default value, or a value range for the property. It also allows editable values. To do so:

1. Click the **Value** tab. The display panel for Value replaces the display panel for General.
2. Select the name of the property in the **Edit properties** display.
3. In the fields for **Max Length** and **Max Multiple Values**, make any changes. The changes will not be accepted unless you also open the **Property Value** dialog box for the property, described in the next step.
4. Right-click the box in the top left-hand corner of the value table and click **Add**. A **Property Value** dialog box appears. Depending on the property type, the dialog box allows you to enter either a value, or both a value and range. Enter the appropriate value or range, and click **OK**.
5. The **Value** panel refreshes to display any changes you made in **Max Length** and **Max Multiple Values**. It displays a table with three columns:

   The **Value** column shows the value that you entered in the **Property Value** dialog box, and any previous values that you created.

   The **Default Value** column allows you to designate any of the values as the default.

   The **Value Range** shows the range that you entered in the **Property Value** dialog box.

After a value has been created and appears in the grid, it can be edited from within the table display. To make a change in an existing value in the table, select an entire row by clicking on the row number. Then right-click in the **Value** field and click **Edit Value**.

Setting dependencies: When you have made your changes to the **General** and **Value** tabs, click **Next**. The **Dependences - Connector-Specific Property Template** dialog box appears.

A dependent property is a property that is included in the template and used in the configuration file *only if* the value of another property meets a specific condition. For example, `PollQuantity` appears in the template only if JMS is the transport mechanism and `DuplicateEventElimination` is set to `True`.
To designate a property as dependent and to set the condition upon which it depends, do this:

1. In the **Available Properties** display, select the property that will be made dependent.
2. In the **Select Property** field, use the drop-down menu to select the property that will hold the conditional value.
3. In the **Condition Operator** field, select one of the following:

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

   <=(less than or equal to)
4. In the **Conditional Value** field, enter the value that is required in order for the dependent property to be included in the template.
5. With the dependent property highlighted in the **Available Properties** display, click an arrow to move it to the **Dependent Property** display.
6. Click **Finish**. Connector Configurator stores the information you have entered as an XML document, under `\data\app` in the`\bin` directory where you have installed Connector Configurator.

Creating a new configuration file

When you create a new configuration file, your first step is to select an integration broker. The broker you select determines the properties that will appear in the configuration file.

To select a broker:

- In the Connector Configurator home menu, click **File>New>Connector Configuration**. The **New Connector** dialog box appears.
- In the I**ntegration Broker** field, select ICS, WebSphere Message Brokers or WAS connectivity.
- Complete the remaining fields in the **New Connector** window, as described later in this chapter.

You can also do this:

- In the System Manager window, right-click on the **Connectors** folder and select **Create New Connector**. Connector Configurator opens and displays the **New Connector** dialog box.

Creating a configuration file from a connector-specific template

Once a connector-specific template has been created, you can use it to create a configuration file:

1. Click **File>New>Connector Configuration**.
2. The **New Connector** dialog box appears, with the following fields:
   - **Name**

     Enter the name of the connector. Names are case-sensitive. The name you enter must be unique, and must be consistent with the file name for a connector that is installed on the system.

     Important: Connector Configurator does not check the spelling of the name that you enter. You must ensure that the name is correct.
   - **System Connectivity**

     Click ICS or WebSphere Message Brokers or WAS.
   - **Select Connector-Specific Property Template**

     Type the name of the template that has been designed for your connector. The available templates are shown in the **Template Name** display. When you select a name in the Template Name display, the **Property Template Preview** display shows the connector-specific properties that have been defined in that template.

     Select the template you want to use and click **OK**.
3. A configuration screen appears for the connector that you are configuring. The title bar shows the integration broker and connector names. You can fill in all the field values to complete the definition now, or you can save the file and complete the fields later.
4. To save the file, click **File>Save>To File** or **File>Save>To Project**. To save to a project, System Manager must be running.
   If you save as a file, the **Save File Connector** dialog box appears. Choose `*.cfg` as the file type, verify in the File Name field that the name is spelled correctly and has the correct case, navigate to the directory where you want to locate the file, and click **Save**. The status display in the message panel of Connector Configurator indicates that the configuration file was successfully created.

   Important: The directory path and name that you establish here must match the connector configuration file path and name that you supply in the startup file for the connector.
5. To complete the connector definition, enter values in the fields for each of the tabs of the Connector Configurator window, as described later in this chapter.

Using an existing file

You may have an existing file available in one or more of the following formats:

- A connector definition file.
  This is a text file that lists properties and applicable default values for a specific connector. Some connectors include such a file in a `\repository` directory in their delivery package (the file typically has the extension `.txt`; for example, CN_XML.txt for the XML connector).
- An ICS repository file.
  Definitions used in a previous ICS implementation of the connector may be available to you in a repository file that was used in the configuration of that connector. Such a file typically has the extension `.in` or `.out`.
- A previous configuration file for the connector.
  Such a file typically has the extension *.cfg.

Although any of these file sources may contain most or all of the connector-specific properties for your connector, the connector configuration file will not be complete until you have opened the file and set properties, as described later in this chapter.

To use an existing file to configure a connector, you must open the file in Connector Configurator, revise the configuration, and then resave the file.

Follow these steps to open a *.txt, *.cfg, or *.in file from a directory:

1. In Connector Configurator, click **File>Open>From File**.
2. In the **Open File Connector** dialog box, select one of the following file types to see the available files:
   - Configuration (`*.cfg`)
   - ICS Repository (`*.in`, `*.out`)

     Choose this option if a repository file was used to configure the connector in an ICS environment. A repository file may include multiple connector definitions, all of which will appear when you open the file.
   - All files (*.*)

     Choose this option if a `*.txt` file was delivered in the adapter package for the connector, or if a definition file is available under another extension.
3. In the directory display, navigate to the appropriate connector definition file, select it, and click **Open**.

Follow these steps to open a connector configuration from a System Manager project:

1. Start System Manager. A configuration can be opened from or saved to System Manager only if System Manager has been started.
2. Start Connector Configurator.
3. Click **File>Open>From Project**.

Completing a configuration file

When you open a configuration file or a connector from a project, the Connector Configurator window displays the configuration screen, with the current attributes and values.

The title of the configuration screen displays the integration broker and connector name as specified in the file. Make sure you have the correct broker. If not, change the broker value before you configure the connector. To do so:

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 134..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

   If you are saving to file, select `*.cfg` as the extension, select the correct location for the file and click **Save**.

   If multiple connector configurations are open, click **Save All to File** to save all of the configurations to file, or click **Save All to Project** to save all connector configurations to a System Manager project.

Before it saves the file, Connector Configurator checks that values have been set for all required standard properties. If a required standard property is missing a value, Connector Configurator displays a message that the validation failed. You must supply a value for the property in order to save the configuration file.

Setting the configuration file properties

When you create and name a new connector configuration file, or when you open an existing connector configuration file, Connector Configurator displays a configuration screen with tabs for the categories of required configuration values.

Connector Configurator requires values for properties in these categories for connectors running on all brokers:

- Standard Properties
- Connector-specific Properties
- Supported Business Objects
- Trace/Log File values
- Data Handler (applicable for connectors that use JMS messaging with guaranteed event delivery)

Note: For connectors that use JMS messaging, an additional category may display, for configuration of data handlers that convert the data to business objects.

For connectors running on **ICS**, values for these properties are also required:

- Associated Maps
- Resources
- Messaging (where applicable)

Important: Connector Configurator accepts property values in either English or non-English character sets. However, the names of both standard and connector-specific properties, and the names of supported business objects, must use the English character set only.

Standard properties differ from connector-specific properties as follows:

- Standard properties of a connector are shared by both the application-specific component of a connector and its broker component. All connectors have the same set of standard properties. These properties are described in Appendix A of each adapter guide. You can change some but not all of these values.
- Application-specific properties apply only to the application-specific component of a connector, that is, the component that interacts directly with the application. Each connector has application-specific properties that are unique to its application. Some of these properties provide default values and some do not; you can modify some of the default values. The installation and configuration chapters of each adapter guide describe the application-specific properties and the recommended values.

The fields for **Standard Properties** and **Connector-Specific Properties** are color-coded to show which are configurable:

- A field with a grey background indicates a standard property. You can change the value but cannot change the name or remove the property.

- A field with a white background indicates an application-specific property. These properties vary according to the specific needs of the application or connector. You can change the value and delete these properties.
- Value fields are configurable.
- The **Update Method** field is informational and not configurable. This field specifies the action required to activate a property whose value has changed.

Setting standard connector properties

To change the value of a standard property:

1. Click in the field whose value you want to set.
2. Either enter a value, or select one from the drop-down menu if it appears.
3. After entering all the values for the standard properties, you can do one of the following:
   - To discard the changes, preserve the original values, and exit Connector Configurator, click **File>Exit** (or close the window), and click **No** when prompted to save changes.
   - To enter values for other categories in Connector Configurator, select the tab for the category. The values you enter for **Standard Properties** (or any other category) are retained when you move to the next category. When you close the window, you are prompted to either save or discard the values that you entered in all the categories as a whole.
   - To save the revised values, click **File>Exit** (or close the window) and click **Yes** when prompted to save changes. Alternatively, click **Save>To File** from either the File menu or the toolbar.

Setting application-specific configuration properties

For application-specific configuration properties, you can add or change property names, configure values, delete a property, and encrypt a property. The default property length is 255 characters.

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property. To add a child property, right-click on the parent row number and click **Add child**.
2. Enter a value for the property or child property.
3. To encrypt a property, select the **Encrypt** box.
4. Choose to save or discard changes, as described for "Setting standard connector properties."

The Update Method displayed for each property indicates whether a component or agent restart is necessary to activate changed values.

Important: Changing a preset application-specific connector property name may cause a connector to fail. Certain property names may be needed by the connector to connect to an application or to run properly.

Encryption for connector properties: Application-specific properties can be encrypted by selecting the **Encrypt** check box in the **Edit Property** window. To decrypt a value, click to clear the **Encrypt** check box, enter the correct value in the **Verification** dialog box, and click **OK**. If the entered value is correct, the value is decrypted and displays.

The adapter user guide for each connector contains a list and description of each property and its default value.

If a property has multiple values, the **Encrypt** check box will appear for the first value of the property. When you select **Encrypt**, all values of the property will be encrypted. To decrypt multiple values of a property, click to clear the **Encrypt** check box for the first value of the property, and then enter the new value in the **Verification** dialog box. If the input value is a match, all multiple values will decrypt.

Update method: Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under "Setting and updating property values" on page 140.

Specifying supported business object definitions

Use the **Supported Business Objects** tab in Connector Configurator to specify the business objects that the connector will use. You must specify both generic business objects and application-specific business objects, and you must specify associations for the maps between the business objects.

Note: Some connectors require that certain business objects be specified as supported in order to perform event notification or additional configuration (using meta-objects) with their applications. For more information, see the *Connector Development Guide for C++* or the *Connector Development Guide for Java*.

If ICS is your broker: To specify that a business object definition is supported by the connector, or to change the support settings for an existing business object definition, click the **Supported Business Objects** tab and use the following fields.

Business object name: To designate that a business object definition is supported by the connector, with System Manager running:

1. Click an empty field in the **Business Object Name** list. A drop-down list displays, showing all the business object definitions that exist in the System Manager project.
2. Click on a business object to add it.
3. Set the **Agent Support** (described below) for the business object.
4. In the File menu of the Connector Configurator window, click **Save to Project**. The revised connector definition, including designated support for the added business object definition, is saved to the project in System Manager.

To delete a business object from the supported list:

1. To select a business object field, click the number to the left of the business object.
2. From the **Edit** menu of the Connector Configurator window, click **Delete Row**. The business object is removed from the list display.
3. From the **File** menu, click **Save to Project**.

Deleting a business object from the supported list changes the connector definition and makes the deleted business object unavailable for use in this implementation of this connector. It does not affect the connector code, nor does it remove the business object definition itself from System Manager.

Agent support: If a business object has Agent Support, the system will attempt to use that business object for delivering data to an application via the connector agent.

Typically, application-specific business objects for a connector are supported by that connector's agent, but generic business objects are not.

To indicate that the business object is supported by the connector agent, check the **Agent Support** box. The Connector Configurator window does not validate your Agent Support selections.

Maximum transaction level: The maximum transaction level for a connector is the highest transaction level that the connector supports.

For most connectors, `Best Effort` is the only possible choice.

You must restart the server for changes in transaction level to take effect.

If a WebSphere Message Broker is your broker: If you are working in stand-alone mode (not connected to System Manager), you must enter the business name manually.

If you have System Manager running, you can select the empty box under the **Business Object Name** column in the **Supported Business Objects** tab. A combo box appears with a list of the business object available from the Integration Component Library project to which the connector belongs. Select the business object you want from the list.

The **Message Set ID** is an optional field for WebSphere Business Integration Message Broker 5.0, and need not be unique if supplied. However, for WebSphere MQ Integrator and Integrator Broker 2.1, you must supply a unique **ID**.

If WAS is your broker: When WebSphere Application Server is selected as your broker type, Connector Configurator does not require message set IDs. The **Supported Business Objects** tab shows a **Business Object Name** column only for supported business objects.

If you are working in stand-alone mode (not connected to System Manager), you must enter the business object name manually.

If you have System Manager running, you can select the empty box under the Business Object Name column in the Supported Business Objects tab. A combo box appears with a list of the business objects available from the Integration Component Library project to which the connector belongs. Select the business object you want from this list.

Associated maps (ICS only)

Each connector supports a list of business object definitions and their associated maps that are currently active in WebSphere InterChange Server. This list appears when you select the **Associated Maps** tab.

The list of business objects contains the application-specific business object which the agent supports and the corresponding generic object that the controller sends to the subscribing collaboration. The association of a map determines which map will be used to transform the application-specific business object to the generic business object or the generic business object to the application-specific business object.

If you are using maps that are uniquely defined for specific source and destination business objects, the maps will already be associated with their appropriate business objects when you open the display, and you will not need (or be able) to change them.

If more than one map is available for use by a supported business object, you will need to explicitly bind the business object with the map that it should use.

The **Associated Maps** tab displays the following fields:

- **Business Object Name**

  These are the business objects supported by this connector, as designated in the **Supported Business Objects** tab. If you designate additional business objects under the Supported Business Objects tab, they will be reflected in this list after you save the changes by choosing **Save to Project** from the **File** menu of the Connector Configurator window.
- **Associated Maps**

  The display shows all the maps that have been installed to the system for use with the supported business objects of the connector. The source business object for each map is shown to the left of the map name, in the **Business Object Name** display.
- **Explicit**

  In some cases, you may need to explicitly bind an associated map.

  Explicit binding is required only when more than one map exists for a particular supported business object. When ICS boots, it tries to automatically bind a map to each supported business object for each connector. If more than one map takes as its input the same business object, the server attempts to locate and bind one map that is the superset of the others.

  If there is no map that is the superset of the others, the server will not be able to bind the business object to a single map, and you will need to set the binding explicitly.

  To explicitly bind a map:

  1. In the **Explicit** column, place a check in the check box for the map you want to bind.
  2. Select the map that you intend to associate with the business object.
  3. In the **File** menu of the Connector Configurator window, click **Save to Project**.
  4. Deploy the project to ICS.
  5. Reboot the server for the changes to take effect.

Resources (ICS)

The **Resource** tab allows you to set a value that determines whether and to what extent the connector agent will handle multiple processes concurrently, using connector agent parallelism.
Not all connectors support this feature. If you are running a connector agent that was designed in Java to be multi-threaded, you are advised not to use this feature, since it is usually more efficient to use multiple threads than multiple processes.

Messaging (ICS)

The messaging properties are available only if you have set MQ as the value of the `DeliveryTransport` standard property and ICS as the broker type. These properties affect how your connector will use queues.

Setting trace/log file values

When you open a connector configuration file or a connector definition file, Connector Configurator uses the logging and tracing values of that file as default values. You can change those values in Connector Configurator.

To change the logging and tracing values:

1. Click the **Trace/Log Files** tab.
2. For either logging or tracing, you can choose to write messages to one or both of the following:
   - To console (STDOUT):
     Writes logging or tracing messages to the STDOUT display.

     Note: You can only use the STDOUT option from the **Trace/Log Files** tab for connectors running on the Windows platform.
   - To File:
     Writes logging or tracing messages to a file that you specify. To specify the file, click the directory button (ellipsis), navigate to the preferred location, provide a file name, and click **Save**. Logging or tracing message are written to the file and location that you specify.

     Note: Both logging and tracing files are simple text files. You can use the file extension that you prefer when you set their file names. For tracing files, however, it is advisable to use the extension `.trace` rather than `.trc`, to avoid confusion with other files that might reside on the system. For logging files, `.log` and `.txt` are typical file extensions.

Data handlers

The data handlers section is available for configuration only if you have designated a value of JMS for DeliveryTransport and a value of JMS for ContainerManagedEvents. Not all adapters make use of data handlers.

See the descriptions under `ContainerManagedEvents` in Appendix A, Standard Properties, for values to use for these properties. For additional details, see the *Connector Development Guide for C++* or the *Connector Development Guide for Java.*

Saving your configuration file

When you have finished configuring your connector, save the connector configuration file. Connector Configurator saves the file in the broker mode that you selected during configuration. The title bar of Connector Configurator always displays the broker mode (ICS, WMQI or WAS) that it is currently using.

The file is saved as an XML document. You can save the XML document in three ways:

- From System Manager, as a file with a `*.con` extension in an Integration Component Library, or
- In a directory that you specify.
- In stand-alone mode, as a file with a `*.cfg` extension in a directory folder.

For details about using projects in System Manager, and for further information about deployment, see the following implementation guides:

- For ICS: *Implementation Guide for WebSphere InterChange Server*
- For WebSphere Message Brokers: *Implementing Adapters with WebSphere Message Brokers*

- For WAS: *Implementing Adapters with WebSphere Application Server*

Changing a configuration file

You can change the integration broker setting for an existing configuration file. This enables you to use the file as a template for creating a new configuration file, which can be used with a different broker.

Note: You will need to change other configuration properties as well as the broker mode property if you switch integration brokers.

To change your broker selection within an existing configuration file (optional):

- Open the existing configuration file in Connector Configurator.
- Select the **Standard Properties** tab.
- In the **BrokerType** field of the Standard Properties tab, select the value that is appropriate for your broker.
  When you change the current value, the available tabs and field selections on the properties screen will immediately change, to show only those tabs and fields that pertain to the new broker you have selected.

Completing the configuration

After you have created a configuration file for a connector and modified it, make sure that the connector can locate the configuration file when the connector starts up.

To do so, open the startup file used for the connector, and verify that the location and file name used for the connector configuration file match exactly the name you have given the file and the directory or path where you have placed it.

Using Connector Configurator in a globalized environment

Connector Configurator is globalized and can handle character conversion between the configuration file and the integration broker. Connector Configurator uses native encoding. When it writes to the configuration file, it uses UTF-8 encoding.

Connector Configurator supports non-English characters in:

- All value fields
- Log file and trace file path (specified in the **Trace/Log files** tab)

The drop list for the `CharacterEncoding` and `Locale` standard configuration properties displays only a subset of supported values. To add other values to the drop list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory.

For example, to add the locale `en_GB` to the list of values for the `Locale` property, open the `stdConnProps.xml` file and add the line in boldface type below:

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
                <ValidType>String</ValidType>
            <ValidValues>
                                <Value>ja_JP</Value>
                                <Value>ko_KR</Value>
                                <Value>zh_CN</Value>
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
```

```
                                <Value>it_IT</Value>
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>
                        <DefaultValue>en_US</DefaultValue>
                </ValidValues>
        </Property>
```

Connector standard properties

This appendix describes the standard configuration properties for the connector component of WebSphere Business Integration adapters. The information covers connectors running on the following integration brokers:

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator, WebSphere MQ Integrator Broker, and WebSphere Business Integration Message Broker, collectively referred to as the WebSphere Message Brokers (WMQI).
- WebSphere Application Server (WAS)

Not every connector makes use of all these standard properties. When you select an integration broker from Connector Configurator, you will see a list of the standard properties that you need to configure for your adapter running with that broker.

For information about properties specific to the connector, see the relevant adapter user guide.

Note: In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes and follow the conventions for each operating system.

New and deleted properties

These standard properties have been added in this release.

New properties

- XMLNameSpaceFormat

Deleted properties

- RestartCount

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

This section describes the standard configuration properties. For information on configuration properties specific to a connector, see its adapter user guide.

Using Connector Configurator

You configure connector properties from Connector Configurator, which you access from System Manager. For more information on using Connector Configurator, refer to the Connector Configurator appendix.

Note: Connector Configurator and System Manager run only on the Windows system. If you are running the connector on a UNIX system, you must have a Windows machine with these tools installed. To set connector properties for a connector that runs on UNIX, you must start up System Manager on the Windows machine, connect to the UNIX integration broker, and bring up Connector Configurator for the connector.

Setting and updating property values

The default length of a property field is 255 characters.

The connector uses the following order to determine a property's value (where the highest number overrides other values):

1. Default
2. Repository (only if WebSphere InterChange Server is the integration broker)
3. Local configuration file
4. Command line

A connector obtains its configuration values at startup. If you change the value of one or more connector properties during a run-time session, the property's **Update Method** determines how the change takes effect. There are four different update methods for standard connector properties:

- **Dynamic**
  The change takes effect immediately after it is saved in System Manager. If the connector is working in stand-alone mode (independently of System Manager), for example with one of the WebSphere message brokers, you can only change properties through the configuration file. In this case, a dynamic update is not possible.
- Component restart
  The change takes effect only after the connector is stopped and then restarted in System Manager. You do not need to stop and restart the application-specific component or the integration broker.
- Server restart
  The change takes effect only after you stop and restart the application-specific component and the integration broker.
- Agent restart (ICS only)
  The change takes effect only after you stop and restart the application-specific component.

To determine how a specific property is updated, refer to the **Update Method** column in the Connector Configurator window, or see the Update Method column in the Property Summary table below.

Summary of standard properties

Table 13 on page 141 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on `RepositoryDirectory`.

You must set the values of some of these properties before running the connector. See the following section for an explanation of each property.

Table 13. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	Valid JMS queue name	*CONNECTORNAME* /ADMININQUEUE	Component restart	Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	*CONNECTORNAME*/ADMINOUTQUEUE	Component restart	Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	Delivery Transport is MQ or IDL: Repository directory is <REMOTE>
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	Application name	Value specified for the connector application name	Component restart	
BrokerType	ICS, WMQI, WAS			
CharacterEncoding	ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 **Note**: This is a subset of supported values.	ascii7	Component restart	
ConcurrentEventTriggeredFlows	1 to 32,767	1	Component restart	Repository directory is <REMOTE>
ContainerManagedEvents	No value or JMS	No value	Component restart	Delivery Transport is JMS
ControllerStoreAndForwardMode	true or false	True	Dynamic	Repository directory is <REMOTE>
ControllerTraceLevel	0-5	0	Dynamic	Repository directory is <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	MQ, IDL, or JMS	JMS	Component restart	If Repository directory is local, then value is JMS only
DuplicateEventElimination	True or False	False	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	Component restart	JMS transport only

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
jms.FactoryClassName	CxCommon.Messaging.jms .IBMMQSeriesFactory or CxCommon.Messaging .jms.SonicMQFactory or any Java class name	CxCommon.Messaging. jms.IBMMQSeriesFactory	Component restart	JMS transport only
jms.MessageBrokerName	If FactoryClassName is IBM, use crossworlds.queue. manager. If FactoryClassName is Sonic, use localhost:2506.	crossworlds.queue.manager	Component restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	10	Component restart	JMS transport only
jms.Password	Any valid password		Component restart	JMS transport only
jms.UserName	Any valid name		Component restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	128m	Component restart	Repository directory is <REMOTE>
JvmMaxNativeStackSize	Size of stack in kilobytes	128k	Component restart	Repository directory is <REMOTE>
JvmMinHeapSize	Heap size in megabytes	1m	Component restart	Repository directory is <REMOTE>
ListenerConcurrency	1- 100	1	Component restart	Delivery Transport must be MQ
Locale	en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR **Note:** This is a subset of the supported locales.	en_US	Component restart	
LogAtInterchangeEnd	True or False	False	Component restart	Repository Directory must be <REMOTE>
MaxEventCapacity	1-2147483647	2147483647	Dynamic	Repository Directory must be <REMOTE>
MessageFileName	Path or filename	InterchangeSystem.txt	Component restart	
MonitorQueue	Any valid queue name	*CONNECTORNAME*/MONITORQUEUE	Component restart	JMS transport only: DuplicateEvent Elimination must be True
OADAutoRestartAgent	True or False	False	Dynamic	Repository Directory must be <REMOTE>

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
OADMaxNumRetry	A positive number	1000	Dynamic	Repository Directory must be <REMOTE>
OADRetryTimeInterval	A positive number in minutes	10	Dynamic	Repository Directory must be <REMOTE>
PollEndTime	HH:MM	HH:MM	Component restart	
PollFrequency	A positive integer in milliseconds no (to disable polling) key (to poll only when the letter p is entered in the connector's Command Prompt window)	10000	Dynamic	
PollQuantity	1-500	1	Agent restart	JMS transport only: Container Managed Events is specified
PollStartTime	HH:MM(HH is 0-23, MM is 0-59)	HH:MM	Component restart	
RepositoryDirectory	Location of metadata repository		Agent restart	For ICS: set to <REMOTE> For WebSphere MQ message brokers and WAS: set to C:\crossworlds\repository
RequestQueue	Valid JMS queue name	*CONNECTORNAME*/REQUESTQUEUE	Component restart	Delivery Transport is JMS
ResponseQueue	Valid JMS queue name	*CONNECTORNAME*/RESPONSEQUEUE	Component restart	Delivery Transport is JMS: required only if Repository directory is <REMOTE>
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	A sensible positive value in minutes: 1 - 2147483547	1	Dynamic	
RHF2MessageDomain	mrm, xml	mrm	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML.

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
SourceQueue	Valid WebSphere MQ name	*CONNECTORNAME*/SOURCEQUEUE	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Component restart	Delivery Transport is JMS
SynchronousRequestTimeout	0 - any number (millisecs)	0	Component restart	Delivery Transport is JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
WireFormat	CwXML, CwBO	CwXML	Agent restart	CwXML if Repository Directory is not <REMOTE>: CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequest Timeout	0 - any number (millisecs)	0	Component restart	WAS only
XMLNameSpaceFormat	short, long	short	Agent restart	WebSphere MQ message brokers and WAS only

Standard configuration properties

This section lists and defines each of the standard connector configuration properties.

AdminInQueue

The queue that is used by the integration broker to send administrative messages to the connector.

The default value is CONNECTORNAME/ADMININQUEUE.

AdminOutQueue

The queue that is used by the connector to send administrative messages to the integration broker.

The default value is CONNECTORNAME/ADMINOUTQUEUE.

AgentConnections

Applicable only if RepositoryDirectory is <REMOTE>.

The AgentConnections property controls the number of ORB connections opened by orb.init[].

By default, the value of this property is set to 1. There is no need to change this default.

AgentTraceLevel

Level of trace messages for the application-specific component. The default is 0. The connector delivers all trace messages applicable at the tracing level set or lower.

ApplicationName

Name that uniquely identifies the connector's application. This name is used by the system administrator to monitor the WebSphere business integration system environment. This property must have a value before you can run the connector.

BrokerType

Identifies the integration broker type that you are using. The options are ICS, WebSphere message brokers (WMQI, WMQIB or WBIMB) or WAS.

CharacterEncoding

Specifies the character code set used to map from a character (such as a letter of the alphabet, a numeric representation, or a punctuation mark) to a numeric value.

Note: Java-based connectors do not use this property. A C++ connector currently uses the value ascii7 for this property.

By default, a subset of supported character encodings only is displayed in the drop list. To add other supported values to the drop list, you must manually modify the \Data\Std\stdConnProps.xml file in the product directory. For more information, see the appendix on Connector Configurator.

ConcurrentEventTriggeredFlows

Applicable only if RepositoryDirectory is <REMOTE>.

Determines how many business objects can be concurrently processed by the connector for event delivery. Set the value of this attribute to the number of business objects you want concurrently mapped and delivered. For example, set the value of this property to 5 to cause five business objects to be concurrently processed. The default value is 1.

Setting this property to a value greater than 1 allows a connector for a source application to map multiple event business objects at the same time and deliver them to multiple collaboration instances simultaneously. This speeds delivery of business objects to the integration broker, particularly if the business objects use complex maps. Increasing the arrival rate of business objects to collaborations can improve overall performance in the system.

To implement concurrent processing for an entire flow (from a source application to a destination application), you must:

- Configure the collaboration to use multiple threads by setting its Maximum number of concurrent events property high enough to use multiple threads.
- Ensure that the destination application's application-specific component can process requests concurrently. That is, it must be multi-threaded, or be able to use connector agent parallelism and be configured for multiple processes. Set the Parallel Process Degree configuration property to a value greater than 1.

The ConcurrentEventTriggeredFlows property has no effect on connector polling, which is single-threaded and performed serially.

ContainerManagedEvents

This property allows a JMS-enabled connector with a JMS event store to provide guaranteed event delivery, in which an event is removed from the source queue and placed on the destination queue as a single JMS transaction.

The default value is `No value`.

When ContainerManagedEvents is set to `JMS`, you must configure the following properties to enable guaranteed event delivery:

- PollQuantity = `1` to `500`
- SourceQueue = `CONNECTORNAME/SOURCEQUEUE`

You must also configure a data handler with the MimeType, DHClass, and DataHandlerConfigMOName (optional) properties. To set those values, use the **Data Handler** tab in Connector Configurator. The fields for the values under the Data Handler tab will be displayed only if you have set `ContainerManagedEvents` to `JMS`.

Note: When ContainerManagedEvents is set to `JMS`, the connector does *not* call its `pollForEvents()` method, thereby disabling that method's functionality.

This property only appears if the `DeliveryTransport` property is set to the value `JMS`.

ControllerStoreAndForwardMode

Applicable only if `RepositoryDirectory` is <REMOTE>.

Sets the behavior of the connector controller after it detects that the destination application-specific component is unavailable.

If this property is set to `true` and the destination application-specific component is unavailable when an event reaches ICS, the connector controller blocks the request to the application-specific component. When the application-specific component becomes operational, the controller forwards the request to it.

However, if the destination application's application-specific component becomes unavailable **after** the connector controller forwards a service call request to it, the connector controller fails the request.

If this property is set to `false`, the connector controller begins failing all service call requests as soon as it detects that the destination application-specific component is unavailable.

The default is `true`.

ControllerTraceLevel

Applicable only if `RepositoryDirectory` is <REMOTE>.

Level of trace messages for the connector controller. The default is `0`.

DeliveryQueue

Applicable only if `DeliveryTransport` is JMS.

The queue that is used by the connector to send business objects to the integration broker.

The default value is CONNECTORNAME/DELIVERYQUEUE.

DeliveryTransport

Specifies the transport mechanism for the delivery of events. Possible values are MQ for WebSphere MQ, IDL for CORBA IIOP, or JMS for Java Messaging Service.

- If ICS is the broker type, the value of the DeliveryTransport property can be MQ, IDL, or JMS, and the default is IDL.
- If the RepositoryDirectory is a local directory, the value may only be JMS.

The connector sends service call requests and administrative messages over CORBA IIOP if the value configured for the DeliveryTransport property is MQ or IDL.

WebSphere MQ and IDL: Use WebSphere MQ rather than IDL for event delivery transport, unless you must have only one product. WebSphere MQ offers the following advantages over IDL:

- Asynchronous communication:
  WebSphere MQ allows the application-specific component to poll and persistently store events even when the server is not available.
- Server side performance:
  WebSphere MQ provides faster performance on the server side. In optimized mode, WebSphere MQ stores only the pointer to an event in the repository database, while the actual event remains in the WebSphere MQ queue. This saves having to write potentially large events to the repository database.
- Agent side performance:
  WebSphere MQ provides faster performance on the application-specific component side. Using WebSphere MQ, the connector's polling thread picks up an event, places it in the connector's queue, then picks up the next event. This is faster than IDL, which requires the connector's polling thread to pick up an event, go over the network into the server process, store the event persistently in the repository database, then pick up the next event.

JMS: Enables communication between the connector and client connector framework using Java Messaging Service (JMS).

If you select JMS as the delivery transport, additional JMS properties such as jms.MessageBrokerName, jms.FactoryClassName, jms.Password, and jms.UserName, appear in Connector Configurator. The first two of these properties are required for this transport.

Important: There may be a memory limitation if you use the JMS transport mechanism for a connector in the following environment:

- AIX 5.0
- WebSphere MQ 5.3.0.1
- When ICS is the integration broker

In this environment, you may experience difficulty starting both the connector controller (on the server side) and the connector (on the client side) due to memory use within the WebSphere MQ client. If your installation uses less than 768M of process heap size, IBM recommends that you set:

- The LDR_CNTRL environment variable in the CWSharedEnv.sh script.

  This script resides in the \bin directory below the product directory. With a text editor, add the following line as the first line in the CWSharedEnv.sh script:

```
export LDR_CNTRL=MAXDATA=0x30000000
```

This line restricts heap memory usage to a maximum of 768 MB (3 segments * 256 MB). If the process memory grows more than this limit, page swapping can occur, which can adversely affect the performance of your system.

- The IPCCBaseAddress property to a value of 11 or 12. For more information on this property, see the *System Installation Guide for UNIX*.

DuplicateEventElimination

When you set this property to true, a JMS-enabled connector can ensure that duplicate events are not delivered to the delivery queue. To use this feature, the connector must have a unique event identifier set as the business object's **ObjectEventId** attribute in the application-specific code. This is done during connector development.

This property can also be set to false.

Note: When DuplicateEventElimination is set to true, you must also configure the MonitorQueue property to enable guaranteed event delivery.

FaultQueue

If the connector experiences an error while processing a message then the connector moves the message to the queue specified in this property, along with a status indicator and a description of the problem.

The default value is CONNECTORNAME/FAULTQUEUE.

JvmMaxHeapSize

The maximum heap size for the agent (in megabytes). This property is applicable only if the RepositoryDirectory value is <REMOTE>.

The default value is 128m.

JvmMaxNativeStackSize

The maximum native stack size for the agent (in kilobytes). This property is applicable only if the RepositoryDirectory value is <REMOTE>.

The default value is 128k.

JvmMinHeapSize

The minimum heap size for the agent (in megabytes). This property is applicable only if the RepositoryDirectory value is <REMOTE>.

The default value is 1m.

jms.FactoryClassName

Specifies the class name to instantiate for a JMS provider. You *must* set this connector property when you choose JMS as your delivery transport mechanism (DeliveryTransport).

The default is CxCommon.Messaging.jms.IBMMQSeriesFactory.

jms.MessageBrokerName

Specifies the broker name to use for the JMS provider. You *must* set this connector property when you choose JMS as your delivery transport mechanism (DeliveryTransport).

The default is crossworlds.queue.manager.

jms.NumConcurrentRequests

Specifies the maximum number of concurrent service call requests that can be sent to a connector at the same time. Once that maximum is reached, new service calls block and wait for another request to complete before proceeding.

The default value is 10.

jms.Password

Specifies the password for the JMS provider. A value for this property is optional.

There is no default.

jms.UserName

Specifies the user name for the JMS provider. A value for this property is optional.

There is no default.

ListenerConcurrency

This property supports multi-threading in MQ Listener when ICS is the integration broker. It enables batch writing of multiple events to the database, thus improving system performance. The default value is 1.

This property applies only to connectors using MQ transport. The DeliveryTransport property must be set to MQ.

Locale

Specifies the language code, country or territory, and, optionally, the associated character code set. The value of this property determines such cultural conventions as collation and sort order of data, date and time formats, and the symbols used in monetary specifications.

A locale name has the following format:

ll_TT.codeset

where:

ll	a two-character language code (usually in lower case)
TT	a two-letter country or territory code (usually in upper case)
codeset	the name of the associated character code set; this portion of the name is often optional.

By default, only a subset of supported locales appears in the drop list. To add other supported values to the drop list, you must manually modify the \Data\Std\stdConnProps.xml file in the product directory. For more information, see the appendix on Connector Configurator.

The default value is en_US. If the connector has not been globalized, the only valid value for this property is en_US. To determine whether a specific connector has been globalized, see the connector version list on these websites:

http://www.ibm.com/software/websphere/wbiadapters/infocenter, or
http://www.ibm.com/websphere/integration/wicserver/infocenter

LogAtInterchangeEnd
Applicable only if RespositoryDirectory is <REMOTE>.

Specifies whether to log errors to the integration broker's log destination. Logging to the broker's log destination also turns on e-mail notification, which generates e-mail messages for the MESSAGE_RECIPIENT specified in the InterchangeSystem.cfg file when errors or fatal errors occur.

For example, when a connector loses its connection to its application, if LogAtInterChangeEnd is set to true, an e-mail message is sent to the specified message recipient. The default is false.

MaxEventCapacity
The maximum number of events in the controller buffer. This property is used by flow control and is applicable only if the value of the RepositoryDirectory property is <REMOTE>.

The value can be a positive integer between 1 and 2147483647. The default value is 2147483647.

MessageFileName
The name of the connector message file. The standard location for the message file is \connectors\messages. Specify the message filename in an absolute path if the message file is not located in the standard location.

If a connector message file does not exist, the connector uses InterchangeSystem.txt as the message file. This file is located in the product directory.

Note: To determine whether a specific connector has its own message file, see the individual adapter user guide.

MonitorQueue
The logical queue that the connector uses to monitor duplicate events. It is used only if the DeliveryTransport property value is JMS and DuplicateEventElimination is set to TRUE.

The default value is CONNECTORNAME/MONITORQUEUE

OADAutoRestartAgent
Valid only when the RepositoryDirectory is <REMOTE>.

Specifies whether the connector uses the automatic and remote restart feature. This feature uses the MQ-triggered Object Activation Daemon (OAD) to restart the connector after an abnormal shutdown, or to start a remote connector from System Monitor.

This property must be set to trueto enable the automatic and remote restart feature. For information on how to configure the MQ-triggered OAD feature. see the *Installation Guide for Windows* or *for UNIX*.

The default value is false.

OADMaxNumRetry
Valid only when the RepositoryDirectory is <REMOTE>.

Specifies the maximum number of times that the MQ-triggered OAD automatically attempts to restart the connector after an abnormal shutdown. The OADAutoRestartAgent property must be set to true for this property to take effect.

The default value is 1000.

OADRetryTimeInterval

Valid only when the RepositoryDirectory is <REMOTE>.

Specifies the number of minutes in the retry-time interval for the MQ-triggered OAD. If the connector agent does not restart within this retry-time interval, the connector controller asks the OAD to restart the connector agent again. The OAD repeats this retry process as many times as specified by the OADMaxNumRetry property. The OADAutoRestartAgent property must be set to true for this property to take effect.

The default is 10.

PollEndTime

Time to stop polling the event queue. The format is HH:MM, where *HH* represents 0-23 hours, and *MM* represents 0-59 seconds.

You must provide a valid value for this property. The default value is HH:MM, but must be changed.

PollFrequency

The amount of time between polling actions. Set PollFrequency to one of the following values:

- The number of milliseconds between polling actions.
- The word key, which causes the connector to poll only when you type the letter p in the connector's Command Prompt window. Enter the word in lowercase.
- The word no, which causes the connector not to poll. Enter the word in lowercase.

The default is 10000.

Important: Some connectors have restrictions on the use of this property. To determine whether a specific connector does, see the installing and configuring chapter of its adapter guide.

PollQuantity

Designates the number of items from the application that the connector should poll for. If the adapter has a connector-specific property for setting the poll quantity, the value set in the connector-specific property will override the standard property value.

PollStartTime

The time to start polling the event queue. The format is *HH:MM*, where *HH* represents 0-23 hours, and *MM* represents 0-59 seconds.

You must provide a valid value for this property. The default value is HH:MM, but must be changed.

RequestQueue

The queue that is used by the integration broker to send business objects to the connector.

The default value is CONNECTOR/REQUESTQUEUE.

RepositoryDirectory

The location of the repository from which the connector reads the XML schema documents that store the meta-data for business object definitions.

When the integration broker is ICS, this value must be set to <REMOTE> because the connector obtains this information from the InterChange Server repository.

When the integration broker is a WebSphere message broker or WAS, this value must be set to *<local directory>*.

ResponseQueue

Applicable only if DeliveryTransport is JMS and required only if RepositoryDirectory is <REMOTE>.

Designates the JMS response queue, which delivers a response message from the connector framework to the integration broker. When the integration broker is ICS, the server sends the request and waits for a response message in the JMS response queue.

RestartRetryCount

Specifies the number of times the connector attempts to restart itself. When used for a parallel connector, specifies the number of times the master connector application-specific component attempts to restart the slave connector application-specific component.

The default is 3.

RestartRetryInterval

Specifies the interval in minutes at which the connector attempts to restart itself. When used for a parallel connector, specifies the interval at which the master connector application-specific component attempts to restart the slave connector application-specific component. Possible values ranges from 1 to 2147483647.

The default is 1.

RHF2MessageDomain

WebSphere message brokers and WAS only.

This property allows you to configure the value of the field domain name in the JMS header. When data is sent to WMQI over JMS transport, the adapter framework writes JMS header information, with a domain name and a fixed value of mrm. A connfigurable domain name enables users to track how the WMQI broker processes the message data.

A sample header would look like this:

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

The default value is mrm, but it may also be set to xml. This property only appears when DeliveryTransport is set to JMSand WireFormat is set to CwXML.

SourceQueue

Applicable only if DeliveryTransport is JMS and ContainerManagedEvents is specified.

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see “ContainerManagedEvents” on page 146.

The default value is CONNECTOR/SOURCEQUEUE.

SynchronousRequestQueue

Applicable only if DeliveryTransport is JMS.

Delivers request messages that require a synchronous response from the connector framework to the broker. This queue is necessary only if the connector uses synchronous execution. With synchronous execution, the connector framework sends a message to the SynchronousRequestQueue and waits for a response back from the broker on the SynchronousResponseQueue. The response message sent to the connector bears a correlation ID that matches the ID of the original message.

The default is CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE

SynchronousResponseQueue

Applicable only if DeliveryTransport is JMS.

Delivers response messages sent in reply to a synchronous request from the broker to the connector framework. This queue is necessary only if the connector uses synchronous execution.

The default is CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE

SynchronousRequestTimeout

Applicable only if DeliveryTransport is JMS.

Specifies the time in minutes that the connector waits for a response to a synchronous request. If the response is not received within the specified time, then the connector moves the original synchronous request message into the fault queue along with an error message.

The default value is 0.

WireFormat

Message format on the transport.

- If the RepositoryDirectory is a local directory, the setting is CwXML.
- If the value of RepositoryDirectory is <REMOTE>, the setting isCwBO.

WsifSynchronousRequest Timeout

WAS integration broker only.

Specifies the time in minutes that the connector waits for a response to a synchronous request. If the response is not received within the specified, time then the connector moves the original synchronous request message into the fault queue along with an error message.

The default value is 0.

XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

A strong property that allows the user to specify short and long name spaces in the XML format of business object definitions.

The default value is `short`.

Chapter 8. Configuring database connection pools

At run-time, it takes time for processes to establish new connections to the database. You can minimize that time by establishing database connection pools in advance for use by your collaboration and map processes. Database connection pools can improve performance and enable direct database access from within a collaboration.

A database connection pool can be used by multiple collaborations and maps, and each collaboration or map can use multiple database connection pools.

This chapter contains the following sections:

When to use database connection pools

This section describes some of the situations in which you might want to use database connection pools.

Performing routing

The logic of a business process may dictate that a business object must be routed to different destination applications depending on the value in one or more fields in the business object.

For instance, a site might store and process customer entities in different applications depending on the value in an attribute such as `CustomerType`. The collaboration template would need to retrieve and evaluate the value in that attribute and make a decision as to which destination application to send the business object to based on the value. Although this can be accomplished with control flow structures in Java, the pairings between the values and destination applications are then hard-coded in the collaboration template. If they need to be changed because of a change in procedure, or added to because a new value and application are introduced into the interface, then the collaboration template must be modified and re-compiled, it must be re-deployed, and so forth. A much more flexible implementation stores the pairings of values and destination applications in a database table. To implement this sort of approach, do the following:

1. Create a database table which has one column to store the routing values and another column to store some information that associates the appropriate destination application with the routing value. Table 14 is an example of such a table.

Table 14. Routing table example

Routing value	Destination application value
Customer	AppA

Table 14. Routing table example (continued)

Routing value	Destination application value
Federal	AppB
Reseller	AppC
Academic	AppD

2. Create a database connection pool and database connection in System Manager as described in "Creating database connection pools and database connections" on page 158.
3. In the collaboration template design the logic to do the following:
   a. Retrieve the value in the attribute to be used for routing and store the value in a variable.
   b. Obtain a connection to the database.
   c. Execute a SQL query that retrieves the value in the column that stores the destination application values where the value in the column that stores routing values is equal to the value stored in the variable in the collaboration template.
   d. Use a decision node to cause the collaboration logic to branch depending on the value returned from the database table. The different branches should lead to different service call nodes responsible for sending the business object out to the appropriate connector and thereby sending it out to the appropriate destination application.

Performing lookups in database tables

You might need to translate one value into another by looking up its equivalent in a table. Frequently you perform these operations by implementing lookup relationships, though using a lookup relationship does not always make sense. Lookup relationships are designed primarily for situations where each application involved in an interface needs its own way of representing a piece of data—a participant is created for each application and the lookup relationship itself connects all the participants much in the same way that the integration broker connects the applications. Sometimes you have a need to transform a value into one of several other values, but there is not the need to maintain a separate representation of that data for each application involved in the interface. In such a case you should create a table in the repository to store the associated values and then use a database connection and SQL select statements to retrieve the desired value.

Furthermore, the API provided for lookup relationships makes it very easy to abstract related data across applications, but does not make it easy to perform more complicated queries. The lookup relationship API is designed to take a piece of data and return the key value that the data shares with the other pieces of application data in the relationship, or to take a key value and return a particular piece of data associated with it. The lookup relationship API cannot return multiple column values, however, or execute stored procedures, which the APIs of the `CwDBConnection` class are able to do.

You can also satisfy this requirement with Java code rather than with a database query, by using control structures such as "if/else" and "switch/case" statements. Consider the following advantages and disadvantages of the different approaches and make the appropriate choice based on the situation:

- It is easier to add new records (such as a new routing value paired with a new destination application) when using a database connection because you only have to add a new row to the database; no components have to be modified, and the new pairing is effectively immediate. Adding new associations in Java code, however, is less convenient because you must modify the code of the component, recompile it, and possibly stop and restart it (if the component is a collaboration). You should pause the components of the interface beforehand as well, to ensure that there no transactions are in progress while the component is being compiled, or they may fail.
- Performing a lookup in Java code should be faster than making a connection to a database and performing a query, though this depends on how many associations there are that the Java control structure must iterate through.
- It is less convenient to migrate an interface that uses a database connection to another environment than it is to migrate an interface that exclusively relies on Java code, because you must remember to migrate not only the map or collaboration that makes the connection, but must also remember to recreate the tables involved in the new environment.

Persisting information

Some customers want to make information about the operation of the business integration system persistent by storing that information in a database so that it can be referenced for problem resolution or historical analysis.

To satisfy this requirement, do the following:

1. Create a database table with as many columns as necessary to store the desired data.
2. Create a database connection pool and database connection in System Manager as described in "Creating database connection pools and database connections" on page 158.
3. In the collaboration template design the logic to do obtain a connection to the database and execute a SQL query that inserts the desired data into the appropriate column.

Typically this requirement involves persisting information contained in the business object being processed by the collaboration (such as the primary key of the entity being processed), or information about the system itself (such as the successful processing of a business object request). As of release 4.2.0, you can persist business object data like primary key values by using the Business Object Probe feature; for more information, see the *Collaboration Development Guide*. As of release 4.2.0, you can persist some information about the system by using the Persistent Monitoring feature; for more information, see the *System Administration Guide*.

When not to use database connection pools

You should not use database connection pools to connect directly to the database of an application. To interact with an application database you should only use IBM WebSphere business integration adapters. Database connections should only be made to databases that do not support an application because adapters use the business logic provided by programming interfaces of the application. If you connect directly to an application database to perform a SQL update statement, for example, then you circumvent any related logic that the API would have performed in response to an update operation. This violates the integrity of the application and the business process.

If you have a need to retrieve information from an application, but do not want to use an adapter because of the impact on performance of sending a business object request to the application and receiving the response, you have several alternatives:

- You can design a custom business object that only has attributes for the fields you need to affect in the operation. For instance, you may have a large application-specific business object designed for an application entity involved in a particular business process. If you need to retrieve just a small subset of information about that entity as part of another business process, you might be concerned about the performance impact of retrieving the full business object for just those few fields. To minimize the impact, you can make a copy of the application-specific business object and reduce its structure to only those fields that are required. That way only a small amount of information must be processed.

  This approach is best for retrieving information in the application database that changes frequently, because each request will retrieve information that is current at the time of the request.
- You can replicate the application information on the same server that hosts the IBM WebSphere InterChange Server databases. You can use the utilities provided by the database vendor to script the structure and data of a table into a file and then reproduce the table and its data in a database other than the one that the application itself uses.

  This approach is most useful for small amounts of information that is flat and static, such as a lookup table. You should not, however, use this approach for large entities that span multiple tables because the queries involved would be difficult to create and maintain, whereas using business objects to represent the entity would be easy for you to develop and for others to maintain. It is also not very good for tables that are very dynamic and have new records added frequently, because you either must manually update the replicated table frequently, or risk the interface having outdated information.

Creating database connection pools and database connections

A database connection pool consists of a number of reserved database connections. The reserved database connections are made available only to the collaboration and map processes that you design to use the pool.

To create a database connection pool, you define the values necessary for making a database connection beforehand. The IBM WebSphere InterChange Server business integration system saves this database connection information and uses it at runtime to more quickly establish connections for collaboration and mapping processes that you have assigned.

The configuration of database connection values that you define can be used by one or more pools. For each pool you will specify a number of connections; these connections will be allocated, used, and freed back to the pool.

Note: It is recommended that you connect to the InterChange Server instance that will use the database connection pool in System Manager before you create the pool, so that you can validate the connection.

Do the following to create a database pool:

1. Right-click the **Database Connection Pools** folder in your integration component library in System Manager and choose **Create New Database Connection** from the context menu.

The “Database Connection” dialog appears.

Note: Although the name of the dialog would suggest that what you are creating at this point is a database connection, it is really a database connection pool. In later steps you will define database connections within the pool.

2. Select the appropriate value in the **Database Driver** drop-down menu—either `DB2 (Type 2)`, `MQ SQL Server(Type 4)`, or `Data Direct Oracle(Type 4)`—depending on the database vendor.
3. If you chose `MQ SQL Server(Type 4)`, or `Data Direct Oracle(Type 4)` in the **Database Driver** drop-down menu, then you must type the name of the computer on which the database server resides in the **Host name** field.
4. Type the name of the database in the **Database** field.
5. If you chose `MQ SQL Server(Type 4)`, or `Data Direct Oracle(Type 4)` in the **Database Driver** drop-down menu, then you must type the port number through which clients communicate with the database server in the **Port number** field.
6. If you are connected to the InterChange Server instance that will use the pool, select that instance from the **Connected Servers** drop-down menu.
7. Type a name for the pool in the **DBConnection Name** field. You specify this database connection pool name when writing Java code to establish the connection in maps or collaboration templates.
8. Type the user name that should be used by InterChange Server to log in to the specified database in the **Login** field.
9. Type the password for the user name specified in step 6d on page 101 in the **Password** field.
10. Type the maximum number of connections that should be established by the pool for all the individual database connection objects you plan to create within it in the **Maximum connections** field, or enable the **Unlimited** checkbox to allow as many connections to be established as are permitted by the database server configuration and licensing.

    Warning: Be careful when working in this field. It is not a single-line text field, even though it should be, so you can accidentally press **Enter**. You will not see the value you entered then, and will justifiably try to re-type the value. Then when you try to finish creating the database connection pool you receive an error that a valid value must be entered for the field. Do not press **Enter** in this field.

    Figure 46 on page 160 shows the “New Database Connection Pool” wizard.

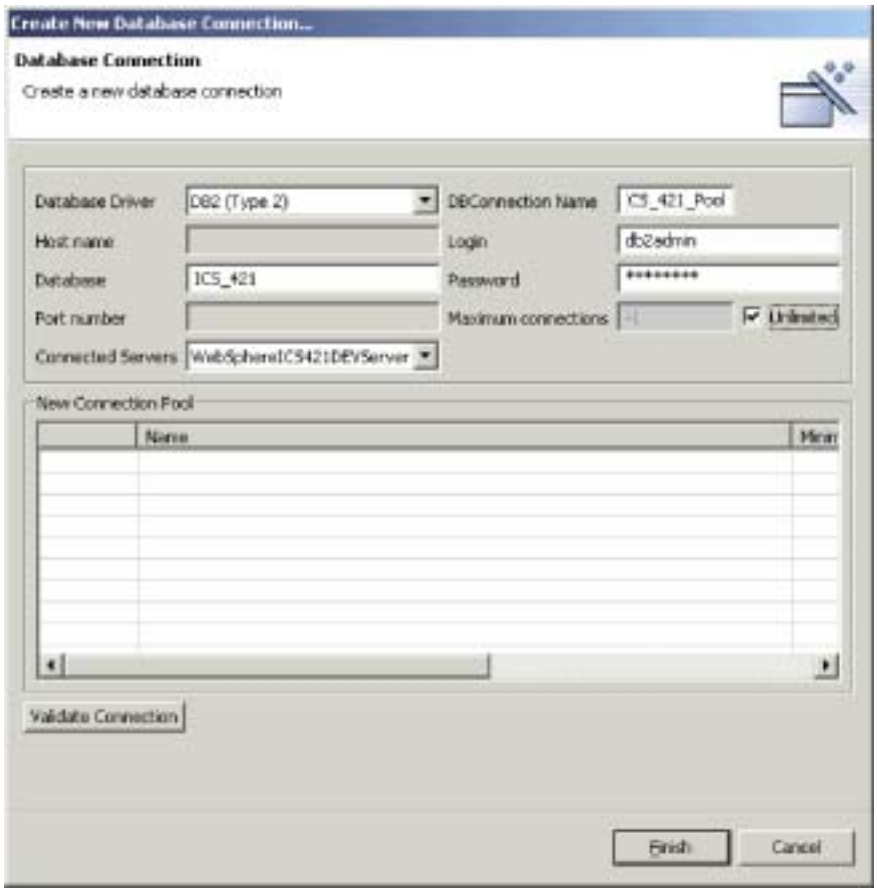

Figure 46. Create New Database Connection wizard

11. To create a new database connection object in the pool, right-click a row in the **New Connection Pool** table at the bottom of the wizard and choose **New Connection Pool** from the context menu.

    Note: Although the interface would suggest that what you are creating at this point is also a database connection pool, it is really a database connection object that you are creating within the pool you just created.

    The "Connection Pool" dialog appears.
12. Type a name for the database connection object in the **Name** field.
13. In the **Minimum connections** field, enter the minimum number of connections that the database connection object should establish. Note that the sum of the minimums for all the pools that you create in pool cannot exceed the maximum number that you specify for the database connection pool object itself. System Manager does not allow you to attempt to add a connection object or modify an existing connection object so that the maximum number of connections for the pool would be exceeded. Figure 47 on page 161 shows the "Connection Pool" dialog.

Figure 47. "Connection Pool" dialog

14. Click **Finish**.

    System Manager saves the database connection pool object and an icon appears under the **Database Connection Pools** folder in the integration component library.

15. Use the wizard to create another database connection pool at this time or click **Cancel** to close the wizard.

Validating database connection pools

After creating a database connection pool you should validate it to make sure it can be used by maps and collaborations at runtime.

Do the following to validate a database connection pool:

1. Start the InterChange Server instance.
2. Connect System Manager to the InterChange Server instance as described in "Connecting to InterChange Server" on page 39.
3. Do one of the following:
   - Right-click either the database connection pool object in an integration component library or a shortcut to it in a user project, then select **Validate Connection** from the context menu, then select the InterChange Server instance name from the submenu.
   - Right-click either the database connection pool object in the server repository in the InterChange Server Component Management view, then select **Validate Connection** from the context menu.
4. If the database connection pool is valid then System Manager displays a prompt indicating that the validation was successful.

   If System Manager displays an error that validation was not successful, resolve the problem by verifying the availability of the database, the privileges of the user account specified, and the accuracy of the configuration information.

Modifying database connection pools

You can modify some properties of both database connection pools and database connection objects.

To modify a database connection pool, do one of the following:

- Right-click either the database connection pool component in an integration component library, or the shortcut to it a user project and select **Properties** from the context menu.
- Select the database connection pool component in an integration component library, or the shortcut to it a user project and do one of the following:

– Select **Component > Properties** from the menu bar.
– Press **Alt+Enter**.

The following sections document the properties of database connection pools and database connection objects that you can change:

- "Modifying login information"
- "Modifying the number of connections"

Modifying login information

Many companies follow a practice of changing passwords at a regular interval to guard against old accounts being used to violate security. If this is the case at the site then you may need to change the login information for the database connection pool.

To change the login name used to access the database resource, type the new value into the Login field in the "Database Connection" screen.

To change the password for the specified login, click **Change** in the "Database Connection" screen, then do the following at the "Password Change" prompt:

1. Type the current password in the **Old Password** field.
2. Type the new password in the **New Password** field.
3. Type the new password again in the **Confirm Password** field.

Modifying the number of connections

You can modify the maximum number of connections for a database connection pool, and the minimum number of connections for a database connection object. The sum of all the minimum connections of all the database connection objects defined in a database connection pool may not exceed the maximum number of connections specified for the pool itself.

Modifying the maximum connections for a database connection pool

Do the following to modify the maximum number of connections for a database connection pool:

1. Access the properties of the database connection pool you want to edit, as described in "Modifying database connection pools" on page 161.
2. Do one of the following to modify the maximum number of connections that can be established by the total number of connection objects in the pool:
   - Type the maximum number of connections that should be established by the pool in the **Maximum connections** field.
   - Enable the **Unlimited** checkbox to allow as many connections to be established as are permitted by the database server configuration and licensing.
3. Click **OK**.

Modifying the minimum connections for a database connection object

Do the following to modify the maximum number of connections for a connection object:

1. Access the properties of the database connection pool you want to edit, as described in "Modifying database connection pools" on page 161.

2. In the “New Connection Pool” pane, right-click the connection object for which you want to change the minimum number of connections and select **Edit** from the context menu.
3. Type the new value in the **Minimum connections** field.
4. Click **OK**.

Deleting database connection objects

Do the following to delete a database connection object:

1. Access the properties of the database connection pool as described in “Modifying database connection pools” on page 161.
2. In the “New Connection Pool” pane, right-click the connection object you want to delete and select **Delete** from the context menu.
3. Click **OK**.

Using database connection pools in collaborations and maps

To use database connection pools in maps and collaboration templates, you use the `getDBConnection()` method in the `BaseCollaboration` class:

```
CwDBConnection getDBConnection(String ConnectionPoolName);
```

The string *ConnectionPoolName* must match exactly a pool name that you have created or will create in System Manager. If the name does not match exactly, or if you set connection pool name values in a collaboration template or map but do not create the database connection pool in System Manager, the collaboration object or map will fail.

For collaborations, you can set pool names in a template either before or after you have created the collaboration objects from that template; you do not need to reconfigure new collaboration objects to match a new connection pool name.

For more information on developing collaboration templates to use database connection pools, see the *Collaboration Development Guide*.

For more information on developing maps to use database connection pools, see the *Map Development Guide*.

Configuring transaction bracketing

You can configure a collaboration or a map to use database connection pools in either Implicit Database Transaction mode (in which the transaction begins as soon as the connection is made and ends when the process is finished) or Explicit Database Transaction mode (in which you begin and end the transaction programmatically).

To configure transaction bracketing for collaboration objects, see “Implicit database transaction” on page 180.

To configure transaction bracketing for maps, see the *Map Development Guide*.

Chapter 9. Configuring collaboration objects

This chapter describes configuration of a collaboration object and contains the following sections:

- "Collaboration objects and groups"
- "Creating a collaboration object" on page 166
- "Modifying collaboration objects" on page 170

For information about designing and creating a collaboration template, and for information about using Process Designer, see the *Collaboration Development Guide*.

Collaboration objects and groups

A collaboration object is an instance of a collaboration template. To configure a collaboration, you must:

1. Create a new collaboration object from a collaboration template. For more information, see "Creating a collaboration object" on page 166.
2. Bind the ports of the collaboration object, either internally (to a connector or another collaboration) or externally (to an external process that calls the collaboration, such as an e-business web application servlet). For more information, see "Binding collaboration object ports" on page 172.
3. Configure the general properties of the collaboration object. For more information, see "Configuring collaboration object general properties" on page 175.
4. Configure the collaboration-specific properties for the collaboration object. For more information, see "Configuring collaboration-specific properties" on page 181.

When you create an internal binding, you can bind only to connectors or collaborations that support the business object expected by the collaboration port. This allows communication between the bound components, allowing the collaboration to send and receive business objects, as well as receive responses to requests. A collaboration cannot run until you bind all its ports.

You can create multiple collaboration objects from a single collaboration template. For example, two collaborations from the same template can implement the same logic across two sets of applications at your site.

Collaboration-object groups

A collaboration-object group is a set of two or more collaboration objects that are bound to one another.

When you configure a collaboration object's port to receive an incoming business object from another collaboration object or to send an outgoing business object to another collaboration object, the combination of collaboration objects is a collaboration-object group. Any number of collaboration objects can be bound into a group.

The behavior of collaboration objects that are members of a collaboration-object group is different from the behavior of those that are not members of a group.

When you apply a command such as start, pause, or stop to a collaboration object that is a member of a group, the command affects all members of the group.

All members of a collaboration-object group must support the same transaction level in order for one member of the group to execute transactionally. If you bind a collaboration object that has transactional level `None` to a collaboration object that has transaction level `Best Effort`, both run at level `None`.

Creating a collaboration object

This section describes how to create a new collaboration object using the System Manager wizard. It describes the general flow of the wizard and references other sections in this chapter to address the specifics of collaboration object configuration. Do the following to create a collaboration object using the wizard:

1. In System Manager, right-click on the **Collaboration Objects** folder in an integration component library and choose **Create New Collaboration Object** from the context menu.

   The "Create New Collaboration" wizard appears, listing the installed templates in the **Template Name** column and a description (if provided by the developer) in the **Description** column. The dialog is the first screen in a wizard that steps you through the initial creation and configuration of a collaboration object. You can later modify the configuration values. Figure 48 shows the "Select collaboration template" screen:

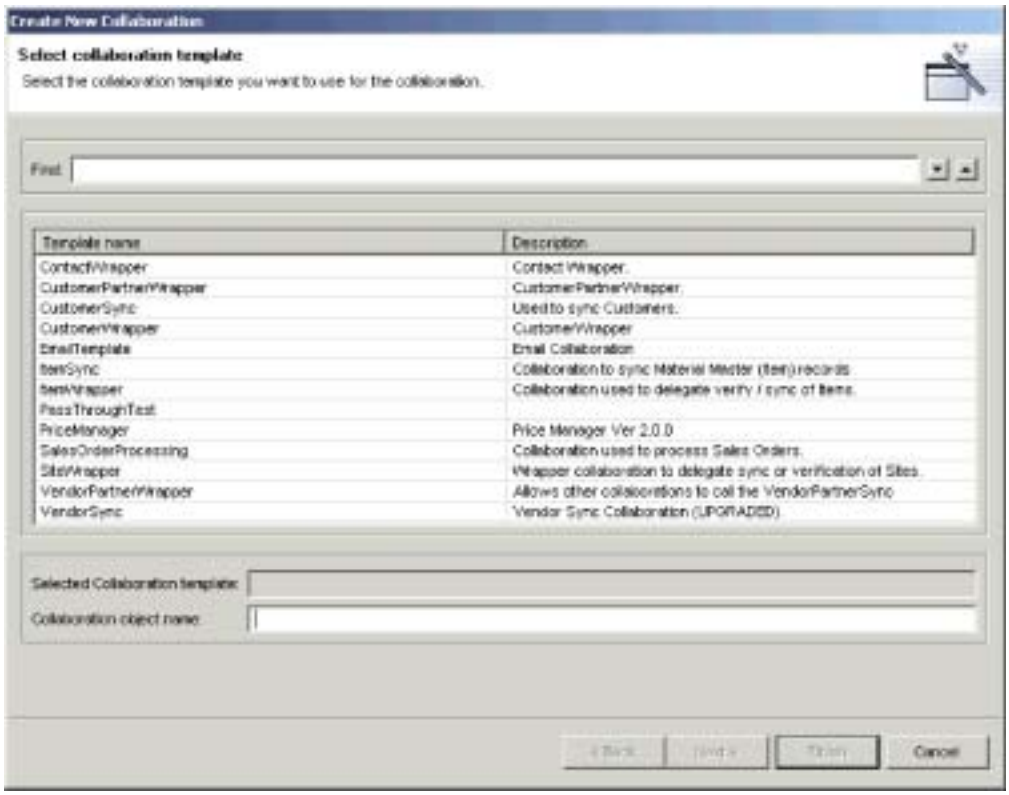

Figure 48. Selecting a collaboration template

2. Select the collaboration template upon which the object should be based from the list of templates. The name you select is displayed in the **Selected Collaboration template** field below the list.

You can also type text in the **Find** field and the dialog will select a collaboration template whose name begins with the characters you type in the field. You can then use the up and down arrows to navigate among templates that share the pattern of characters, if there are several that match.

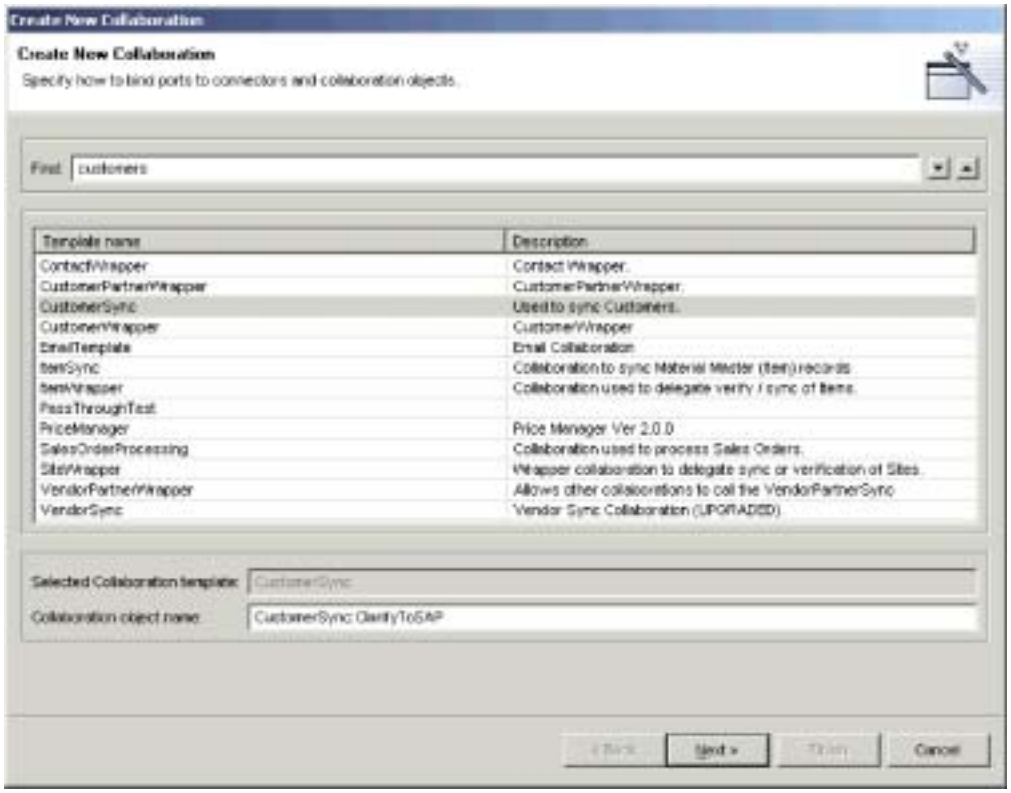

Figure 49. Using the Find field

3. Type a name for the collaboration object in the **Collaboration object name** field.

   Refer to the *Naming Components Guide* for requirements and conventions regarding collaboration object names.

4. Click **Next** to advance the wizard to the Bind ports screen.

   The “Bind ports” screen displays the ports defined in the collaboration template, the business object definition supported by each port, the type of component to which the port might be bound, and the particular components of the specified type to which each port can be bound. Figure 50 on page 168 shows the “Bind ports” screen:

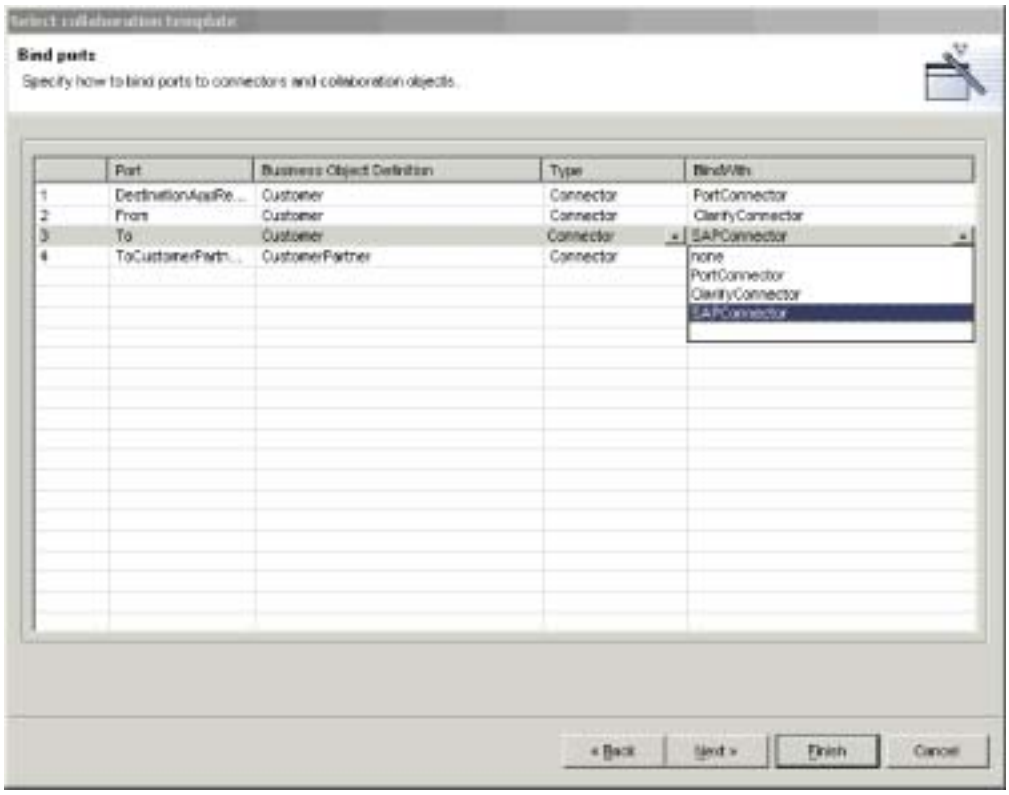

Figure 50. Binding collaboration ports

Depending on the value in the **Type** column, the **Bind With** column displays either the connectors or the collaborations that support the business object type in the **Business Object Definition** field. For each port displayed, choose either `Collaboration`, `Connector` or `Web Service` from the **Type** column and then choose the desired component in the **Bind With** column.

To configure a port so that it can receive business object requests from an external programmatic entity (such as a servlet), you must follow the instructions in "Configuring external port bindings" on page 174.

You can bind some, all, or none of the ports at this screen of the wizard. If you do not bind all the ports at this point you can always modify the port bindings after finishing the wizard. This way, if a particular component that you have to bind to a port does not support the necessary business object definition you can finish the collaboration object wizard, add support for the business object definition to the required component, and then reconfigure the collaboration object.

For more information about binding ports, and information about how to modify port bindings, see "Binding collaboration object ports" on page 172.

5. Click **Next** to advance the wizard to the "Collaboration General Properties" screen. This screen allows you to configure the properties that belong to all collaboration templates defined in the WebSphere InterChange Server business integration system. Figure 51 on page 169 shows the "Collaboration General Properties" screen:

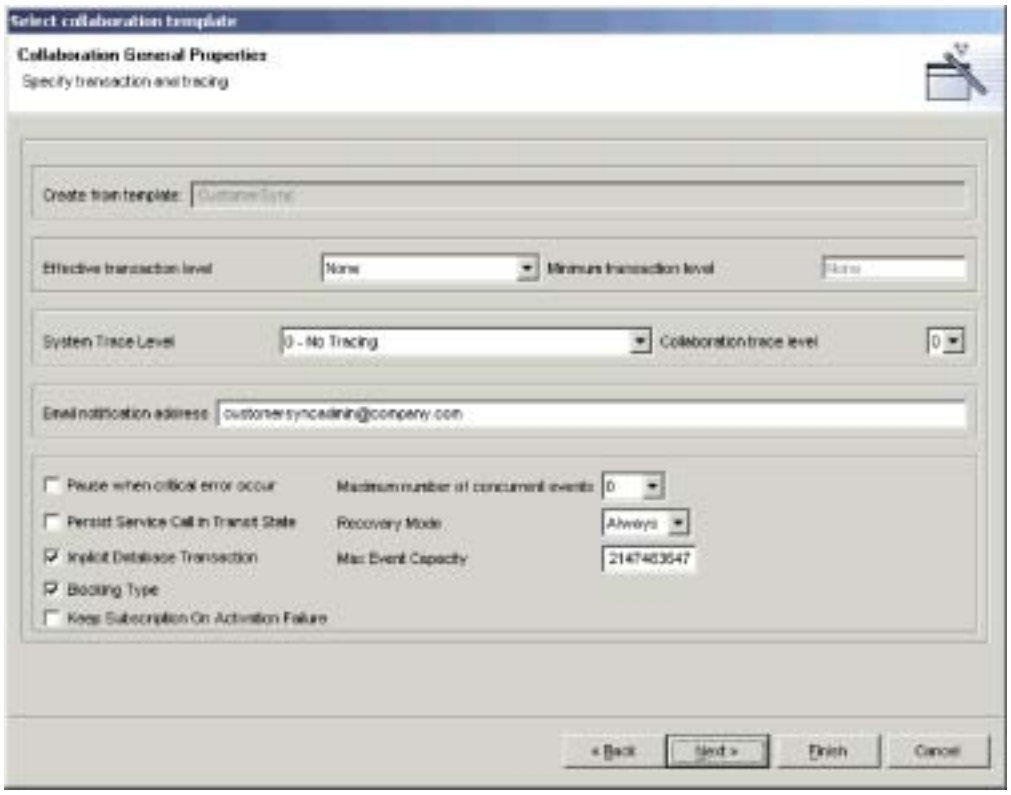

Figure 51. Configuring collaboration object general properties

Set each property to the desired value.

For more information about collaboration general properties and how to modify them for an existing collaboration object, see “Configuring collaboration object general properties” on page 175.

6. Click **Next** to advance the wizard to the Properties screen. This screen allows you to configure the properties that are specific to the particular collaboration template.

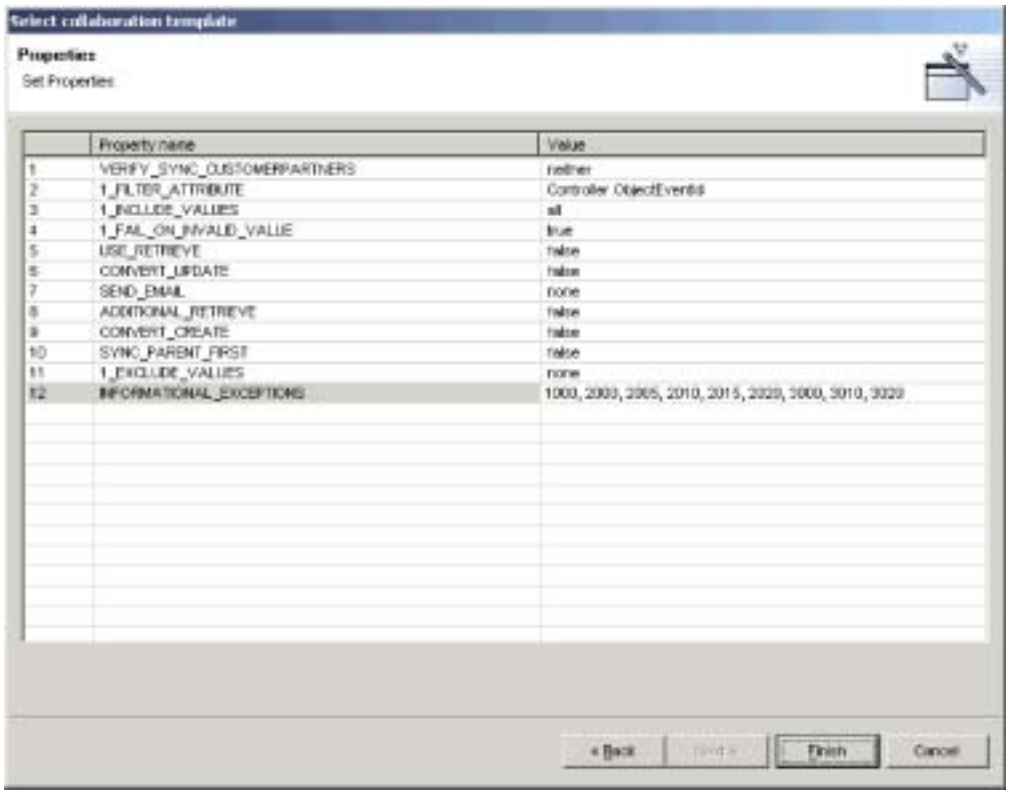

Figure 52. Configuring collaboration template-specific properties

Enter a value for each property or accept the default value. For information about each property, refer to the documentation for the particular collaboration template.

Click **Finish** to complete the collaboration object creation wizard. The new collaboration object is displayed in the Graphical View tab. For more information about the collaboration object graphical view and tree view, see “Collaboration object views” on page 171.

Modifying collaboration objects

This section describes how to modify a collaboration object. You might have to modify the port bindings or properties of a collaboration object for the following reasons:

- You created the collaboration object using the wizard as described in “Creating a collaboration object” on page 166, but had to bind a port to an external programmatic entity, which cannot be done through the wizard
- You created the collaboration object using the wizard as described in “Creating a collaboration object” on page 166, but the necessary components did not support the business object definitions associated with the ports, so you had to modify the component definitions
- You imported the collaboration object definition from another integration component library or InterChange Server and you need to make environment-specific changes
- You need to modify a collaboration object that has been running successfully in a production environment, but requires an administrative modification, such as changing the e-mail alias to which e-mail notifications are sent in the event of an error

To modify a collaboration object, double-click the collaboration object in the **Collaboration Objects** folder either in the Integration Component Libraries folder or in the **User Projects** folder in System Manager.

Collaboration object views

There are two view representations for collaboration objects: graphical view and tree view. The following sections describe them and their benefits.

Collaboration object graphical view

The collaboration object graphical view is the view shown by default after creating a new collaboration object. As shown in Figure 53, the graphical view represents the collaboration object with an icon in the center of the view and with icons representing each port in the collaboration radiating outward. This view is useful to demonstrate the “flow” of the business process: you can drag-and-drop ports around the view pane to position them in places that suggest the directionality of the collaboration communication. For instance, you can position the port that receives the triggering business object on the left-hand side of the pane and position the port that sends the business object to the destination application on the right-hand side of the pane.

Note: There is no way for the system to be able to automatically display the ports that receive triggering business objects on one side and the ports that send business objects to destination applications on another side. The developer of a collaboration template determines the naming and function of its ports and there is no way for the system to know the role a developer had in mind for a particular port. You should read the documentation for a collaboration template and then reposition the ports in the graphical view of any collaboration objects based on the template in a way that best facilitates your use of the tools.

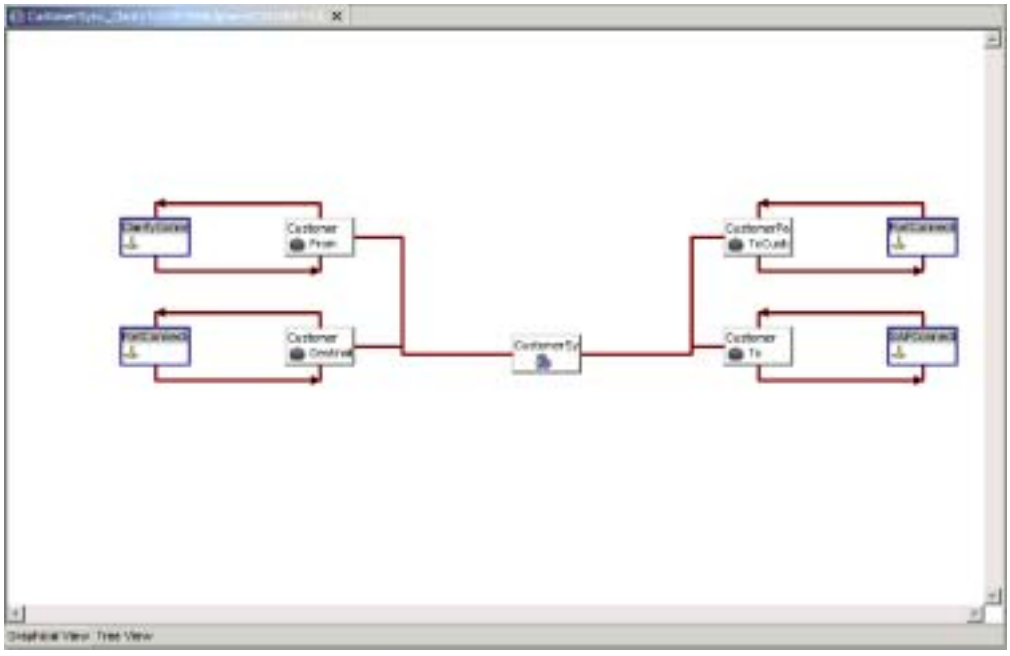

Figure 53. Collaboration object graphical view

Collaboration object tree view

You can switch to the collaboration object tree view from the graphical view by clicking the Tree View tab. As shown in Figure 54, the tree view represents the collaboration object with an icon at the top of a hierarchical tree and represents the ports with icons branching underneath it. Although this view does not suggest the "flow" of the business process very well, it presents the ports and their bound components in a very orderly manner, making it easier to find a particular port that you might need to reconfigure. This can be particularly helpful with collaboration objects based on templates that have many ports, which can appear very confusing in the graphical view. You cannot rearrange the icons in the collaboration object tree view.

Figure 54. Collaboration object tree view

Binding collaboration object ports

Collaboration ports are the interfaces through which collaborations send and receive business objects. You bind collaboration object ports to different components to configure them to exchange the type of business object supported by the port.

You can bind collaboration object ports when creating the collaboration object initially by using the wizard as described in "Creating a collaboration object" on page 166. You can also edit the port bindings after creating the collaboration object. To edit the port bindings in the graphical view, either double-click the icon for a port or right-click the icon for a port and choose **Bind Port** from the context menu. To edit the port bindings in the tree view, right-click the icon for a port and choose **Bind Port** from the context menu.

To bind a collaboration port to an internal component such as a connector or the port of another collaboration object, leave the radio button labeled **Internal** in the Type pane enabled and see "Configuring internal port bindings" on page 173.

To bind a collaboration port to an external programmatic entity such as a web servlet, enable the radio button labeled **External** in the Type pane and see "Configuring external port bindings" on page 174.

Configuring internal port bindings

To bind a collaboration port to a connector or to the port of another collaboration object, do the following:

1. Enable the radio button labeled **Internal** in the Type pane of the port configuration dialog.
2. Select the appropriate component type, either `Connector` or `Collaboration` in the Bind With pane.
3. Select the specific component from the list.
4. Click **OK**.

Figure 55 shows the port configuration dialog with an internal binding type.

Figure 55. Configuring internal collaboration object ports

Only the components that support the type of business object definition shown in the **Business Object Definition** field are listed in the dialog. If you do not see a component you expect to see listed then it is either not of the specified type (and you should change the **Bind With** radio button), or the component does not support the business object definition. If the **Bind With** radio button is set to the proper component type and the component you expect to see is not shown, then modify the definition for the particular component to add support for the business object definition and then launch the port configuration dialog afterwards.

You must bind all of the ports of a collaboration object to start it, so a collaboration object cannot run until you have bound all of its ports.

Some collaboration templates have ports defined that support an optional course of business logic. For instance, many collaboration templates are designed so that they can attempt to retrieve the entity that might have just been created in the destination application to ensure that the operation was successful. This sort of behavior is frequently optional and configurable through the collaboration-specific properties of the collaboration. All of the ports of a collaboration object must be bound to a component for it to start, as mentioned earlier, so even if you do not intend to take advantage of the optional behavior you must configure ports that might exist only to support it. In this case, just add support for the business object definition to an unused component (such as the `PortConnector`) and be sure to configure the collaboration properties to not use the optional functionality.

Configuring external port bindings

To bind a collaboration port to a connector or to the port of another collaboration object, do the following:

1. Enable the radio button labeled **External** in the Type pane of the port configuration dialog.
2. In the Configure As pane, select the **Incoming** radio button if business object requests are received by the port, or select the **Outgoing** radio button if business object responses are sent out of the port.
3. Do one of the following to associate the desired components with the port:
   - Drag-and-drop a business object definition from the integration component library folder into either the pane with the **Incoming Maps** or **Outgoing Maps** column, depending on whether business object requests are received by the port or if business object responses are sent out of the port.

     When presented with the Business Object Type dialog, either select the **Source Business Object** or **Destination Business Object** radio button, depending on whether the business object you drag-and-dropped is a source or destination object in the map that transforms it.
   - Drag-and-drop a map definition that transforms the type of business object supported by the port from the integration component library folder into either the pane with the **Incoming Maps** or **Outgoing Maps** column, depending on whether business object requests are received by the port or if business object responses are sent out of the port.
4. Click **OK**.

Figure 56 on page 175 shows the port configuration dialog with an external binding type.

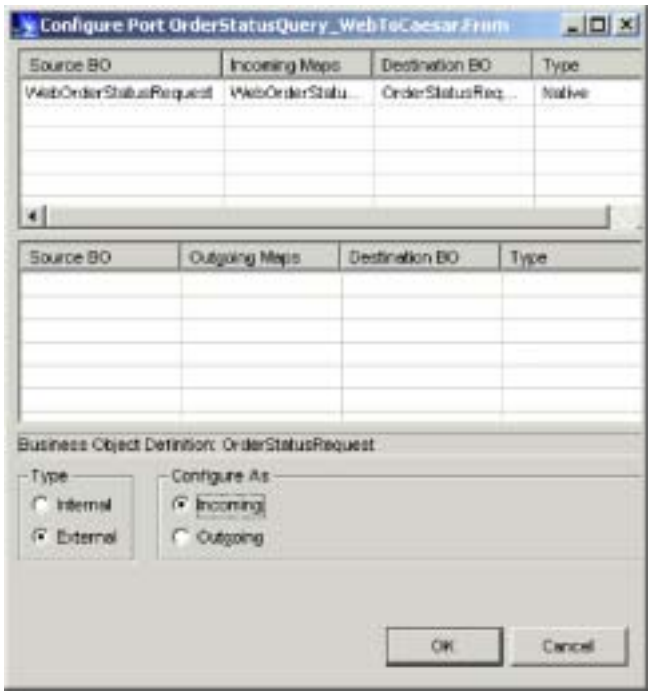

Figure 56. Configuring external collaboration object ports

For more information on implementing integrations with external programmatic entities, see the *Access Development Guide*.

Configuring collaboration object general properties

Collaboration general properties are properties that belong to all collaboration objects, regardless of how the developer designed the template upon which the objects are based. They affect the behavior of the collaboration object in the system as a whole.

To modify a collaboration object's general properties, do the following:

1. Access the collaboration object Properties dialog by doing one of the following:
   - Right-click the collaboration object in an integration component library and select **Properties** from the context menu; the dialog displays the Collaboration General Properties tab by default
   - Double-click the collaboration object icon in graphical view; the dialog displays the Collaboration General Properties tab by default
   - Right-click the collaboration object icon in either graphical view or tree view and choose **Properties** from the context menu; the dialog displays the Collaboration General Properties tab by default
2. Set the property to the desired value. For a description of each property and their possible values, see Table 15 and the sections that follow.
3. Click **OK**.

Table 15. Collaboration object general properties

Property name	Possible values
Create from template	Read-only name of the template upon which the collaboration object is based
Effective transaction level	`None`, `Minimal Effort`, `Best Effort`, or `Stringent`

Table 15. Collaboration object general properties (continued)

Property name	Possible values
Minimum transaction level	None, Minimal Effort, Best Effort, or Stringent, depending on the design of the collaboration template
System trace level	0 - No Tracing, 1 - Collaboration operations, 2 - And collaboration events, 3 - And state transactions, 4 - And incoming/outgoing messages, 5 - And detailed message contents
Collaboration trace level	0 through 5
Email notification address	any valid e-mail alias
Pause when critical error occurs	checked or unchecked
Maximum number of concurrent events	**0** through **9999**
Persist service call in transit state	checked or unchecked
Recovery mode	**Always** or **Deferred**
Implicit database transaction	checked or unchecked
Max event capacity	an integral value between 1 and 2147483647
Blocking type	checked or unchecked

Create from template

This read-only text field displays the name of the collaboration template on which the collaboration object is based.

Effective transaction level

The effective transaction level is a range between the highest maximum transaction level of all collaboration objects and the lowest maximum transaction level of all connectors that are bound to the object.

You can lower the effective transaction level of a collaboration if you need to bind to a connector that does not support a certain transaction level.

To change the effective transaction level, choose the desired value from the **Effective transaction level** drop-down list and then click **OK**.

Note: You cannot bind objects whose transaction levels are incompatible. If you need to bind such objects to a collaboration, you must adjust the supported transaction levels accordingly.

Minimum transaction level

The minimum transaction level specifies the lowest transaction level of the collaboration template and any collaboration objects based on it.

For instance, if the developer of the collaboration template specified a minimum transaction level of Best Effort for the template, then all objects based on the template must operate at Best Effort or Stringent. As described in the section on "Effective transaction level," all components bound to the collaboration object must support its effective transaction level. The minimum transaction level is, then, a

way for the collaboration template developer to dictate the lowest transaction level at which the entire interface in which collaboration objects based on the template will execute.

The **Minimum transaction level** field is read-only when configuring a collaboration object; it can only be changed in the collaboration template.

For more information about making changes to collaboration templates, see the *Collaboration Development Guide*.

System trace level

You can configure collaboration objects so that information about the execution of the collaboration is reported in the server output. To do so, select the desired value in the **System trace level** drop-down menu. Table 16 describes the different levels and the types of information reported at each:

Table 16. System trace levels

System trace level	Reported information
0 - No Tracing	No information is traced at this level
1 - Collaboration operations	Traces the receipt of business objects from connectors and the starting of scenarios
2 - And collaboration events	Prints messages for level 1, as well as the start and completion of each scenario, including both forward execution and rollback
3 - And state transactions	Prints messages for levels 1 and 2, as well as the execution of each scenario decision block or action node
4 - And incoming/outgoing messages	Prints messages for levels 1 through 3, as well as the sending and receipt of each business object by each scenario
5 - And detailed message contents	Prints messages for levels 1 through 4, as well as printing the structure of the business object being processed, with the value of each attribute

Collaboration trace level

Collaboration developers can code collaboration templates with template-specific tracing. While system tracing (described in “System trace level”) offers tracing information about the collaboration runtime in general, collaboration tracing provides information about the specific collaboration; for instance:

- The developer might design the collaboration to report business decisions that are made based on the data in the business object being processed by the collaboration.
- The developer can provide system-type tracing but only do so at certain points in the execution of the collaboration. For instance, system trace level 5 provides full dumps of the business object data in the server output, which can be very helpful for debugging problems, but can also be very cumbersome to read through when it occurs several times during the course of a single collaboration execution. A collaboration developer can provide a full business object dump that is executed during a level of collaboration tracing, and only do it at one particularly problematic point in the flow of the collaboration, so that there is only one business object dump the user must find and read in the server output.

To set the collaboration trace level, select the desired value in the range of 0 through 5 in the **Collaboration trace level** drop-down menu.

To find out what kind of information is reported at the different trace levels for a particular collaboration, reference the documentation for that collaboration template.

For more information about making changes to collaboration templates to implement collaboration tracing, see the *Collaboration Development Guide*.

Email notification address

You can configure a collaboration object so that e-mail notifications are sent in the event of errors related to that specific object. This facilitates interface-specific administration if that is desired. For instance, a site might have one administrator responsible for errors related to the WebSphere InterChange Server itself, another administrator specifically for a customer account synchronization interface, and another administrator specifically for an order processing interface.

Type the SMTP-compliant e-mail addresses to which you want error notifications sent into the **Email notification address** field. You can type multiple addresses as long as they are separated by commas. For more information on enabling the system to send e-mail notifications, see "Configuring e-mail notification properties using System Manager" on page 108 and the *System Administration Guide*.

Many collaboration templates feature greater configurability with regard to when e-mail notifications are sent, above and beyond the setting of the value of the **Email notification address** field. The level of configurability depends entirely on the design of the collaboration template, so reference the documentation for the specific collaboration template for information on how it approaches e-mail notification.

Pause when critical error occurs

Errors can occur that prevent a collaboration from having its business object requests processed by connectors to which it sends those requests. Some such errors are:

- Inability of the connector to log in to the application.
- A time-out in the communication between the connector and the application.
- A connector agent changing to an unknown state.

If these types of errors occur, the collaboration sends the request to the connector but then the flow fails because of the problem. This happens for any request sent until the problem is resolved, and can result in a large number of failed flows for the interface if the transaction volume is high and the error persists for a significant amount of time.

You can configure a collaboration object so that it stops sending requests to a connector after a request fails due to a critical error. To do so, enable the **Pause when critical error occurs** checkbox.

For more information on critical errors, a collaboration object's response to them, and the functionality of this feature, see the *System Administration Guide*.

Maximum number of concurrent events

You can configure collaboration objects to process multiple event-triggered flows concurrently, which can improve throughput for the interface. To do so, set the

Maximum number of concurrent events drop-down menu to the number of events between `0` and `9999` that you want the collaboration object to process concurrently.

To truly receive the benefit of this collaboration ability you must also configure other components that participate in the interface to behave similarly. For more information, see "Implementing concurrent processing of event-triggered flows" on page 293.

Persist service call in transit state

It is possible for an error to occur after a collaboration has sent a business object request to any destination connectors to which it is bound and before it has received a response from them as to whether or not they successfully processed the request in their respective applications. If the connectors were unable to process the request then it needs to be processed again when the error is resolved. If the connectors were able to successfully process the request, though, and are in the process of sending InterChange Server a notification of that successful processing when the error occurs then InterChange Server would not receive the notification. The last record of the state of the request would indicate that the request still needs to be processed. As a result of this inaccurate state record the request would be processed a second time, resulting in duplicate data.

To guard against this complication when configuring non-transactional collaborations you can enable the **Persist service call in transit state** checkbox. This causes InterChange Server to persist any business object requests that are in transit to destination applications when an error occurs. The requests are not sent when the system recovers, reducing the risk of the request being processed twice in the destination applications. You can then use Flow Manager to examine the requests and can investigate the destination applications to determine whether or not the requests were processed successfully before the error occurred. You should discard any requests that were processed successfully and resubmit any that were not processed successfully.

There are also programmatic measures that can be taken to handle these types of transport-related failures. For more information, refer to the information on handling service call transport exceptions in the *Collaboration Development Guide*.

Recovery mode

In previous releases, if a fatal system error occurred then all flows being processed at the time of failure would be recovered when the system was restarted. The flows were all read from persistent storage and resubmitted for processing. If there were many such flows, the system memory could actually be consumed entirely by the act of retrieving the events into memory, possibly resulting in another fatal system error related to lack of sufficient memory. Furthermore, you could not effectively manage the system with the tools until InterChange Server had completed its recovery processing.

As of release 4.0.0 it is possible to defer recovery for a collaboration object by setting the **Recovery mode** drop-down menu to the value `Deferred`. If you configure a collaboration object for deferred recovery in this manner, any flows that are in progress at the time of a fatal system error are treated as failed flows and are not recovered immediately when the system is restarted. The administrator can then use Flow Manager to resubmit the flows after the system has restarted and any desired administrative actions have been taken prior to the recovery efforts.

Important: For some interfaces it is important that event be processed in the order in which they are received. If event sequencing is not important to the integrity of an interface then you can implement deferred recovery without regard to the order in which failed or new flows are processed. If event sequencing is important, however, and you need to implement deferred recovery then you must document and follow administrative procedures to ensure that event sequencing is maintained. For instance, the administrator must make sure that the collaboration does not receive and process new flows when the system is restarted before the failed flows are resolved. They can do this managing the startup or polling of the source connector agents that would send new events to the collaboration in question.

The need for deferred recovery has been mitigated by optimized recovery measures introduced in release 4.2.0. Rather than reading the entire business object data for in-progress transactions into memory, the server only reads enough information in memory to locate the business object in persistent storage. Although deferred recovery is still available, you may not require it because of this new optimized recovery approach.

Implicit database transaction

If the collaboration template upon which the collaboration object you are configuring is based implements transactional database logic then you need to configure the collaboration object for either implicit or explicit transaction bracketing. If the collaboration was developed so that the transaction semantics are handled explicitly by the code written by the developer then you should leave the **Implicit database transaction** checkbox cleared. If the collaboration was not developed with explicit management of the database transactions, however, then you should enable the **Implicit database transaction** checkbox.

For more information about implicit and explicit transaction bracketing, see the *Collaboration Development Guide.*

Max event capacity

As business objects are received by InterChange Server they are queued in memory for processing. Prior to release 4.2.0 it was possible for interfaces that have a high volume of transactions and a low rate of processing to have so many business objects queue up that InterChange Server experiences a fatal out-of-memory error.

You can mitigate the risk of out-of-memory errors by configuring a maximum event capacity for a collaboration object. To do so, set the **Max event capacity** field to a value that specifies the maximum number of events you want queued for a collaboration object. The system will not queue more events than the number configured for this property in memory; depending on how you configure the “Blocking type” on page 181 property of the collaboration object, the system then responds differently to the receipt of new business objects for the collaboration.

The valid range of values for this property is from 1 to 2147483647.

For more information on the **Blocking type** property, see “Blocking type” on page 181.

For more information on flow control properties of connector definitions, and on how to use Connector Configurator to modify connector definitions, see Chapter 7, “Configuring connectors,” on page 125.

For more information on system properties related to flow control, see the *System Administration Guide.*

Blocking type

Use the "Blocking type" property in conjunction with the "Max event capacity" on page 180 property to regulate the flow of business objects to a collaboration.

If you enable the **Blocking Type** checkbox when configuring a collaboration object and the number of events in the collaboration's in-memory queue equals the number specified for its **Max event capacity** property, then the connector controller responsible for sending business objects to the collaboration will cease to do so. When the number of events in the collaboration's queue is no longer equal to the number specified for its **Max event capacity** property, then the connector controller will resume sending events to the collaboration.

If you leave the **Blocking Type** checkbox cleared when configuring a collaboration object and the number of events in the collaboration's in-memory queue equals the number specified for its **Max event capacity** property, then new events submitted to the collaboration by the connector controller are stored persistently in the database. The events are then read from the database into memory as the events currently in the collaboration's queue are processed.

For more information on the **Max event capacity** property, see "Max event capacity" on page 180.

For more information on system properties related to flow control, see the *System Administration Guide*

Configuring collaboration-specific properties

Collaboration developers can design collaboration templates so that they have their own individual properties that can then be used to affect the business process logic executed at runtime. This provides a very flexible architecture so that collaboration templates can be customized to implementation-specific requirements.

To modify a collaboration object's template-specific properties, do the following:

1. Access the Properties dialog for the collaboration object by doing one of the following:
   - Right-click the collaboration object in an integration component library and select **Properties** from the context menu and then click the Properties tab
   - Double-click the collaboration object icon in graphical view and then click the Properties tab
   - Right-click the collaboration object icon in either graphical view or tree view and choose **Properties** from the context menu, then click the Properties tab
2. Type the desired value in the **Value** field for the property listed in the **Property name** field.
3. Click **OK**.

For information about the properties specific to a collaboration template and their valid values, see the documentation for the collaboration template.

For information on adding properties to a template or modifying the usage of existing properties in a collaboration template, see the *Collaboration Development Guide.*

Chapter 10. Using Relationship Manager

Relationship Manager allows you to view and perform operations on relationship runtime data, including participants and their data. For background information about relationships, see the *Map Development Guide*.

You create relationship definitions with Relationship Designer. At runtime, instances of the relationships are populated with the data that associates information from different applications. This relationship instance data is created when the maps that use the relationships execute. The data is stored in the relationship tables specified in the relationship definition. Relationship Manager provides a graphical interface to interact with the relationship tables regardless of the database vendor.

For each relationship instance, Relationship Manager displays a hierarchical listing of its participant definitions and participant instances, which are a set of key and non-key attributes. The relationship tree also provides detailed information about each of the participants in the relationship instance such as the type of entity, its value, and the date it was last modified. A relationship instance ID is automatically generated when the relationship instance is saved in the relationship table. Relationship Manager displays this instance ID at the top level of the relationship tree.

Figure 57 on page 184 shows a sample of a relationship tree in Relationship Manager for an identity relationship.

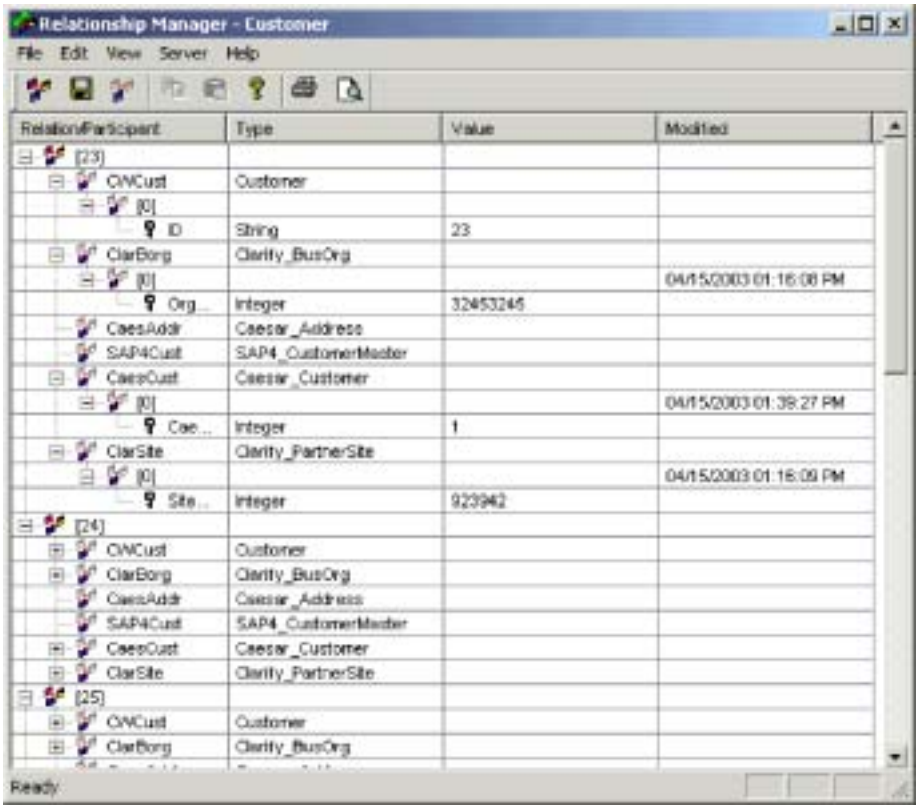

Figure 57. Relationship Manager

You can use Relationship Manager to work on entities at all levels: the relationship instance, participant instance, and attribute levels. For example, you can use Relationship Manager to:

- Create and delete relationship instances
- Modify the contents of a relationship instance, such as adding and deleting participants
- Add and save a participant's data, load a participant's data from or save data to a file, and copy and paste the data of a participant from another relationship into the relationship instance, thus creating a new participant (as long as the participant types are identical)
- Activate and deactivate participants
- Retrieve participants based on instance IDs, business object attribute values, or data
- Filter a participant's activity within a time interval
- Salvage a situation when problems with data arise. For example, when corrupt or inconsistent data from a source application has been sent to the generic and destination application relationship tables, you can use Relationship Manager to rollback (or clean up) the data back to a point of time when you know the data is reliable.

Starting Relationship Manager

Do one of the following to start Relationship Manager:

- Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Administrative > Relationship Manager**

- In Relationship Designer, select a relationship definition and then select **Tools > Relationship Manager** from the menu bar

Relationship Manager starts. At this point it is disconnected from the server; you must connect to an InterChange Server instance as described in "Connecting to InterChange Server" to proceed further.

Connecting to and disconnecting from a server

You must connect Relationship Manager to InterChange Server to work with relationship instances and data. Follow the instructions in the following sections to connect Relationship Manager to a server and disconnect Relationship Manager from it:

- "Connecting to InterChange Server"
- "Disconnecting from InterChange Server" on page 186

Connecting to InterChange Server

Do the following to connect Relationship Manager to InterChange Server:

1. Select **Server > Connect to Server** from the menu bar of Relationship Manager.
2. Do one of the following to populate the name of the InterChange Server instance to which you want to connect in the **Server Name** field:
   - Type the name of the InterChange Server instance in the **Server Name** field.

     Important: The name of an InterChange Server instance is case-sensitive, so you must be sure to be accurate when specifying the name.
   - Select a cached server name from the drop-down menu.
   - Do the following to browse for the InterChange Server instance on the network:
     a. Click the browse button.
     b. At the "Servers" dialog, select the desired InterChange Server instance from the list.
     c. Click **OK**.
3. Type the user name to interact with the InterChange Server instance in the **User Name** field.
4. Type the password for the user name supplied in step 3 in the **Password** field.
5. If you do not want to have to supply the user name and password each time you have to connect to the InterChange Server instance in System Manager then enable the **Remember user name and password** checkbox.
6. If you want to open a relationship at this time type the name of the relationship definition in the **Relationship** field.

   If you do not want to open a relationship at this time you can open it after connecting to the server. For more information, see "Opening a relationship" on page 186.
7. Click **Connect**.

   If you connect to InterChange Server in Relationship Manager and specify a relationship to open as described in step 6, then Relationship Manager displays the "retrieve relationships" window, described in "Retrieving relationship instances" on page 187.

   Figure 58 on page 186 shows the "Connect to InterChange Server" dialog.

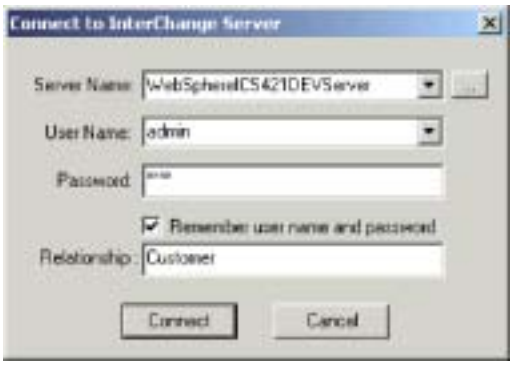

Figure 58. Connecting to InterChange Server

Disconnecting from InterChange Server

To disconnect Relationship Manager from InterChange Server, select **Server > Disconnect** from the menu bar of Relationship Manager.

Working with relationships

Once you have started Relationship Manager and connected it to an InterChange Server, you can use Relationship Manager to work with relationship data as described in the following sections.

Opening a relationship

Do the following to open a relationship definition in Relationship Manager after it is already connected to the server:

1. Select **File > Open** from the menu bar of Relationship Manager.
2. At the "Open Relationship" window, select the name of the relationship you want to open.

   Figure 59 on page 187 shows the "Open Relationship" window.

Figure 59. Opening a relationship

3. Click **OK**.

   When you open a relationship, Relationship Manager displays the “retrieve relationships” window, described in “Retrieving relationship instances.”

Retrieving relationship instances

Select **File > Retrieve Relationship** from the menu bar of Relationship Manager to retrieve relationship instances or return a count of how many instances there are for a relationship. This displays the “retrieve relationships” window shown in

Figure 60 on page 188 shows the “retrieve relationships” window.

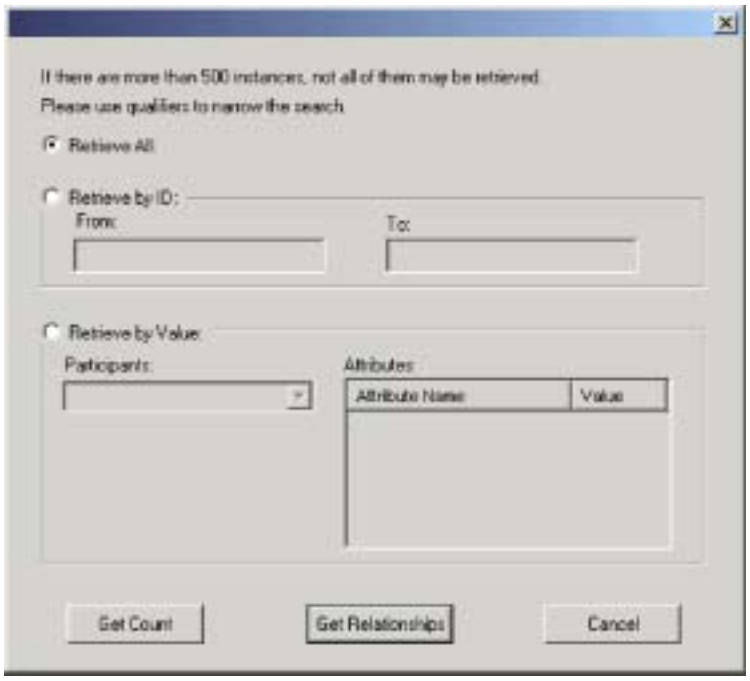

Figure 60. Retrieving relationship instances

The “retrieve relationships” window is also displayed when you open a relationship or connect to InterChange Server and specify a relationship to open.

You can perform the following operations with the “retrieve relationships” window:

- Retrieve the first 500 instances for the relationship, as described in “Retrieving all instances.”
- Retrieve a range of instances for the relationship based on the relationship instance ids, as described in “Retrieving by relationship ID.”
- Retrieve a relationship instance that contains a participant of a particular value you specify, as described in “Retrieving by participant data” on page 189.
- Return a count of the number of instances for the relationship, as described in “Returning a count of the relationship instances” on page 189.

Depending on the number of participants in the relationship definition and the number of participant instances in each relationship instance, these retrieval queries may take some time.

Retrieving all instances

Do the following to retrieve the first 500 instances for a relationship:

1. At the “retrieve relationships” window click **Retrieve All**.
2. Click **Get Relationships**.

   Relationship Manager displays the first 500 instances for the relationship.

Retrieving by relationship ID

Do the following to retrieve a range of up to 500 instances:

1. At the “retrieve relationships” window click **Retrieve by ID**.
2. Type the id of the first instance in the range you want to retrieve in the **From** field.

3. Type the id of the last instance in the range you want to retrieve in the **To** field.
4. Click **Get Relationships**.

   Relationship Manager displays up to 500 instances in a range of the ids you specify.

Retrieving by participant data

Do the following to retrieve a relationship instance based on values for key or nonkey attributes of selected participants:

1. At the "retrieve relationships" window click **Retrieve by Value**.
2. Select the participant whose value you want to search on from the **Participants** drop-down menu.

   For identity relationships, the drop-down menu lists the participant names followed by the business object definition with which the participant is associated.

   For lookup relationships, the drop-down menu lists the participant names followed by the word "Data".
3. Type the one of the types of values listed in Table 17 in the **Value** column in the "Attributes" pane.

Table 17. Supported values for retrieving relationship instances by participant data

Value	Description
Participant data	The data of the selected participant. For example, if the relationship is an identity relationship you would specify the id of the participant instance that you know to find the relationship instance in which it exists. If the relationship is a lookup relationship you would specify the non-key data value of the participant instance.
%	Any string of characters. This option is case-sensitive; numbers are included in the character set. For example, if %A were specified for a participant that stores abbreviated forms of the names of the United States, the values CA, GA, IA, LA, MA, PA, VA, and WA would be returned.
_	Any single character. As an example, _00 would retrieve 100, 200, a00, b00, and so forth.

4. Click **Get Relationships**.

   Relationship Manager displays the first 500 relationship instances that match the specified value.

Returning a count of the relationship instances

To return the number of relationship instances that satisfy a retrieval criteria, select the options for the criteria as described in "Retrieving all instances" on page 188, "Retrieving by relationship ID" on page 188, or "Retrieving by participant data," and then click **Get Count** instead of **Get Relationships**.

Creating relationship instances

Do the following to create a new instance for a relationship:

1. Create the new relationship instance by performing any of the following tasks:
   - Select **File > New** from the menu bar.
   - Use the keyboard shortcut **Ctrl+N**.

- Click **New Relationship** in the toolbar.

Relationship Manager displays the new relationship instance.

Highlighted at the top of the hierarchal relationship tree, on the entry line with the relationship icon is the placeholder for the relationship instance ID, which displays three question marks (???). Once you save the relationship instance or any of its participants, InterChange Server automatically generates the new relationship instance ID and Relationship Manager replaces the question marks with this instance ID.

2. Expand the new relationship instance by clicking on the plus (+) sign next to the ??? placeholder icon.

   The relationship tree displays participant definitions, participant instances, and participant key and non-key attributes beneath the relationship instance in descending order.

3. Do the following to create a new participant instance in the relationship instance:
   a. In the relationship tree, select the participant definition for which you want to create an instance.
   b. Do one of the following to add an instance for the participant:
      - Right-click a participant definition in the listing and choose **Add Participant** from the context menu.
      - Click **Add Participant** in the standard toolbar.
   c. Expand the new participant instance by clicking on the plus (+) sign next to it.
   d. Select the new participant instance.
   e. Click the **Value** column for the participant instance once, then type the desired value into the cell.

      Note: If the **Value** field for the attribute displays three question marks (???), the participant is managed by InterChange Server. You cannot enter values for these participants because InterChange Server automatically generates them when you save the relationship instance. The value is the same value as the relationship instance ID.

At this point, you can perform any of the tasks in Table 18.

Table 18. Tasks for Participant Data

Task	Action
Save the participant instance.	To save the new participant instance, right-click the participant instance and choose **Save Participant** from the context menu. Relationship Manager saves in the appropriate relationship table the data for this participant. The **Modified** column for the participant instances displays the date the participant was saved, which is the create date, in this case. **Note:** Once the participant data has been saved, it cannot be changed. To change its data, the participant must be deleted and another created.
Add more participant instances.	Repeat repeat step 3 in the previous list. **Note:** If you are working with an identity relationship, you cannot create more than one participant instance for a participant definition.

Table 18. Tasks for Participant Data (continued)

Task	Action
Delete a participant.	If necessary, you can delete a saved participant instance by right-clicking on the participant instance and choosing **Delete Participant** from the context menu. Relationship Manager removes the participant instance from the relationship table. If you do not want to remove the participant instance from the database, use the **Deactivate Participant** option (see "Deactivating and activating participants" on page 191). A deactivated participant retains its instance ID and its values.
Save the relationship instance.	Save the relationship instance by performing one of the following tasks: • Select **File > Save** from the menu bar (activated when a relationship instance is selected). • Right-click the relationship instance and choose **Save Relationship** from the context menu. InterChange Server generates the relationship instance ID and Relationship Manager replaces the ??? placeholder with this new ID. Relationship Manager updates the modified date on all saved participant instances to this date. **Note:** At least one participant instance and all key attribute data must be created before the relationship instance can be saved.
Save all relationship instances.	Select **File > Save All** from the menu bar. InterChange Server generates the relationship instance IDs for any relationship instances that do not have one. Relationship Manager replaces any "???" placeholders with the new IDs. Relationship Manager updates the modified date on all saved participant instances to this date.

Deleting relationship instances

To delete a relationship instance from the relationship tables, select the relationship instance you want to delete and perform one of the following actions:

- Select **File > Delete Relationship** from the menu bar.
- Right-click the relationship instance and select **Delete** from the context menu.

The relationship instance and its data are deleted from the relationship tables for the current relationship.

Deactivating and activating participants

A participant instance can be deactivated, or made inactive. Deactivating a participant instance removes it from the relationship instance and prevents it from displaying in the Relationship Manager window, but its record remains in the relationship table so it can be re-activated in the future.

Deactivating a participant

To deactivate a participant instance, right-click the participant instance you want to deactivate and choose **Deactivate Participant** from the context menu. The participant is removed from the Relationship Manager display but not from the relationship tables.

Activating a participant

To activate a participant instance, take the following steps:

1. Select **View > Deactivated Participants** from the menu bar.

   The Deactivated Participants window displays as shown in Figure 61 on page 192.

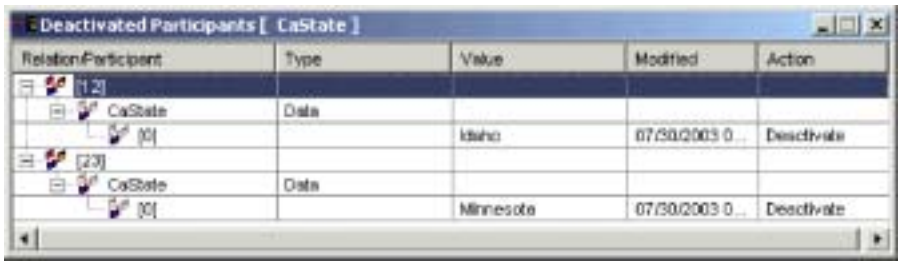

Figure 61. Deactivating participants

2. Select the relationship instance that contains the deactivated participant you want to activate from the list.
3. Expand this relationship instance until the deactivated participant instances display in the list.
4. Right-click the participant instance to reactive and choose **Activate** from the context menu.
5. Select **Edit > Refresh** from the menu bar.

   The activated participant instance displays in its relationship instance in the Relationship Manager window.

   Note: If a participant instance in an identity relationship is deactivated and another participant is added in its place (that is, assigned the same instance ID), the original participant is removed from the Deactivated Participants listing, but remains in the database.

Copying participants

You can create a new participant instance by copying an existing participant instance. To copy a participant instance, take the following steps:

1. In the relationship instance, right-click the participant definition and choose **Add Participant** from the context menu.
2. Right-click the participant instance you want to copy and choose **Copy Participant** from the context menu.
3. Right-click the newly created participant instance and choose **Paste Participant** from the context menu.

Loading and unloading business object files

You can load a business object file of the same type into a participant. To load a business object data file into a participant, take the following steps:

1. Right-click the participant instance where you want to load the business object file and choose **Load Participant with Business Object**.

   The Participant window displays the business object associated with that participant instance, as shown in Figure 62 on page 193.

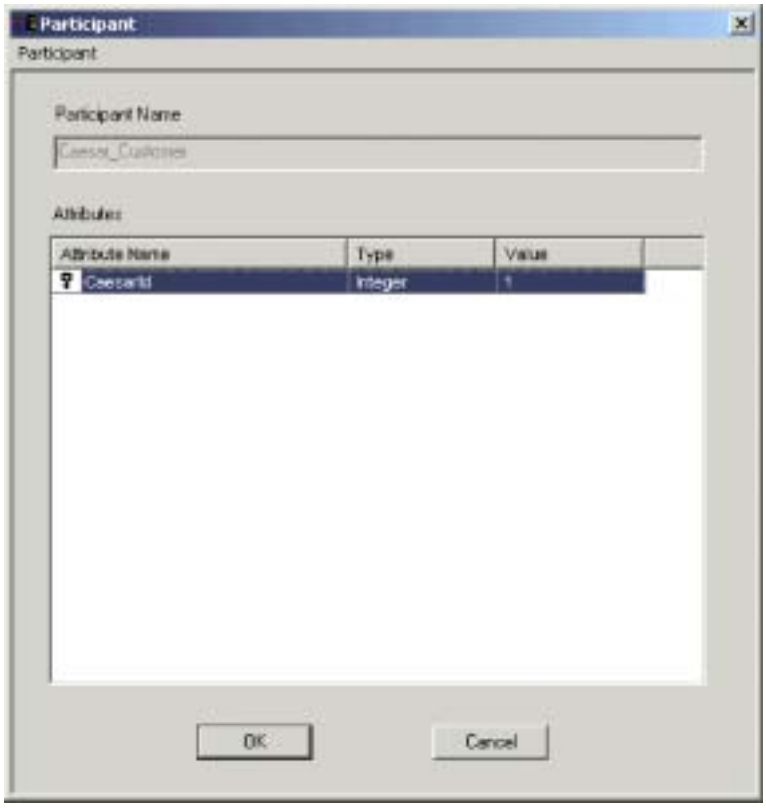

Figure 62. Loading participants with business objects

2. Select **Participant > Load** from the menu bar.
3. Navigate to and open the business object file you want to load.
4. Click **OK**.

Note: Only the first instance of a relationship is loaded if more than one instance exists in the file.

Working with relationship data

An important feature of Relationship Manager is its ability to access and manipulate relationship runtime data contained in the relationship tables. This section describes how to use Relationship Manager to manipulate and access runtime data.

Searching for participants

You can search for participant instances based on different criteria. Depending on how specific your search criteria is, your searches can locate a unique participant instance or a group of participant instances. You can find participant instances in the following ways:

Finding instances by business object

This option searches for instances whose data type is an attribute in a business object.

Do the following to search for instances by business object:

1. Select a participant instance in Relationship Manager.

2. Select **Edit > Find Instances by Business Object** from the menu bar.

   Relationship Manager displays the “Participant” window, as shown in Figure 62 on page 193.
3. Type the participant value by which you want to search in the **Value** cell.
4. Click **OK**.

   Relationship Manager displays any matching instances in a dialog box.
5. Double-click any of the instances in the dialog displayed by Relationship Manager to navigate to and highlight the instance.

Finding instances by data

This option searches for instances whose type is Data.

Do the following to search for instances by data:

1. Select a participant instance in Relationship Manager.
2. Select **Edit > Find Instances by Data** from the menu bar.

   Relationship Manager displays the “Find Instances by Data” window, as shown in Figure 63.

Figure 63. Finding instances by data

3. Type the participant value by which you want to search in the **Enter the data** cell.
4. Click **OK**.

   Relationship Manager displays any matching instances in a dialog box.
5. Double-click any of the instances in the dialog displayed by Relationship Manager to navigate to and highlight the instance.

Filtering the displayed participants

You can filter the participants to only display those created or modified between certain dates. Do the following to filter the displayed participants:

1. Select a participant in Relationship Manager.
2. Select **View > Filter** from the menu bar.

   Relationship Manager displays the “Filter” dialog, as shown in

Figure 64. Filtering participant results

3. In the "Filter" dialog, enter the earliest date of creation or modification for the participant in the **from** field and the latest date of creation or modification for the participant in the **to** field.

   Use the following techniques to enter the date value:

   - Type letters in the text field to cycle through days of the week. For instance, type **S** to cycle through dates that occur on Saturdays and Sundays.
   - Click the small up and down arrows to increment or decrement the date by one day.
   - Click the large down arrow to display a calendar that you can use to select a date.

4. Click the **Include inactive participants as well** checkbox if you want to include inactive participants in the resulting display.
5. Click **OK**.

   Relationship Manager displays the history of activity for the filtered interval in the "Filter Results" dialog. The filtered display includes inactive participants if the **Include inactive participants as well** option box is checked. Figure 65 shows the "Filter Results" dialog.

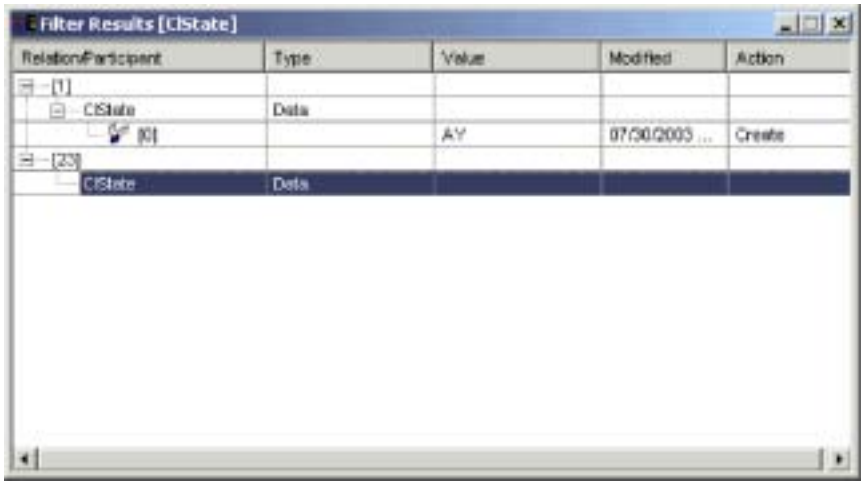

Figure 65. Viewing filtered participant data

Cleaning up participants

Do the following to clean up participants due to inconsistent or corrupt data in the source application or generic object:

1. Select a participant in Relationship Manager.
2. Select **View > Clean Up Participants** from the menu bar.

Relationship Manager displays the “Clean up Participants” dialog, as shown in Figure 66.

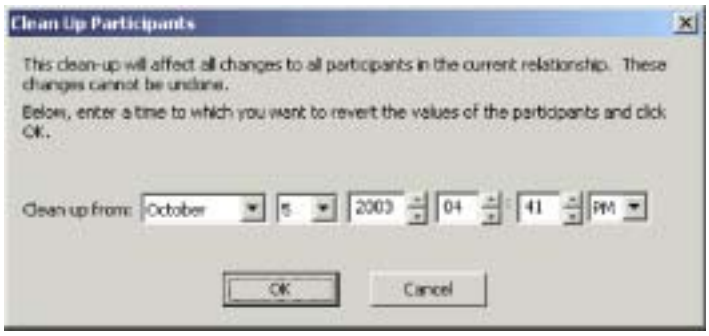

Figure 66. Cleaning up participants

3. In the “Clean Up Participants” dialog box, enter the date to which you want to revert the participant values to in the **Clean up from** field.

   Use the following techniques to enter the date value:

   - Type letters in the text field to cycle through days of the week. For instance, type **S** to cycle through dates that occur on Saturdays and Sundays.
   - Click the small up and down arrows to increment or decrement the date by one day.
   - Click the large down arrow to display a calendar that you can use to select a date.

4. Click **OK**.

   All participant adds, deactivations, and activations since that point in time are erased from the database. A participant that has been deleted or whose value has been modified cannot be cleaned up.

Printing relationship data

Relationship Manager allows you to print information about a relationship’s runtime data. It creates a tree representation of the runtime data, much like the data appears in the tool’s main window. You can manage the printing of this output tree with the following print features:

- **Print**—Send the formatted runtime data to your system printer.
- **Print Preview**—Send the formatted runtime data to a window of Relationship Manager.
- **Page Setup**—Change the layout of the runtime data so that it fits onto a page

Sending relationship data to a printer

The printing ability of Relationship Manager sends the current contents of the relationship tree in the main window to the printer. Do the following to print relationship runtime data:

1. Expand the relationship tree of Relationship Manager so that the information you want to print is displayed.
2. Print relationship runtime data in any of the following ways:
   - Select **File > Print** from the menu bar.
   - Use the keyboard shortcut **Ctrl+P**.
   - Click **Print** in the toolbar.

Using Print Preview with relationship data

Do the following to preview the output of relationship data as it would be printed:

1. Expand the relationship tree of Relationship Manager so that the information you want to print is displayed.
2. Initiate the print preview in any of the following ways:
   - Select **File > Print Preview**.
   - Use the keyboard shortcut **Ctrl+Alt+P**.
   - Click **Print Preview** in the toolbar.

Setting up the page

Relationship Manager tries to print the output tree as it appears in the tool's main window. If the output tree is narrow enough to fit on the page, you do not need to take any special formatting tasks. If, however, the output tree does not fit on the page, you can set formatting options to change the layout of the output tree so that it fits on the page.

You can set up the print format with the "Print Page Setup" feature, which you initiate in any of the following ways:

- Select **File > Page Setup** from the menu bar to display the "Print Page Setup" dialog, where you can configure information such as printer setting, paper size and orientation.
- Use the keyboard shortcut **Shift+Alt+P**.

Any of these commands invokes the "Print Page Setup" dialog.

From the "Print Page Setup" dialog, you can provide the following formatting information:

- Column formatting options appear in the "Column Settings" area of the "Print Page Setup" dialog. Table 19 lists the options contained in this area.

Table 19. Column formatting options

Column formatting option	Description
Column headers on every page	When this option is enabled, the column headers (such as Relation, Type, Value) appear on every page, not just the first one.
Resize columns to fit all data	When this option is enabled, Relationship Manager attempts to enlarge each column to fit any strings wider than the column itself. If the resulting tree does not fit onto the printed page, Relationship Manager tries to eliminate any extra room inside the columns. If this field and its subfields are greyed out, their values indicate the default values that Relationship Manager uses when formatting the output tree. Leave these fields in their greyed-out state if you want to confirm the formatting every time Relationship Manager generates the output tree. If you activate these fields, Relationship Manager uses the options you set for *every* output tree it generates.
Use smaller font if necessary	When this option is enabled, Relationship Manager tries to reduce the font size if data is still larger than the page. It reduces font size in three steps. At each step, it reduces the font size by 20%.

Table 19. Column formatting options (continued)

Column formatting option	Description
Confirm new font size	When this option is enabled, Relationship Manager prompts you to choose how to continue when it has reduced the font size sufficiently to fit the output tree on a page. It provides you with the following choices: • Print with this font • Restore and print with the original font size • Cancel printing In the process of reducing font size, this size can become as small as half the original size. If this option is disabled, printing starts when Relationship Manager finds a font size that fits the output tree on a page.
If altered during print/preview, adjust column widths	When this option is enabled, Relationship Manager saves any changes to the column sizes that it has made to get the output tree to fit on a page. It uses the saved settings for future printings of the output tree.

The **Resize columns to fit data** field and its associated subfields indicate the actions that Relationship Manager takes when it formats the output tree. These fields can appear in one of three states:

- Checkmark not selected (field is empty)—indicates that Relationship Manager does not perform the associated task when it formats the output tree.
- Checkmark selected and greyed out—indicates that Relationship Manager conditionally performs the associated task when it formats the output tree; that is, Relationship Manager displays a confirmation dialog before it takes the associated action.
- Checkmark selected—indicates that Relationship Manager unconditionally performs the associated task when it formats the output tree; that is, Relationship Manager does not display a confirmation dialog before it takes the associated action.

As Table 19 indicates, these fields initially display in a greyed-out state. You can set these fields from the confirmation dialogs that display during the formatting process. For example, if the values in a column do not fit in the current width of this column, Relationship Manager displays a confirmation dialog with the **Yes** and **No** options and a **Make this the default choice** field. If you select the **Make this the default choice** check box, Relationship Manager sets the **Resize columns to fit data** field based on the option you have selected from the confirmation dialog, as follows:

- **Yes**—Relationship Manager deselects the associated check boxes and therefore does not perform the associated action.
- **No**—Relationship Manager selects the associated check box and therefore unconditionally performs the associated action.
- Margin formatting options appear in the Margin Settings area of the “Print Page Setup” dialog. Table 20 lists the options contained in this area.

Table 20. Margin formatting options

Margin formatting option	Description
Use maximum page area	When this option is enabled, Relationship Manager uses the maximum printing area of the page (according to your printer's capabilities).
Use these margins (inches)	When this option is enabled, you can set the left, right, top and bottom margins to specified values.

- Page formatting options appear in the "Page Settings" area of the "Print Page Setup" dialog. Table 21 lists the options contained in this area.

Table 21. Page formatting options

Page formatting option	Description
Print page header on every page	When this option is enabled, Relationship Manager puts the relationship name and print time at the top of every page, not just the first one.
Print page footer	When this option is enabled, Relationship Manager puts page numbers at the bottom of every page.
Synchronize document with print preview	When this option is enabled, Relationship Manager displays in its main window the relationship data that you were last viewing in the print preview. For example, if you had paged through the print preview to the third "page" of instances and then exited from the print preview, Relationship Manager scrolls through the relationship data so that this same "page" of instances appears in its main window. When this option is disabled, Relationship Manager displays in its main window the same data it displayed before you performed the print preview.

- Grid formatting options appear in the "Grid Settings" area of the "Print Page Setup" dialog. Table 22 lists the options contained in this area.

Table 22. Grid formatting options

Grid formatting option	Description
Print grid (uncheck to reset)	When this option is enabled, Relationship Manager includes grid in the output tree to separate rows and columns. When you click this field twice, you include both vertical and horizontal g rid lines in the output tree.
Horizontal grid only	When this option is enabled, Relationship Manager includes only a horizontal grid (to separate rows) in the output tree. If the output tree does *not* include a horizontal grid, each printed page can fit an additional 2-3 rows of data.
Vertical grid only	When this option is enabled, Relationship Manager includes only a vertical grid (to separate columns) in the output tree.

Chapter 11. Using Test Connector

Test Connector simulates the activities of a connector to allow you to test your integration components without the complexity of running an actual connector. This chapter consists of the following sections:

Recommended testing procedure

This is the recommended test procedure for testing components in the WebSphere business integration system:

1. If your integration broker is InterChange Server, consider using the System View view, which can be very helpful in determining if a flow you have sent ends in success or failure.

   For more information, see the *System Administration Guide*.
2. Set up Test Connector to emulate a source connector.
   a. Launch Test Connector as described in "Starting Test Connector" on page 202.
   b. Create a profile for the source connector in the interface as described in "Creating a new profile" on page 204.
   c. Connect Test Connector to the agent to begin emulating the source connector, as described in "Emulating a connector" on page 206.
3. Set up instances of Test Connector to emulate each destination connector involved in the interface.
   a. Launch Test Connector as described in "Starting Test Connector" on page 202.
   b. Create a profile for a destination connector as described in "Creating a new profile" on page 204.
   c. Connect Test Connector to the agent to begin emulating the destination connector as described in "Emulating a connector" on page 206.
   d. Repeat 3athrough 3cabove for all destination connectors involved in the interface.
4. Arrange the instances of Test Connector on your screen so that you can easily identify the connector being emulated in each Test Connector window. For example, in Figure 67 on page 202the source Test Connector is arranged to the left of the destination Test Connector.

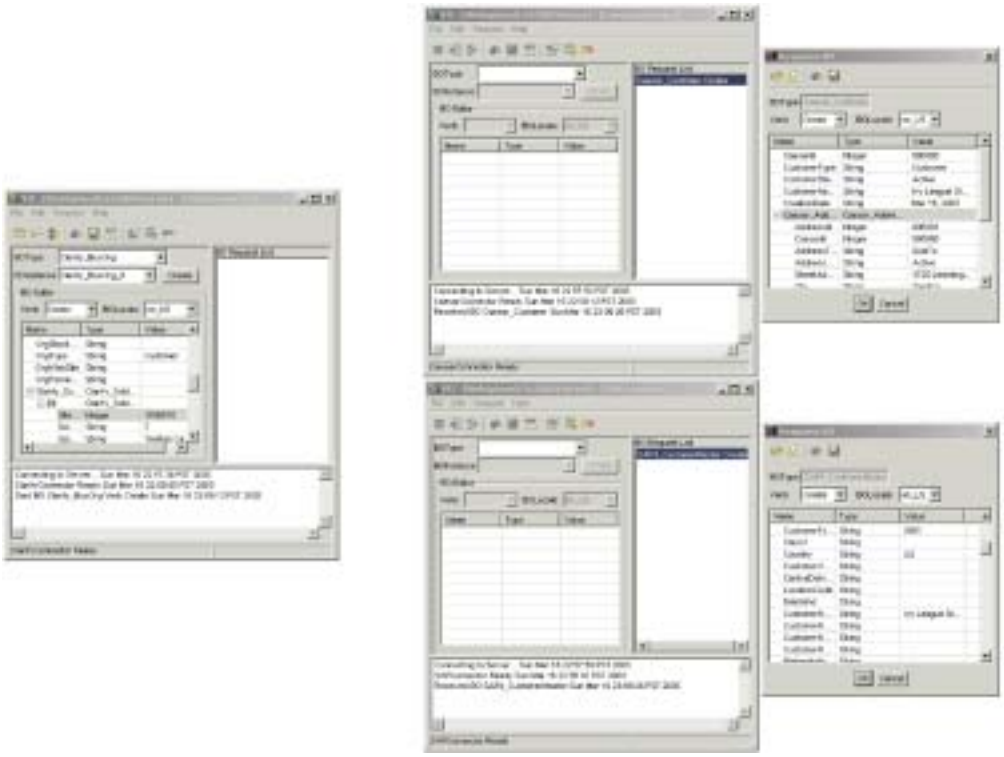

Figure 67. Source and destination instances of Test Connector.

5. Send a request business object from the source connector. From the source Test Connector, do the following:
   a. Create a business object that is managed by the interface you need to test as described in “Creating request business objects” on page 206.
   b. Save the business object to a file to use in subsequent tests as described in “Saving a business object” on page 210.
   c. Send the business object as described in “Sending request business objects” on page 207.
6. Simulate the response to the request business object from the destination connector. From the destination Test Connector window, do the following:
   a. Accept the request business object as described in “Accepting a request business object” on page 211.
   b. Send the business object as a response as described in “Sending a response business object” on page 212.
7. Repeat step 5through step 6as many times as necessary to test each interface.

Starting Test Connector

To start Test Connector, do one of the following depending on your integration broker:

- If your integration broker is InterChange Server, select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Development > Test Connector**.
- If your integration broker is WebSphere Application Server or a WebSphere message broker (WebSphere MQ Integrator, WebSphere MQ IntegratorBroker, or WebSphere Business Integration Message Broker) select **Start > Programs > IBM WebSphere Business Integration Adapters > Tools > Test Connector**.

Figure 68 shows Test Connector after starting.

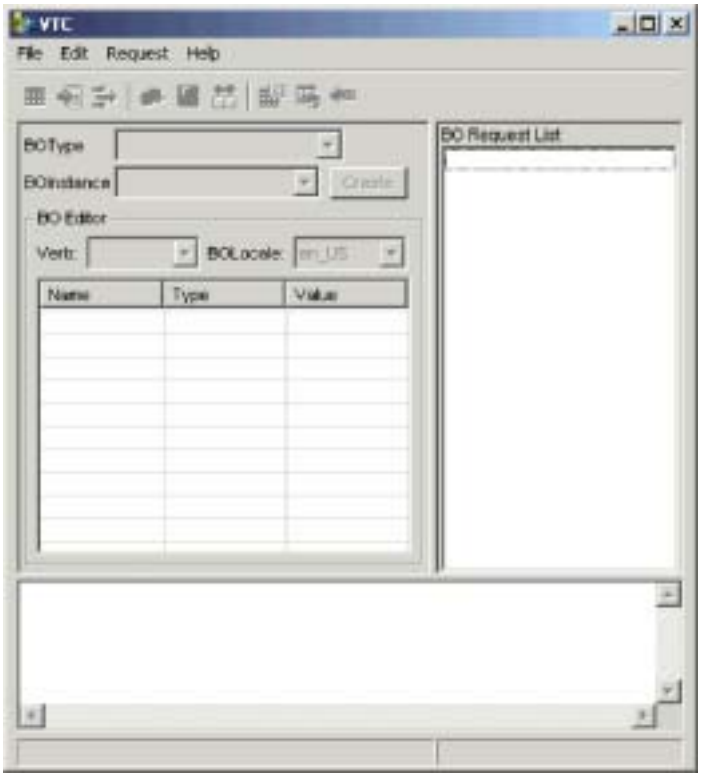

Figure 68. Test Connector

The Test Connector window includes the following panes:

- The “Supported Business Objects” pane in which you can create business object instances to send
- The “BO Request List” pane, which displays any business object requests that the connector has received
- The “Output” pane, which displays messages about Test Connector’s operations, such as when a business object has been sent.

Shutting down Test Connector

To shut down Test Connector and cause it to stop emulating a connector agent, select **File > Exit** from the menu bar. When presented with the “Shutdown” prompt, click **Yes**.

Creating and editing connector profiles

Test Connector uses profiles to store the information it needs to emulate a connector. You must create a profile for each connector you want to emulate. You can edit and delete existing profiles.

Saving the connector definition to a file

To emulate a connector using Test Connector, you must save the connector definition to a file. Do the following to save a connector definition to a file:

1. Open the connector definition in Connector Configurator.
2. Select **File > Save As > To File** from the menu bar.

3. Navigate to the directory in which you want the file saved, type a name in the **File name** field, ensure that the value `Configuration (*.cfg)` is displayed in the Save as type drop-down menu, and click **Save**.

   Connector Configurator saves the connector definition to a file with the specified name.

Creating a new profile

You must create a profile for any connector you want to emulate in Test Connector. The profile specifies information such as the name of the connector, the configuration file to be used, and the type of integration broker with which the connector communicates. To create a new connector profile, do the following:

1. Select **File > Create/Select Profile** from the menu bar to display the "Connector Profile" window.
2. In the "Connector Profile window", select **File > New Profile** from the menu bar.
3. In the "New Profile" window, click **Browse** and then navigate to the configuration file for the connector you preparing in "Saving the connector definition to a file" on page 203.
4. Type the name of the connector in the **Connector Name** field. You must type the exact name of the connector definition as it exists in the integration broker repository. For the adapter for JText, for instance, you must type `JTextConnector`, without any spaces between the words `JText` and `Connector`, and with each letter being the proper case.
5. Select the proper integration broker in the **Broker Type** drop-down menu—`ICS`, `WMQI` or `WAS`.

   Note: Select `WMQI` if your broker is any WebSphere message broker.
6. If you selected `ICS` as your broker type in step 5,do the following as well:
   a. Type the name of the InterChange Server instance in the **Server** field.

      Be sure to type the name precisely; it is case-sensitive and Test Connector will not be able to communicate with InterChange Server if the name is not correct.

   b. Type the password for the `admin` user account in the **Password** field. The default password is `null`.

      Figure 69 shows the "New Profile" window:

Figure 69. Creating a new connector profile

7. Click **OK** to close the "New Profile" window.

   The "Connector Profile" window displays the name of the connector in the **Connector** column, the name of the InterChange Server instance in the **Server** column (if the integration broker is ICS), and the path and name of the connector configuration file in the **Configuration File** column.

   Figure 70 shows the "Connector Profile" window with a profile for the `ClarifyConnector` configured to communicate with an InterChange Server instance, and a profile for the `JTextConnector` configured to communicate with a WMQIB server.

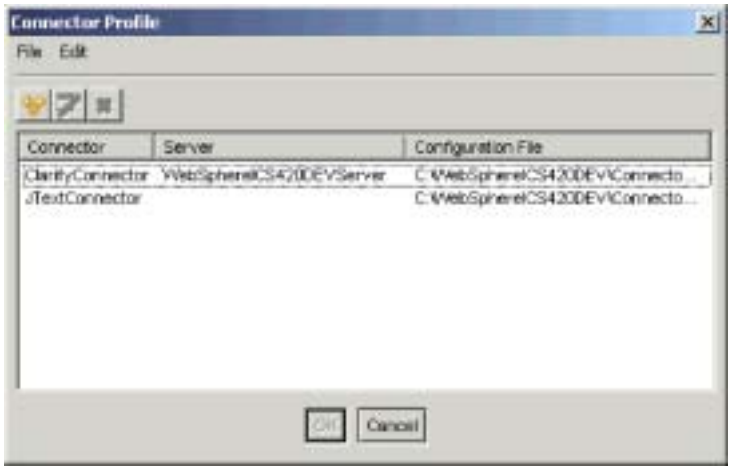

Figure 70. The "Connector Profile" window

8. Click **OK** to close the "Connector Profile" window.

Editing a profile

Follow the steps below to make changes to an existing connector profile:

1. Select **File > Create/Select Profile** from the menu bar of Test Connector or use the keyboard shortcut **Ctrl+N** to display the Connector Profile window.
2. In the "Connector Profile" window select the profile you want to edit and then select **Edit > Edit Profile** from the menu bar.
3. Type new values in the fields of the "New Profile" window and use the **Browse** button to change the configuration file as necessary to make your edits.
4. Click **OK** to close the "New Profile" window.

Deleting a profile

Do the following to delete a connector profile:

1. Select **File > Create/Select Profile** from the menu bar of Test Connector or use the keyboard shortcut **Ctrl+N** to display the "Connector Profile" window.
2. In the "Connector Profile" window, select the profile you want to delete and then select **Edit > Delete Profile** from the menu bar.

Emulating a connector

After creating a profile for a connector, you may use that profile to connect Test Connector to the agent. Once you connect Test Connector to the agent, Test Connector begins emulating the connector defined in the selected profile.

To connect Test Connector to the agent, do the following:

1. Select **File > Create/Select Profile** from the menu bar of Test Connector.
2. In the "Connector Profile" window, select the name of the connector whose profile you want to open.
3. Click **OK**.
4. Select **File > Connect** from the menu bar.

   Test Connector displays messages in the "Output" pane as it attempts to emulate the connector. When it finishes connecting, it displays a message indicating that it is "ready" in the "Output" pane and populates the **BOType** list in the "Supported Business Objects" pane.

Working with business objects

To test whether a business process interface has been developed correctly, you need to verify that business objects can be successfully exchanged and processed. This section describes how to:

- Create, modify, delete, and save business object test data
- Compare the attribute values of business objects to easily and quickly view changes made during processing
- Send and receive business objects

Working with request business objects

Request business objects are those that you send from Test Connector when it is emulating a connector that is the source of the events that trigger an interface. Working with request business objects consists of creating a business object instance, populating it with data, and sending the request.

Creating request business objects

To create a new business object in Test Connector, do the following:

1. In the "Supported Business Objects" pane, select the name of the business object you want to create from the **BOType** drop-down menu.
2. Click **Create** next to the **BOInstance** field.
3. When presented with the "New Instance" dialog, type a name for the instance in the **Enter Name** field.
4. Select the desired verb from the **Verb** drop-down menu.
5. Select the desired locale from the the **BOLocale** drop-down menu.
6. Provide values for the simple attributes and child business objects within the top-level object, as described in "Setting values for business object attributes" on page 209.

   Figure 71 on page 207 shows a business object named `Caesar_Customer` with the `Create` verb, the `en_US` locale, values specified for each of its simple attributes, and a single instance of the `Caesar_Address` child business object.

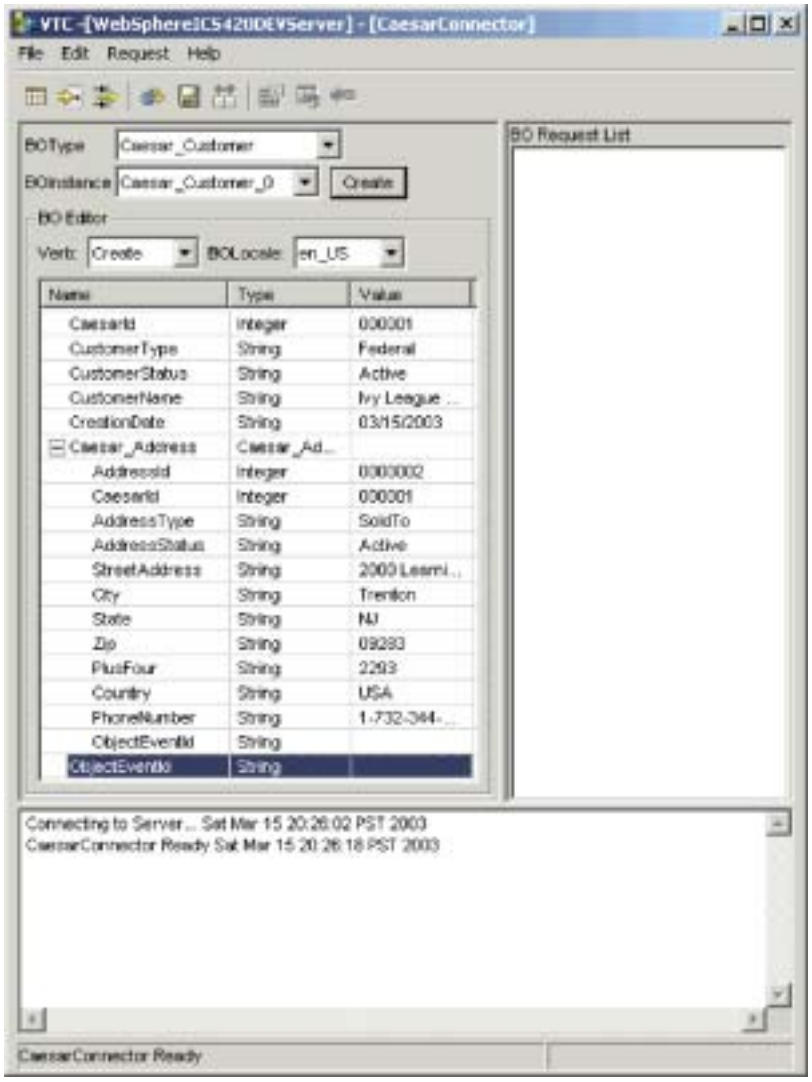

Figure 71. Populating a business object with data

7. Click **OK**.

Sending request business objects

Once you have created or loaded a business object and specified values for its attributes, you have several ways to send the business object as a request to the integration broker.

Sending request business objects asynchronously: When a source connector sends a request business object in asynchronous mode, it does not expect to get back a response business object. Once the request business object is dispatched, the source connector's role in the transaction is finished. The response business object is typically processed by the integration broker. The default mode for Test Connector is asynchronous.

To send a business object asynchronously, do the following:

1. Select **Request > Mode >Asynchronous** from the menu bar.

   Note: Test Connector operates in "Asynchronous" mode by default, so you only have to perform this step if you previously were sending

synchronous requests from the connector. Furthermore, you do not have to set the mode before sending each request.

2. Select **Request > Send** from the menu bar.

If the broker specified in the connector definition is InterChange Server then the business object request is sent to the server for processing.

If the broker specified in the connector definition is one of the supported message brokers or WebSphere Application Server then the business object is placed on the queue specified in the `RequestQueue` standard property.

Sending request business objects synchronously: When a source connector sends a request business object synchronously, it expects to get back a response business object from the integration broker after any destination applications have processed the request. In synchronous mode, Test Connector puts the response business object on the queue specified by the source connector's Synchronous Request Queue property. The default mode for Test Connector is asynchronous.

1. Set Test Connector to synchronous mode by selecting **Request > Mode > Synchronous** from the menu bar.
2. Select **Request > Send** from the menu bar.
3. If the broker specified in the connector definition is InterChange Server then the "Select Collaboration" dialog is displayed. Select the collaboration to which the business object should be sent from the **Collaboration** drop-down menu and click **OK**.

If the broker specified in the connector definition is InterChange Server then the business object request is sent to the configured port of the collaboration object chosen for processing.

If the broker specified in the connector definition is one of the supported message brokers or WebSphere Application Server then the business object is placed on the queue specified in the `SynchronousRequestQueue` standard property.

Sending request business objects in batch mode: In batch mode, Test Connector lets you specify the number of instances of a particular business object you want to send, as well as one attribute in the top-level object —a primary key attribute, for example—that you want set to a unique value for each instance. Test Connector copies the business object as many times as you have specified, incrementing the value of the single attribute you specified, and sends each business object. This option allows you to create a large number of business objects quickly and easily.

If the selected attribute is a key field that participates in dynamic cross-referencing as part of an identity relationship, then you must guarantee that the initial value and all those that follow it are unique. Otherwise, the cross-referencing logic will fail, causing the request business objects to fail.

To ensure that the values are unique, you can use Relationship Manager or execute SQL statements against the table for the relationship participant as follows.

- Determine the highest current value for the participant and set the Initial Value field to an even higher value. The first business object instance in the batch and all those that follow will then be unique.
- Delete the existing table entries for the participant, thus guaranteeing that no entries have the same attribute value as any of the batch business objects.

To send business objects in batch mode, do the following:

1. Select the name of the business object you would like to send from the **BOType** drop-down menu.
2. Select **Request > Send Batch** from the menu bar.
3. In the "Batch Mode" window, select the desired verb from the **Verb** drop-down menu.
4. Select the desired locale from the the **BOLocale** drop-down menu.
5. Select from the **Attribute** list the attribute in the top-level business object that you want incremented with each business object request in the batch.

   The selected attribute should typically be an attribute that uniquely identifies the business object, such as a primary key.
6. In the **Initial Value** field, type the starting value for the attribute to be incremented.
7. In the **No. of BO's** field, type the number of business object instances you want generated and sent.
8. Click **OK**.

   Test Connector generates the number of business objects you specified, all identical with the exception of the one specified attribute, whose value is incremented for each instance.

   If the broker specified in the connector definition is InterChange Server then the business object request is sent to the server for processing.

   If the broker specified in the connector definition is one of the supported message brokers or WebSphere Application Server then the business object is placed on the queue specified in the `RequestQueue` standard property.

Figure 72 shows a batch mode configuration in which:

- Fifty business objects are to be sent.
- The value of the attribute `OrgObjid` is to be incremented.
- The starting value for the attribute is 100001.

Figure 72. The Batch Mode Window

Setting values for business object attributes

The following sections describe the various ways you can set the values of simple and compound attributes in a business object instance:

- "Setting values for simple attributes" on page 210
- "Adding child business objects" on page 210
- "Removing child business objects" on page 210
- "Setting the verb of a child business object" on page 210

Setting values for simple attributes

To provide a value for a simple attribute, click its cell in the **Value** column and enter a value.

Adding child business objects

To add an instance of a child business object, right-click the attribute that represents the child object and select **Add Instance** from the context menu.

A plus sign (+) is added next to the attribute that represents child business object to show that there is at least one child business object instance. If you expand the child object attribute, numbered entries are displayed for each instance. The individual instances also have plus signs (+) next to them, so you can expand them and set values for their attributes.

To add more child business object instances, right-click the attribute that represents the child object and select **Add Instance** from the context menu.

Note: If the **Card** property of the attribute that references the child business object is set to the value 1 (indicating it is of single-cardinality), then you will only be able to add one instance of the child object.

Removing child business objects

To remove an instance of a child business object, right-click the instance and select **Remove Instance** from the context menu.

To remove all instances of a child business object, right-click the attribute that represents the child business object and select **Delete All Instances** from the context menu.

Setting the verb of a child business object

You can set the verb of a child business object to test the effect that value has on the business process. This can be helpful when you are troubleshooting logic that involves the cross-referencing of child objects.

To set the verb of a child business object instance, right-click it and choose **Set Verb** from the context menu. When presented with the "Select Verb" prompt, selected the desired verb and click **OK**.

Using the Response BO toolbar

You can edit the attributes of a business object received by a destination connector before you send it as a response. The toolbar of the "Response BO" dialog that you use when doing so has several toolbar buttons that can be used to set the values of the business object. For more information, see "Editing response business objects" on page 212.

Saving a business object

You can save a business object in Test Connector so that it can be used for later tests, shared with technical support (to help troubleshoot problems), or used as response data. You can save any business object, including ones that you have created and ones that appear as requests in the Test Connector window of a destination connector. By default, business objects are saved to a file with a business object extension (`.bo`).

It is recommended that you create a directory or directory structure specifically for test data files, with subdirectories dedicated to each interface or to each connector, as appropriate. This organization makes the necessary files are easy to locate and

makes testing more efficient. Furthermore, it is recommended that you give the test data file for a business object the same name as the business object definition itself.

Saving request business objects

Do the following to save a business object instance that you have created as a request:

1. Select the business object you want to save.
2. From the menu bar, select **Edit > Save BO**.
3. Navigate to the desired directory and specify a name for the file in the **File name** field.
4. Click **Save**.

Saving response business object

Do the following to save a business object instance that has been received by a destination instance of Test Connector and will be sent as a response:

1. Select the business object instance in the "BO Request List" pane.
2. Select **Request > Edit Response** from the menu bar.
3. Click **Save BO**.
4. Navigate to the desired directory and specify a name for the file in the **File name** field.
5. Click **Save**.

Loading a business object

To load a business object that has been saved to a file, do the following:

1. Select **Edit > Load BO** from the menu bar of Test Connector.
2. Navigate to the business object test data file and open it.
3. When presented with the "New Instance" dialog, type a name for the instance in the **Enter Name** field.
4. Click **OK**.

Deleting a business object

To delete a business object from Test Connector, select **Edit > Delete BO** from the menu bar.

Note: This action only removes the business object from the Test Connector. It does not remove the connector's support for the business object definition.

Accepting a request business object

When you send a business object as a request, the business object appears in the "BO Request List" pane of any Test Connector instances that are emulating destination connectors in the interface, provided that the transaction did not fail.

After you have accepted the request business object, you can edit it if necessary as described in "Editing response business objects" on page 212.

Working with response business objects

Response business objects are those that you send from Test Connector when it is emulating a connector that is the recipient of business object requests in an interface. Working with request business objects consists of editing the values in the business object instance and sending the response back to the broker.

Editing response business objects

When you receive a business object request in a destination instance of Test Connector, you commonly want to edit the values of the attributes. For instance, you will want to provide unique values for primary key attributes that participate in relationships, or you will want to modify the value of other attributes to test map or collaboration logic that responds differently depending on the exact values in the business object. Do the following to set the values of business object attributes :

1. Select the business object instance in the "BO Request List" pane.
2. Select **Request > Edit Response** from the menu bar.
3. Do the following to edit the attributes of the business object:
   - Use one of the techniques described in "Setting values for business object attributes" on page 209to modify the values of the business object attributes.
   - Click **Reset BO to default** to set the values of the business object attributes to their default values as specified in the business object definition.
   - Click **Clear BO values** to clear the values of all the attributes in the business object.
   - Click **Load BO** to populate the attributes of the business object with test data from a file.

     The ability to load saved data into a business object request is very useful in situations where you have to populate a response business object with data before sending it as a reply. Instead of manually typing a value for each attribute that requires response data, you can type the values once, save the business object (as described in "Saving a business object" on page 210), and then load the saved data on subsequent tests.

Sending a response business object

After you accept a request business object, edit the business object, if needed, and send it back as a reply.

Table 23 lists Test Connector's reply options and shows their corresponding connector return codes for both C++ and Java connectors. For more detailed information about C++ or Java Connector return codes, see the *Connector Development Guide for Java or C++*.

Table 23. Test Connector reply types and connector return codes.

Test Connector reply type	C++ connector return code	Java connector return code
Success	BON_SUCCESS	SUCCESS
Fail	BON_FAIL	FAIL
Multiple Hits	BON_MULTIPLE_HITS	MULTIPLE_HITS
Retrieve By Content Fail	BON_FAIL_RETRIEVE_BY_CONTENT	RETRIEVEBYCONTENT_FAILED
Not Found	BON_BO_DOES_NOT_EXIST	BO_DOES_NOT_EXIST
Value Duplicate	BON_VALDUPES	VALDUPES

To reply to a request business object, do the following:

1. Select the business object in the "BO Request List" pane.
2. From the menu bar, select **Request > Reply**.
3. Select an item from the **Reply** submenu.

You will typically want to use the “Success” response type to ensure that the business process behaves as designed. If you edit the response business object before replying, as described in “Editing response business objects” on page 212, then the response object is sent through the map configured for it to be returned to the collaboration. If you do not edit the response business object before replying then a status message is returned to the collaboration object. The response object is not mapped in this case.

You will want to use the “Fail” response type to test the reaction of the interface to an unsuccessful flow.

Comparing business object instances

Test Connector can compare two business objects of the same type and display the attributes that differ in value. You can use this function to view changes to a business object at different points in the execution of a transaction (for instance, you could compare a business object that has been sent to the integration broker with the same business object after the integration broker has updated it). To compare two business objects, do the following:

1. Create a request business object instance by following the instructions in either “Creating request business objects” on page 206or “Loading a business object” on page 211.
2. Select the response business object instance in the “BO Request List” pane that you would like to compare the request business object instance to.
3. From the menu bar, select **Edit > Compare BO’s**.

   Test Connector opens the “Compare Business Objects” window with a table that displays the attributes which have different values in the two business objects. Figure 73 on page 213shows a comparison between two business object instances.

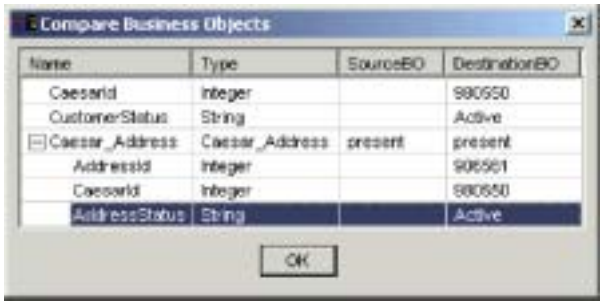

Figure 73. The Business Object Comparison Window.

4. Click **OK** to close the window.

Chapter 12. Using Integrated Test Environment

Integrated Test Environment is a workbench perspective that enables you to conveniently test an entire business process interface. As described in “Interfaces” on page 1, an interface centers around a collaboration object, so a test performed in Integrated Test Environment centers around a collaboration object as well.

To perform a test in Integrated Test Environment, you must do a number of tasks such as register ICS as a test server, create a test unit, deploy the components in the interface to the server, start the server, emulate the connectors in the interface, and exchange business objects between the connectors. The following characteristics describe the use of Integrated Test Environment:

- You only have to perform some of the tasks a single time. For instance, you only have to create a test unit for an interface once.
- You have to perform some tasks multiple times. For instance, you might test how an interface responds when you change the value in a particular attribute, so you will have to send business object requests for the interface multiple times.
- You can perform some tasks in multiple ways. For instance, you can deploy components to the server before you prepare the test unit, or you can deploy all of the components for a test unit by using the Task Manager view, or you can deploy single components by using the Test Unit view.

Most of the sections in this chapter describe tasks you must perform as part of the greater task of performing a test, or describe particular elements of the Integrated Test Environment interface.

The section “Performing a test using Integrated Test Environment” on page 269 describes the workflow you will typically follow to perform a single test of a single interface. It is a list of references to the sections in this chapter that document the specific sub-tasks and interface elements, and it provides a task-oriented structure for using Integrated Test Environment, whereas the other sections are more interface-oriented. In situations where you might perform a task in one of several ways it references the way that is most effective and efficient in most situations. To use this chapter as best as possible, it is recommended that you follow the workflow described in “Performing a test using Integrated Test Environment” on page 269, but substitute particular sub-tasks from other sections where the recommended approach is not appropriate for your environment or situation.

Registering InterChange Server as a test server

To test an interface using Integrated Test Environment, you must register the InterChange Server instance so that it starts in test mode. This allows Integrated Test Environment to do things such as the following:

- Locate the batch file that starts InterChange Server, so that it can be started from within the Integrated Test Environment interface.
- Re-direct the InterChange Server logging and tracing information to a view within Integrated Test Environment, rather than to the console.

To register InterChange Server in test mode, see “Registering an InterChange Server instance” on page 38 and perform each of the steps necessary to register the

server instance as a local test server. Integrated Test Environment can only work with a server instance that is installed on the same machine on which the workbench is installed. You cannot, for instance, use Integrated Test Environment to work with a server instance that is installed on a Unix computer.

Starting Integrated Test Environment

To start Integrated Test Environment, do the following:

1. Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Administrative > System Manager**.
2. Select **Window > Open Perspective > Other** from the menu bar.
3. Select `Integrated Test Environment` from the list of perspectives and then click **OK**.

   The workbench starts and appears. Figure 2 on page 35 shows the Integrated Test Environment perspective and “Integrated Test Environment interface” describes the interface and its elements.

Integrated Test Environment interface

The Integrated Test Environment perspective has several views and one editor in the default configuration it opens with. Figure 2 on page 35 shows the default Integrated Test Environment perspective.

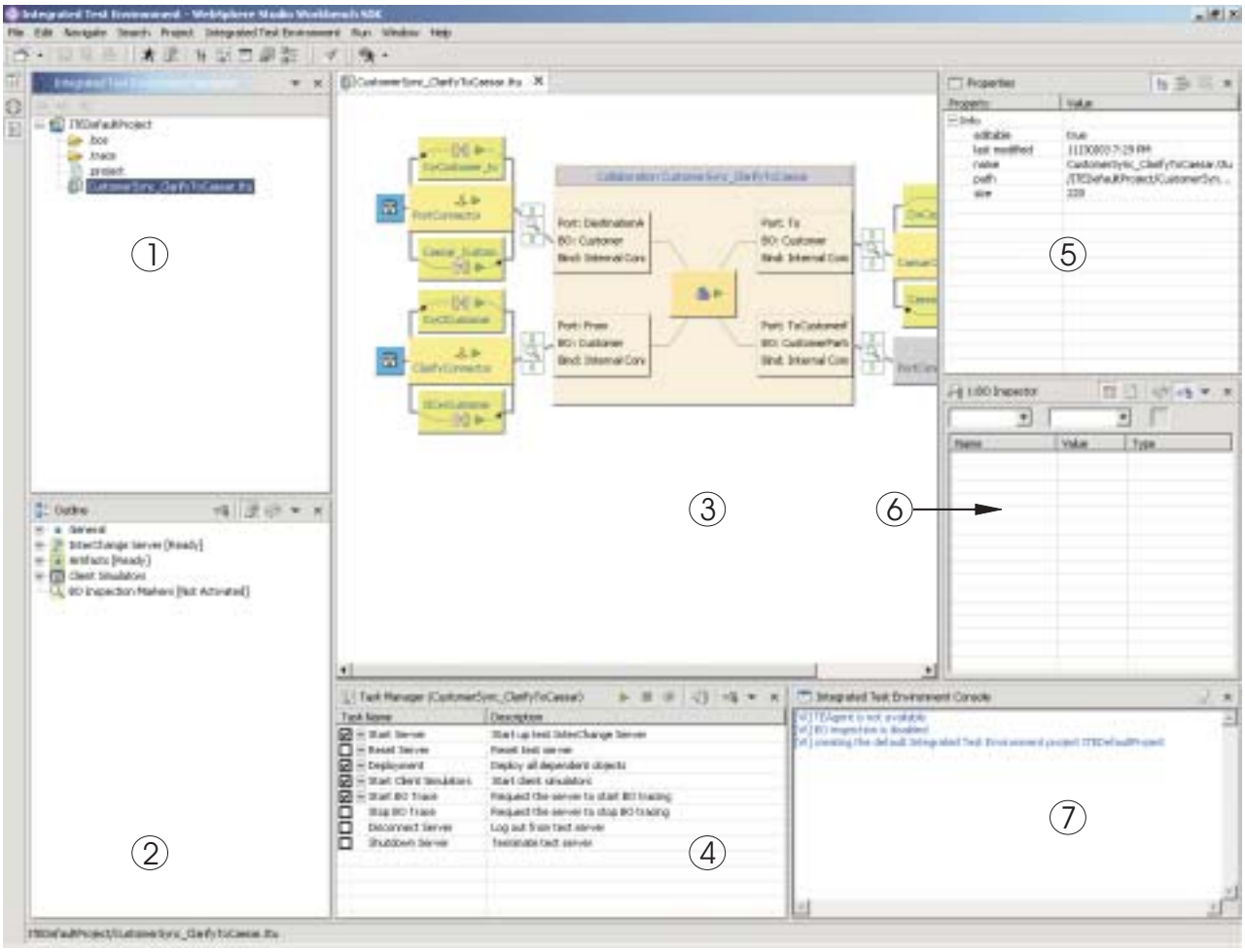

Figure 74. Integrated Test Environment perspective

Table 24 describes the interface elements of the Integrated Test Environment perspective, identified by the numbers in Figure 74:

Table 24. Integrated Test Environment perspective interface elements

Interface element number	Interface element name
1	“Integrated Test Environment Navigator view”
2	“Outline view” on page 218
3	“Test Unit editor” on page 218
4	“Task Manager view” on page 218
5	“Properties view” on page 218
6	“BO Inspector view” on page 218
7	“Integrated Test Environment Console view” on page 218

Integrated Test Environment Navigator view

This view lists the projects and test units you create. For more information on working with test projects and test units, see “Working with test projects and units” on page 224.

Outline view

This view lists the components and elements that comprise the interface and are displayed in the "Test Unit editor." As you select items in this view, their properties are displayed in the "Properties view." This view also allows you to determine if the interface dependencies have been resolved, and if other required conditions have been satisfied prior to testing.

This is a default workbench view.

Test Unit editor

This editor provides a graphical representation of the interface. It allows you to do things such as the following:

- Deploy components, as described in "Managing the repository using the Test Unit view" on page 245.
- Manage the states of components, as described in "Managing component states using the Test Unit view" on page 245.
- View business object data, as described in "Using the BO Inspector view" on page 262.

Task Manager view

The Task Manager view allows you execute tasks that must be performed as part of a test, such as starting InterChange Server and starting Client Simulator views. For more information on this view, see "Using the Task Manager view" on page 234.

Properties view

You can select items in the ITE Navigator, Outline, and Test Unit views to display properties of the items in this view. The property information is largely useless to you and you will probably want to overlay this view with a Client Simulator view or BO Inspector view.

For more information on the Client Simulator view, see "Using the Client Simulator view" on page 248.

For more information on the BO Inspector view, see "Using the BO Inspector view" on page 262.

This is a default workbench view.

BO Inspector view

This view allows you to view business object data. For more information, see "Using the BO Inspector view" on page 262.

Integrated Test Environment Console view

The Integrated Test Environment Console displays information about the testing tasks. For instance, entries are written to it when you start the InterChange Server instance and start Client Simulator views. Use the information in it to troubleshoot the testing process itself.

If you start InterChange Server from within Integrated Test Environment, then a second tab named InterChange Server Console is added to this view. The

InterChange Server Console displays the logging and tracing information of InterChange Server. Use the information displayed in it to troubleshoot the interface you are testing.

This is a default workbench view.

Selecting a server configuration

If you have registered multiple InterChange Server instances as local test servers then you must select the server configuration you want to work with in Integrated Test Environment. Do the following to select a server configuration:

Note: The InterChange Server must be registered in the System Manager perspective before performing these steps. For more information, see "Registering an InterChange Server instance" on page 38.

1. Do one of the following to display the InterChange Server Configuration window:
   - Select **Integrated Test Environment > Test Server Configuration** from the menu bar of Integrated Test Environment.
   - Expand the "InterChange Server" node in the Outline view, then right-click the "Registration" node and choose **Show test server configuration** from the context menu
2. At the InterChange Server Configuration window, select the test server you want to use from the pane at the left.

   When you select a server, the configuration information for it is presented in the pane on the right-hand side of the window. The read-only text field at the top of the window indicates whether the server is running, and if Integrated Test Environment is connected to it.

   Figure 75 shows the InterChange Server Configuration window.

Figure 75. Test Server Configuration window

3. Click **OK**.

Note: Although you can register several InterChange Server instances as test servers, they must all be installed on the computer on which Integrated Test Environment is running. You cannot use Integrated Test Environment to test interfaces running on a server that is installed on another computer than the one on which Integrated Test Environment is installed.

Configuring RMI settings

Integrated Test Environment uses a software agent that communicates with InterChange Server to perform tests. This agent must communicate with ICS and the tools through a port using Remote Method Invocation (RMI). Integrated Test Environment also requires a block of nine contiguous ports over which to communicate with each connector that is emulated in the interface being tested.

You must configure your environment so that there are ports available for these communication requirements. The default RMI port is 1099, and this is the port that the Integrated Test Environment agent is configured to use by default. Integrated Test Environment is configured by default to use ports 1100 and up to communicate with the connectors being emulated in the interface.

Other programs installed on your system may be configured to use port 1099, however, so you must determine which ports you can use. After determining which ports you can use you must configure settings in Integrated Test Environment to use the desired port as well as set a property in the `start_server.bat` batch file.

Determining which ports to use

Before attempting to use Integrated Test Environment, you should use the techniques described in the following sections to determine if the ports it is configured to use by default are available or not.

Using the netstat command

You can execute the `netstat` command at a command line interface in Windows to generate a list of the ports that are currently in use. It is recommended that you use the `-a` option to list all of the ports in use, rather than just those used for TCP/IP. You might also want to use the `-n` option, which orders the ports numerically.

The following example shows the use of the netstat command and a portion of the returned output:

```
C:\>netstat -a -n
Active Connections
Proto  Local Address          Foreign Address        State
TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
TCP    0.0.0.0:371            0.0.0.0:0              LISTENING
TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
TCP    0.0.0.0:1029           0.0.0.0:0              LISTENING
TCP    0.0.0.0:1030           0.0.0.0:0              LISTENING
```

The first entry shows that port 80 is being used and the last line shows that port 1030 is being used. Examine the lines to determine if ports 1099 through 1108 are being used. It is recommended that you copy the output into a text file where you can search for the port numbers.

If ports 1099 through 1108 are in use already then you must change the settings as described in “Setting the RMI port in start_server.bat” on page 222 and “Setting the RMI port in start_server.bat” on page 222.

Checking the services file

Besides using the `netstat` command to determine which ports are currently in use, you can also check the `services` file to determine which ports are reserved. Do the following:

1. Use a text editor to open the file named `services` in the `Winnt\System32\drivers\etc` directory in your file system.
2. Search the entries in the file to determine if there are any services listed that are associated with ports 1099 through 1108.

   If you find any such entries, you might experience complications when trying to use Integrated Test Environment with the default configuration. Change the settings as described in “Setting the RMI port in start_server.bat” on page 222 and “Setting the RMI port in start_server.bat” on page 222 if so.

Setting the RMI port in the Integrated Test Environment preferences

If you follow the instructions in “Determining which ports to use” on page 220 and determine that the ports used by Integrated Test Environment by default are already in use then you must set the preferences for Integrated Test Environment to use ports that are available.

Integrated Test Environment requires a single port for the test environment agent to communicate with the server. It also requires nine ports to communicate with up to nine connector agents that can be emulated during a test. You configure the port number that Integrated Test Environment will use for the first of the nine possible connector agents, and then it uses the eight ports above that number for the others, so you must ensure that there is a block of nine contiguous ports available. If you do not have nine connectors that require emulation in the interface then you can use a smaller block of ports. Ideally, you should find a block of ten available ports so that all of the ports related to the Integrated Test Environment are contiguous.

Do the following to configure the port usage for Integrated Test Environment:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Expand **Integrated Test Environment**.
3. Select **Test Server**.

   Figure 76 on page 222 shows the Test Server preferences.

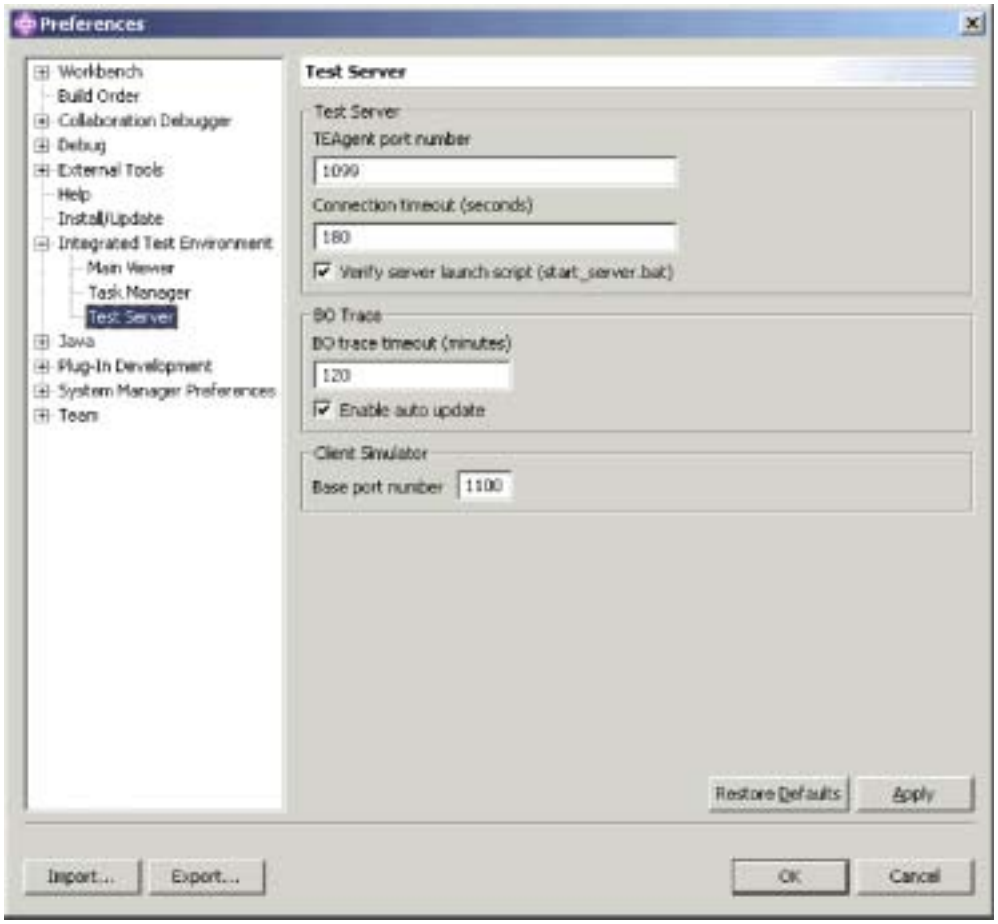

Figure 76. Test Server preferences

4. In the “Test Server” pane, enter the port number for the test environment agent in the **TEAgent port number** field.

   The default value is 1099.

5. In the “Client Simulator” pane, enter the port number that Integrated Test Environment should use for the first of the nine possible connector agents or access clients in the **Base port number** field.

   The default value is 1100. Integrated Test Environment uses the value in the **Base port number** field for the first connector agent that must be emulated, and use the subsequent port numbers for other connector agents.

6. Follow the instructions in “Enabling Integrated Test Environment to create a custom batch file” on page 224 to have Integrated Test Environment use a custom batch file to which it adds the necessary port configuration information specified in the previous steps.

7. Click **OK**.

Setting the RMI port in start_server.bat

If you do not configure the Integrated Test Environment preferences to use a custom batch file to start the server then you must add the RMI port configuration information to the `start_server.bat` batch file yourself.

Even if you plan to use the default RMI port number of 1099, you must add the property to the batch file and set it equal to the default value. Do the following to do so:

1. Use a text editor to open the batch file named start_server.bat in the bin directory of the product installation.
2. Add the **-DTEAgent** Java system property to the line in the batch file where the Java program is executed to start the server, and set it equal to the value in the **TEAgent port number** field in the Integrated Test Environment preferences. The following example shows how the batch file would appear if the **-DTEAgent** property were set to the default RMI port number of 1099:

   ```
   %CWJAVA% -Djava.ext.dirs=%JRE_EXT_DIRS%;"%MQ_LIB%";"%DB2_LIB%"
   -Duser.home="%CROSSWORLDS%" -mx%CW_MEM_HEAP%m -DTEAgent=1099
   -DCW_MEMORY_MAX=%CW_MEM_HEAP% %ORB_PROPERTY% -classpath %JCLASSES%
   ServerWrapper -s%SERVERNAME% %2 %3
   ```

   Note: You might want to make the modifications described in "Configuring InterChange Server to start in design mode" at this point, as those modifications also involve the start_server.bat batch file.
3. Save and close the file.

Configuring InterChange Server to start in design mode

If you plan to deploy the components you will test through Integrated Test Environment then you must modify the start_server.bat batch file so that the InterChange Server instance starts in design mode. This is because Integrated Test Environment may deploy the components in such an order that dependencies are not immediately resolved. For the deployment to succeed in such a situation, the server must run in design mode.

As described in "InterChange Server modes" on page 41, you typically start InterChange Server in design mode by passing the -design option at the command line or in the **Target** field of the shortcut. When you use Integrated Test Environment, you start the server through the Integrated Test Environment perspective, and it does not use the shortcut, so even if you modify your InterChange Server shortcut to use the -design option the server will not start in design mode.

To start the server in design mode when using Integrated Test Environment, you can either add the -design option to the start_server.bat batch file, or have Integrated Test Environment use a temporary batch file to which it adds the option itself. Follow the instructions in one of the following sections depending on your preference:

Manually editing the start_server.bat file

Do the following to edit the start_server.bat batch file yourself and add the -design option to it:

1. Use a text editor to open the batch file named start_server.bat in the bin directory of the product installation.
2. Add the **-design** option to the line in the batch file where the Java program is executed to start the server. You should add the **-design** option after the **-s%SERVERNAME%** option. The following example shows how the batch file should appear:

```
%CWJAVA% -Djava.ext.dirs=%JRE_EXT_DIRS%;"%MQ_LIB%";"%DB2_LIB%"
-Duser.home="%CROSSWORLDS%" -mx%CW_MEM_HEAP%m -DTEAgent=1200
-DCW_MEMORY_MAX=%CW_MEM_HEAP% %ORB_PROPERTY% -classpath %JCLASSES%
ServerWrapper -s%SERVERNAME% -design %2 %3
```

3. Save and close the file.

Enabling Integrated Test Environment to create a custom batch file

You can configure Integrated Test Environment to start InterChange Server in design mode automatically. With this approach, Integrated Test Environment reads in the `start_server.bat` file and determines if the `-design` option has been specified properly or not. It then creates a file named `start_server_ite_tmp.bat` as a copy of the `start_server.bat` file and adds the `-design` option if it has not been properly specified. If you start the server from within Integrated Test Environment then it uses this temporary file.

Do the following to configure the Integrated Test Environment preferences to create a temporary batch file:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Expand **Integrated Test Environment**.
3. Select **Test Server**.

   Figure 76 on page 222 shows the Test Server preferences.
4. Enable the **Verify server launch script (start_server.bat)** checkbox to have Integrated Test Environment use a custom batch file to which it adds the necessary configuration information.

   If you do not plan to use the custom batch file that Integrated Test Environment can create, leave the **Verify server launch script (start_server.bat)** checkbox disabled and be sure to follow the instructions in “Setting the RMI port in start_server.bat” on page 222 to add the configuration information yourself.
5. Click **OK**.

Important: For this approach to work, you must start the server from Integrated Test Environment. If you start the server using the program group shortcut or the command line then this approach will not work, because those startup techniques do not use the temporary startup script.

Working with test projects and units

All resources in an Eclipse-based platform are contained in projects, as described in “Projects” on page 32.. You must create integration component library projects to contain integration components, you must create user projects to contain shortcuts to components, and you must create **test projects** in the Integrated Test Environment to contain definitions for tests.

A test unit is a resource created in the workbench that defines a test. Integrated Test Environment is designed to let you test an entire interface, and an interface typically centers around a collaboration object, as described in “Interfaces” on page 1, so a test unit centers around a collaboration object as well. You can define all the test units you want to run in a single test project, or can create multiple test projects and add test units to them as appropriate to organize them according to your preference.

Configuring test project preferences

Before creating either test projects or test units you should configure the preferences related to both. Do the following to configure your preferences related to test projects and units:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Select **Integrated Test Environment**.

   Figure 77 shows the Integrated Test Environment preferences.

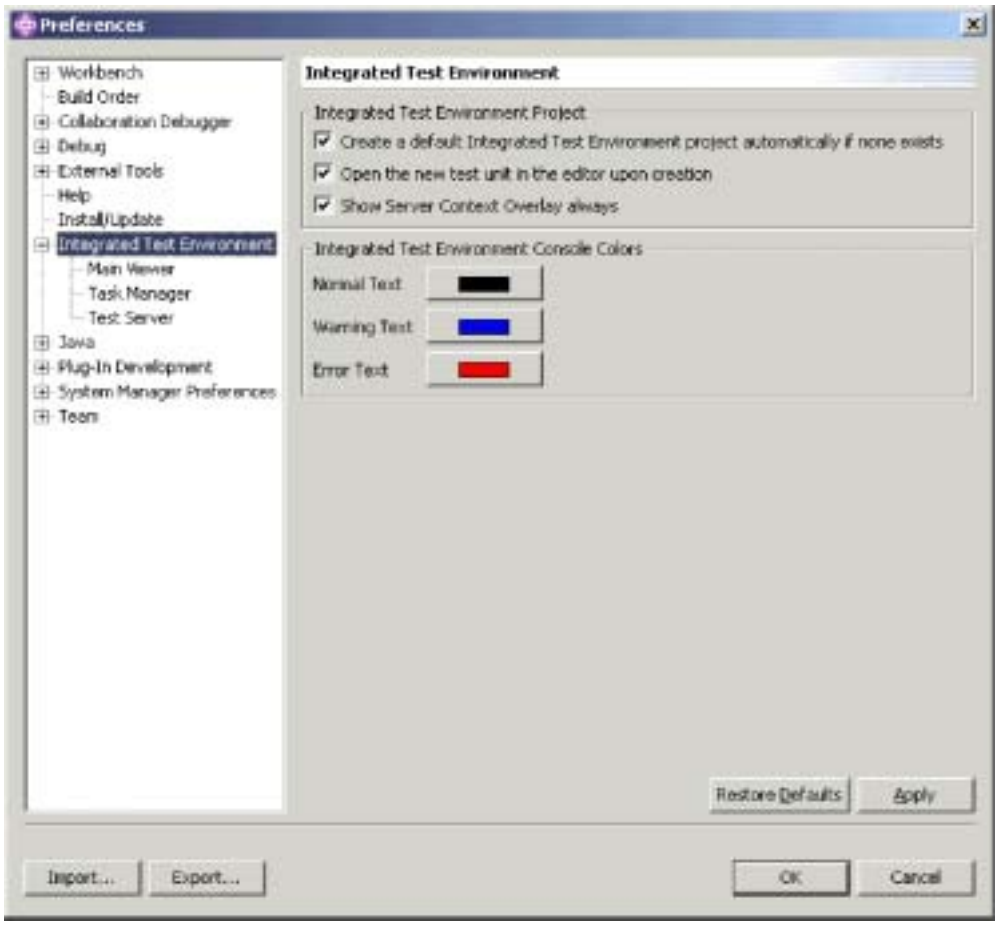

Figure 77. Integrated Test Environment preferences

3. Enable the **Create a default Integrated Test Environment project automatically if none exists** checkbox if you want Integrated Test Environment to create a default project named `DefaultITEProject` if you are creating a test unit and no project already exists.
4. Enable the **Open the new test unit in the editor upon creation** checkbox if you want to automatically open a test unit definition in the Test Unit Editor after you create it.

   For more information about opening test units if you decide not to enable this option, see “Opening a test unit” on page 228.
5. Enable the **Show Server Context Overlay always** checkbox if you want to always have the Server Context Overlay enabled.

   For more information about the Server Context Overlay, see “Using the Server Context Overlay” on page 243.
6. Click **OK**.

Creating a test project

Do the following to create a test project to store the individual test units you will create:

1. Select **File > New > Integrated Test Environment Project** from the menu bar.
2. At the "New Integrated Test Environment Project" screen, type a name for the test project in the **Project name** field.

   Project names can only contain alphanumeric characters and underscores, and must be specified in English.
3. To have the folder for the library created in the default location (your workspace) and with a name identical to the name specified for the library, leave the **Use default location** checkbox enabled.

   If you want to specify the name and location of the library folder, do the following:

   a. Clear the **Use default location** checkbox.

   b. Type the full path and name of the directory that you want to use for the library in the **Location** field, or click **Browse** to select an existing directory.

   Note: There is no way to create the folder for a library in the path of the workspace other than to let System Manager do it by use of the **Use default location** checkbox.

   Figure 78 shows the "New Integrated Test Environment Project" wizard.

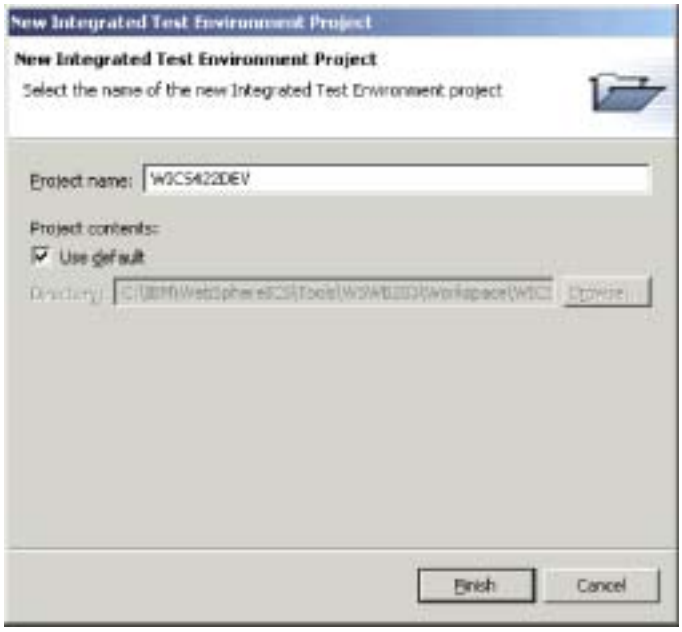

Figure 78. Creating a new WBI ITE project

4. Click **Finish**.

   The project is created and a folder is added for it to the "Integrated Test Environment Navigator" view.

Creating test units

A test unit is the workbench resource that contains the configuration information for a test you want to perform. You can either create a test unit from within Integrated Test Environment, or from within System Manager.

If you have the **Open the new test unit in the editor upon creation** checkbox enabled in the Integrated Test Environment preferences, then the test unit opens after you create it. If you do not have this preference enabled then you must open the test unit using the instructions in "Opening a test unit" on page 228. For more information on Integrated Test Environment preferences, see "Configuring test project preferences" on page 225.

Creating a test unit within Integrated Test Environment

Do the following to create a test unit within Integrated Test Environment:

1. Select **File > New > Integrated Test Environment Test Unit** from the menu bar.
2. At the "Select Collaboration" screen, select the collaboration object you want to test from the list of all the collaboration objects in all the integration component libraries defined in the system.

   Figure 79 shows the "Select Collaboration" screen.

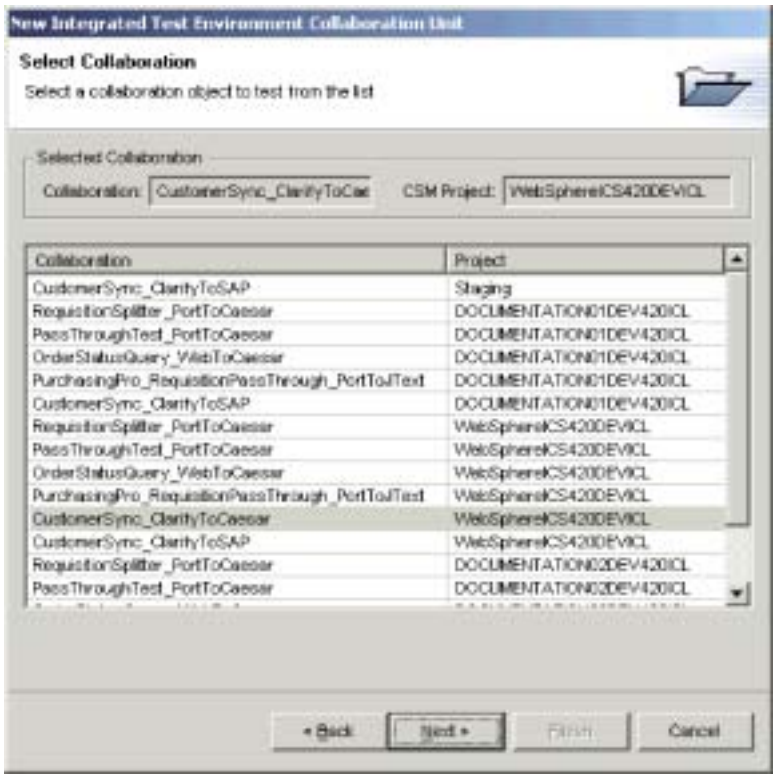

Figure 79. Selecting a collaboration to test

3. Click **Next**.
4. At the "Create Integrated Test Environment Test Unit" screen, do the following:
   a. Type a name for the test unit in the **Test Unit** field.
   b. Select the test project in which the test unit should be created from the **ITE Project** drop-down menu.

   Figure 80 on page 228 shows the "Create Integrated Test Environment Test Unit" screen.

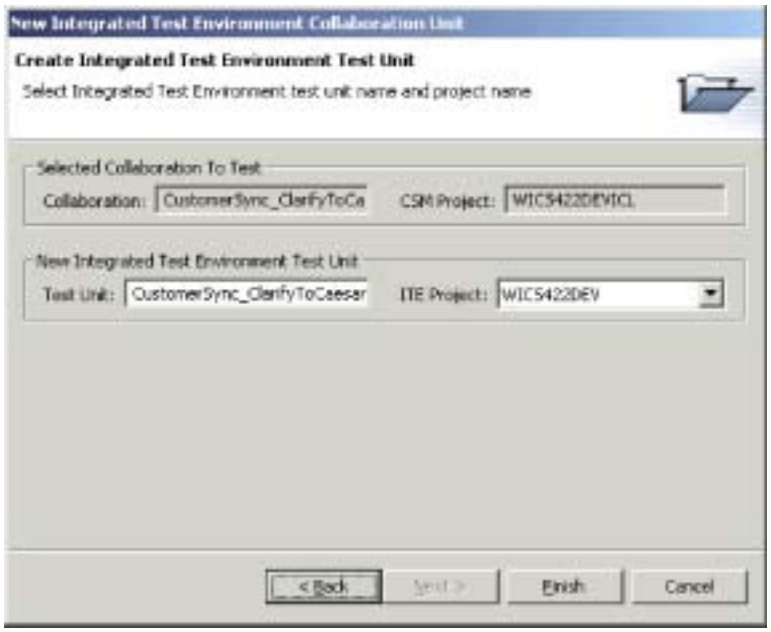

Figure 80. Specifying the test unit name and project

5. Click **Finish**.

Creating a test unit within System Manager

Do the following to create a test unit from within System Manager:

1. Create a test project to contain the test unit as described in "Creating a test project" on page 226.
2. Right-click the collaboration object that represents the interface and select **Debug in Integrated Test Environment** from the context menu.
3. At the "Create Integrated Test Environment Test Unit" screen, type a name for the test unit in the **Test Unit** field, and select the test project it should be created in from the **ITE Project** drop-down menu.

   Figure 80 shows the "Create Integrated Test Environment Test Unit" screen.
4. Click **Finish**.

Opening a test unit

Once you have created a test unit you can open it to view its layout and configure it. Do the following to open a test unit:

1. Expand the test project that contains the test unit in the "Integrated Test Environment Navigator" view.
2. Do one of the following to open a test unit:
   - Right-click the test unit you want to work with and choose **Open** from the context menu.
   - Double-click the test unit you want to work with.

   The test unit opens, displaying a representation of the collaboration object in the Test Unit editor and populating the views of the perspective with information specific to the interface. Figure 74 on page 217 shows the appearance of the Integrated Test Environment perspective after you have opened a test unit.

Using the Outline view

The Outline view allows you to do a number of things described in the following sections:

- "Working with dependents using the Outline view"
- "Managing the repository using the Outline view" on page 231
- "Managing component states using the Outline view" on page 231
- "Executing tasks using the Outline view" on page 231
- "Verifying test readiness using the Outline view" on page 232
- "Showing and closing Client Simulator views using the Outline view" on page 233
- "Showing and closing BO Inspector views using the Outline view" on page 233
- "Refreshing BO Markers using the Outline view" on page 234

Figure 81 shows the Outline view.

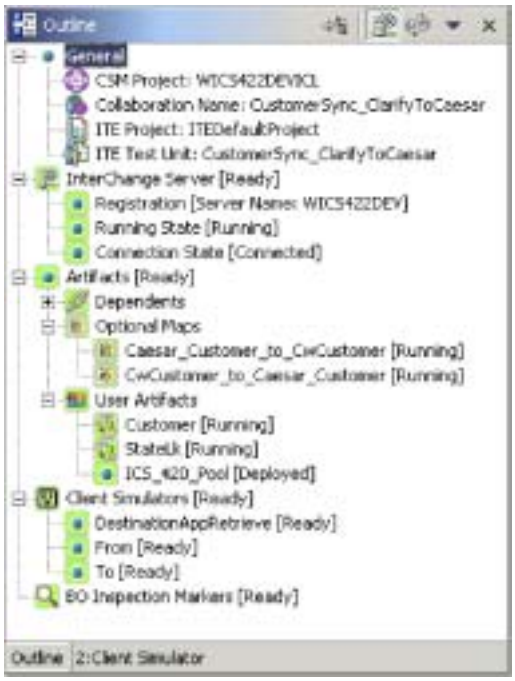

Figure 81. Outline view showing User Artifacts

Working with dependents using the Outline view

As described in "Dependencies and references" on page 82, IBM WebSphere InterChange Server components depend on other components to function properly. To test an interface, you must make sure that all of its dependencies are resolved.

The Outline view in Integrated Test Environment lists the component dependencies that must be resolved to test the interface. The system is able to determine many dependencies among components, and lists them under the **Dependents** node beneath the **Artifacts** node in the Outline view.

"Dependencies that cannot be detected by the system" on page 84 describes the situations in which some components cannot be automatically detected by the system. If you use Integrated Test Environment views to deploy components when

testing an interface then you must make sure that even the dependencies that cannot be detected by the system are resolved. To do this, you can specify those undetectable dependencies yourself. After you do so, the components are listed under the **User Artifacts** node beneath the **Artifacts** node in the Outline view.

Adding user dependents

Do one of the following to add a component to a test unit as a user dependent:

1. In the Outline view, expand the **Artifacts** node.
2. Right-click the **User Artifacts** node and select **Add User Artifacts** from the context menu.
3. At the **User Dependents** window select the components you want to add. You can use standard multiple-selection techniques, such as holding down **Shift** to select contiguous rows and holding down **Ctrl** to select non-contiguous rows. Figure 82 shows the "User Dependents" dialog:

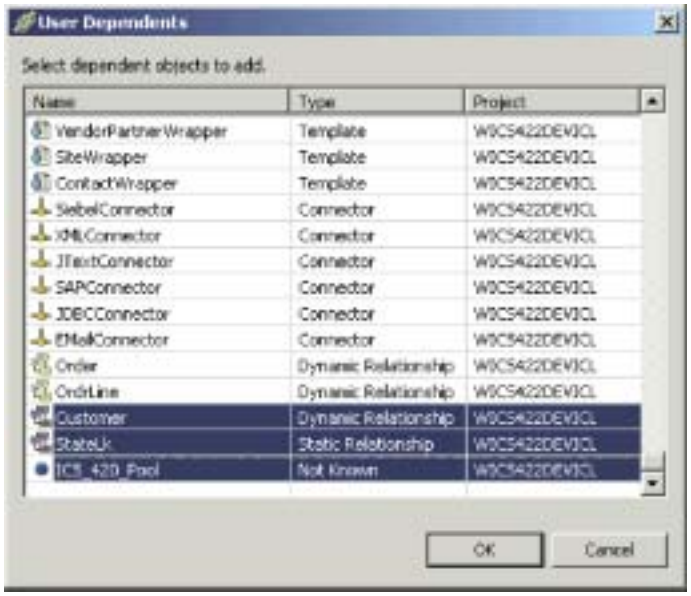

Figure 82. Adding user dependents

4. Click **OK**.

Note: Only components that are not already registered as dependents are displayed in the list.

Deleting user dependents

Expand the **Artifacts** node in the Outline view and do one of the following to remove components from the list of user artifacts:

- Right-click the **User Artifacts** node and select **Remove All User Artifacts** to remove all user artifacts from the list.
- Expand the **User Artifacts** node, then select the user artifacts you want to remove, right-click one of the selected artifacts, and choose Remove from the context menu.

  You can use standard Windows selection techniques to select multiple user projects for deployment at the same time, such as the following:

  – Hold down **Shift** to select contiguous items.

  – Hold down **Ctrl** to select non-contiguous items.

Storing user dependents

You can store the user dependents that you add to a test unit so that you do not have to add them each time. Do one of the following to store the user dependents:

1. Expand the **Artifacts** node in the Outline view.
2. Right-click the **User Artifacts** node and select **Save User Dependents** from the context menu.

Managing the repository using the Outline view

All of the components required for an interface must be deployed to the server repository to test the interface. The Outline view provides a way to work with the InterChange Server repository to ensure that it contains the required components.

When you deploy components, you can either use the **Deploy** option, which deploys the component, or the **Deploy with overwrite** option, which will deploy the component even if it already exists in the server repository. If the selected components already exist in the server repository and you use the **Deploy** option then the deployment will fail.

Do one of the following and choose either **Deploy** or **Deploy with overwrite** from the context menu to deploy the desired components:

- Right-click the “Artifacts” node to deploy all the components
- Right-click the “Dependents”, “Optional Maps”, or “User Artifacts” nodes to deploy all of the components that are listed under those nodes
- Right-click an individual component

To remove a component from the server repository using the Outline view, right-click the component in the view and choose **Remove from server** from the context menu.

Note: You can use standard Windows selection techniques to affect multiple components at the same time, such as the following:

- Hold down **Shift** to select contiguous items
- Hold down **Ctrl** to select non-contiguous items

Entries are written to the Integrated Test Environment Console and InterChange Server Console indicating whether the deployment was successful or not.

Managing component states using the Outline view

You can change the state of a component within Integrated Test Environment. This allows you to resolve problems you discover during testing—you can deactivate the component, modify it, re-deploy it, and re-activate it without using System Manager, repos_copy, or System Monitor.

To change the state of a component, right-click it in the Outline view and choose the desired state operation—**Start**, **Pause**, or **Stop**. Not all component types have all state operations available.

For information on component states, see the *System Administration Guide*.

Executing tasks using the Outline view

You can execute some of the tasks that are part of performing a test using the Outline view. For more information on tasks and task groups, see Table 25 on page 235.

To execute all of the tasks in the “Start Server” task group, right-click either the “InterChange Server” or “Running State” nodes in the Outline view and choose **Start Server** from the context menu.

To execute the “Shutdown Server” task, right-click either the “InterChange Server” or “Running State” nodes in the Outline view and choose **Stop Server** from the context menu.

To execute the “Connect To Server” task, right-click the “Connection State” node in the Outline view and choose **Connect To Server** from the context menu.

To execute the “Disconnect Server” task, right-click the “Connection State” node in the Outline view and choose **Disconnect Server** from the context menu.

To execute all of the tasks in the “Start Client Simulators” task group, right-click the “Client Simulators” node in the Outline view and choose **Start client simulators** from the context menu.

To execute all of the tasks in the “Start BO Trace” task group, right-click the “BO Inspection Markers” node in the Outline view and choose **Start BO Trace** from the context menu.

To execute the “Stop BO Trace” task, right-click the “BO Inspection Markers” node in the Outline view and choose **Stop BO Trace** from the context menu.

Verifying test readiness using the Outline view

The Outline view gives a visual indicator of the state of several conditions that must be satisfied to perform a test. This allows you to quickly confirm if you are ready to perform tests in Integrated Test Environment without having to search for indicators in log text. Each condition that must be satisfied has a green background if it has been satisfied and has a red background if it has not been satisfied. Each condition also has a phrase that indicates its status. The following are the different nodes whose conditions must be satisfied, along with descriptions of the requirements and the phrases displayed when the conditions have been satisfied:

- The “InterChange Server” node must be in a “Ready” state to test an interface. This node represents the state of the server and Integrated Test Environment’s connection to it; it encompasses the following nodes:
  - The “Registration” node lists the InterChange Server instance that Integrated Test Environment has been configured to communicate with.

    For more information about specifying the InterChange Server instance to use, see “Selecting a server configuration” on page 219.
  - The “Running State” node indicates if the InterChange Server instance is running or not; it must be in the “Ready” state to perform a test.
  - The “Connection State” node indicates if Integrated Test Environment is connected to the InterChange Server instance; it must be in the “Connected” state to perform a test.
- The “Artifacts” node must be in a “Ready” state to test an interface. For this node to be in a “Ready” state, all of the components in each of the nodes beneath it must be deployed to the server and running.

  Components that have states, such as connectors and collaboration objects, must be in a “Running” state. Components that do not have states but that are required for the interface, such as business object definitions and collaboration templates, must be in the “Deployed” state.

- The “Client Simulators” node must be in a “Ready” state to test an interface. For this node to be in a “Ready” state, all of the ports listed beneath it must have Client Simulator views associated with them and connected to the server.
- The “BO Inspection Markers” node must be in a “Ready” state for business object tracing to work while testing an interface. For this node to be in a “Ready” state, the “Start BO Trace” task must be started successfully.

Showing and closing Client Simulator views using the Outline view

You use Client Simulator views to emulate connectors and access clients when testing an interface. Client Simulator views provide interfaces to create business object instances with test data, send them as business object requests, and reply to them as business object responses.

To load the profile for a connector into a particular Client Simulator view, right-click the desired port under the “Client Simulators” node and select **Use Client Simulator *x***, where ***x*** is the number of the Client Simulator view position you want to use for that connector.

Table 29 on page 249 shows the default interface positions in which the different Client Simulator views are displayed.

Views in Integrated Test Environment can be concealed by other views during different stages of a test; for example, a Client Simulator view might be concealed by a BO Inspector view. To reveal a Client Simulator view that is concealed, right-click the desired port under the “Client Simulators” node and select **Show Client Simulator** to give focus to the Client Simulator view.

To close a Client Simulator view, right-click the desired port under the “Client Simulators” node and select **Close Client Simulator**.

Note: This option not only closes the Client Simulator view, but also causes the Client Simulator view to stop emulating the connector.

For more information on working with Client Simulator views, see “Using the Client Simulator view” on page 248.

Showing and closing BO Inspector views using the Outline view

If you use business object tracing, Integrated Test Environment records business object data at certain points during the execution of the interface.

To show BO Inspector views for all the business object markers in the interface, right-click the “BO Inspection Markers” node in the Outline view and select **Show All BO Viewers** from the context menu. BO Marker icons are identified in Figure 85 on page 244.

To close all the displayed BO Inspector views in the interface, right-click the “BO Inspection Markers” node in the Outline view and select **Close All BO Viewers** from the context menu.

Table 31 on page 263 shows the default interface positions in which the different BO Inspector views are displayed.

For more information on using business object tracing, see Table 25 on page 235.

For more information on working with BO Inspector views, see “Using the BO Inspector view” on page 262.

Refreshing BO Markers using the Outline view

If you are using business object tracing but have disabled auto-update of the trace information then you must explicitly refresh the BO Markers to update them.

To update BO Markers, right-click the “BO Inspection Markers” node in the Outline view and select **Refresh BO Markers** from the context menu.

Using the Task Manager view

The Task Manager view lets you automate many of the tasks you have to perform when testing an interface.

Several tasks must commonly be executed at certain stages when performing a test. For instance, you must start InterChange Server and connect the Integrated Test Environment perspective to it when preparing to perform a test. Later on in the test you must launch Client Simulator views for the connectors and access clients in the interface, and connect them to the InterChange Server instance. To make it easier to execute all of the tasks that tend to relate to a particular stage during the running of a test, Task Manager displays related tasks in groups. The tasks for launching Client Simulator views and connecting them to the InterChange Server instance, for example, are collected in a task group named “Start Client Simulators”. Task Manager allows you to execute either a single task, or a task group as a whole.

Figure 83 shows the Task Manager view with several tasks and task groups selected.

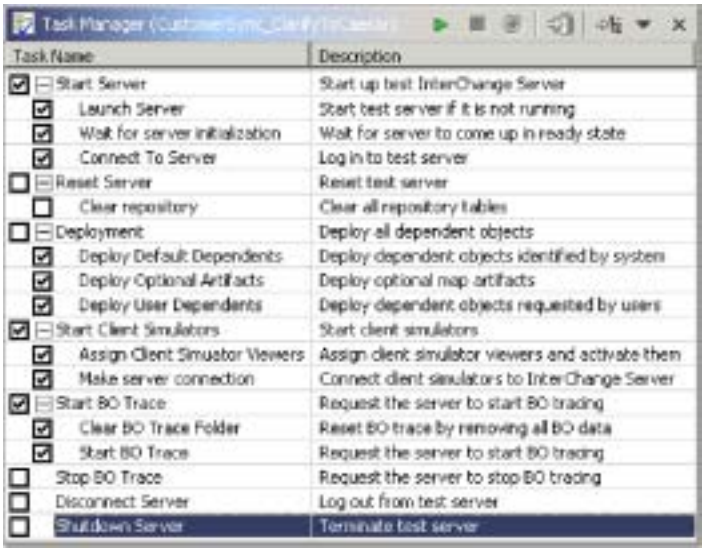

Figure 83. Task Manager view

Table 25 on page 235 describes the tasks and task groups in the Task Manager view.

Table 25. Tasks and task groups in the Task Manager view

Task name	Task description
Start Server	This is the group for all tasks related to starting InterChange Server.
Launch Server	This task starts the InterChange Server instance. This task belongs to the **Start Server** task group.
Wait for server initialization	This task causes Integrated Test Environment to wait until InterChange Server has started successfully before trying to connect to it. If you have not selected this task and Integrated Test Environment tries to connect to InterChange Server before the server starts, then Integrated Test Environment will not be able to connect to it. Enabling this option ensures that the task of connecting Integrated Test Environment to InterChange Server is automated.
Connect to Server	This task connects Integrated Test Environment to the InterChange Server instance so that you can manage and deploy components and view data. This task belongs to the **Start Server** task group. **Note:** The IBM Java Object Request Broker (ORB) must be running for Integrated Test Environment to connect to the server.
Reset Server	This is the group for the **Clear Repository** task.
Clear Repository	This task deletes the repository of the InterChange Server instance. You would want to perform this task if you plan to deploy a large number of components through Integrated Test Environment and some of the components already exist in the repository. This task belongs to the **Reset Server** task group.
Deployment	This is the group for all tasks related to deploying components to the InterChange Server repository through the Task Manager view. All of the dependent components for an interface must be deployed to test the interface. To deploy components using Integrated Test Environment, you must make sure the InterChange Server instance starts in design mode. For more information, see “Configuring InterChange Server to start in design mode” on page 223. For more information on the Dependency view, see “Using the Outline view” on page 229.
Deploy Default Dependents	This task deploys all the components that are determined by the system to be required for the interface to function, and that are listed beneath the **Default Dependents** node in the Dependency view. This task belongs to the **Deployment** task group.
Deploy Optional Maps	
Deploy User Dependents	This task deploys all the components that you add to the **User Artifacts** node in the Outline view. This task belongs to the **Deployment** task group.

Table 25. Tasks and task groups in the Task Manager view (continued)

Task name	Task description
Start Client Simulators	This is the group for all tasks relating to emulating connector agents and access clients.
Assign Client Simulator Viewers	This task starts Client Simulator view windows for all of the connectors and access clients involved in the interface. This task belongs to the **Start Client Simulators** task group.
Make server connection	This task causes the Client Simulator views to connect to the InterChange Server instance. Connector agents and access clients must be connected to InterChange Server to exchange business objects. By default a connector starts using its definition in the repository. You can use the Client Simulator view to cause the agent to connect with a configuration file as described in "Emulating a connector using the repository definition" on page 250. Client Simulators can only automatically connect to the server in this way if the `DeliveryTransport` property of the connector being emulated is set to the value `IDL`. This task belongs to the **Start Client Simulators** task group. **Note:** The IBM ORB must be running for Integrated Test Environment to connect to the server.
Start BO Trace	This is the group for all tasks relating to business object tracing. Business object tracing records business object data as business objects are processed by components in the system. For more information, see "Using the BO Inspector view" on page 262.
Clear BO Trace Folder	This task clears the folder in which business object trace data is stored. You might want to clear the business object trace folder for the following reasons: • Business object trace files take up disk storage and use resources to be loaded for viewing in the Integrated Test Environment perspective. You might want to clear the folder to preserve storage and resources. • You might not want old test data displayed in the Integrated Test Environment. You can clear the business object test data from previous tests so that only data from subsequent tests is displayed. This task belongs to the **Start BO Trace** task group.
Start BO Trace	This task starts business object tracing. You must start business object tracing for Integrated Test Environment to store business object data during the tests you perform. This task belongs to the **Start BO Trace** task group.

Table 25. Tasks and task groups in the Task Manager view (continued)

Task name	Task description
Stop BO Trace	This task stops business object tracing. Business object tracing can be a very resource-intensive task, so depending on your objectives for the testing you might want to stop business object tracing if it is more disruptive than helpful.
Disconnect Server	This task disconnects Integrated Test Environment from the InterChange Server instance. It does not shut down the InterChange Server instance.
Shutdown Server	This task shuts down the InterChange Server instance.

Making task selections

There are several ways to select tasks and task groups. Table 26 describes the different ways. It is recommended that you combine the different approaches to be most efficient.

Table 26. Ways of selecting tasks and task groups in the Task Manager view

To accomplish this...	Do this...
Check a single task within a group	Click the checkbox next to the task.
Check a group of tasks	Click the checkbox next to the task group.
Check all tasks in the view	Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Check All** from the menu.
Uncheck all tasks in the view	Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Uncheck All** from the menu.
Check your preferred tasks	Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Preference Selection** from the menu. For more information, see “Configuring your preferred task selections” on page 240.

Table 26. Ways of selecting tasks and task groups in the Task Manager view (continued)

To accomplish this...	Do this...
Check the default tasks	Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Default Selection** from the menu. The default task and task group selections are: • Start Server – Launch Server – Wait for server initialization – Connect to Server • Deployment – Deploy Default Dependents – Deploy Optional Artifacts – Deploy User Dependents • Start Client Simulators – Assign Client Simulator Viewers • Start BO Trace – Start BO Trace

Executing tasks using the Task Manager view

There are several ways to execute the tasks you have selected in the Task Manager view. The following sections describe the different methods:

- "Executing selected groups of tasks"
- "Executing a group of tasks" on page 239
- "Executing a single task" on page 239

Executing selected groups of tasks

You can execute all of the tasks within several task groups at once. This makes it easy to prepare the interface for testing without manually managing it through the different stages.

A convenient approach, for instance, is to execute all of the tasks within the **Start Server**, **Start Client Simulators**, and **Start BO Trace** task groups.

Do the following to execute all of the tasks within a selection of task groups:

1. Enable the checkboxes for the tasks you want to execute.
2. Enable the checkboxes for the task groups that the tasks you want to execute belong to.
3. Do one of the following to execute the selected tasks in the selected task groups:
   - Select **Integrated Test Environment > Run** from the menu bar.
   - Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Run** from the menu.
   - Click the **Execute tasks** button in the title bar of the Task Manager view.

Be sure not to enable checkboxes for tasks or task groups that have conflicting results. For instance, do not enable both the **Launch Server** and **Shutdown Server** tasks.

Executing a group of tasks

To execute a single group of tasks, right-click the task group and choose **Run**. Only the tasks in the task group you right-clicked will execute. The checkbox for the task group does not have to be enabled for its tasks to execute, though the checkboxes for the tasks you want to execute within the group do have to be enabled.

You might use this approach frequently with the **Start Client Simulators** task group. You will frequently execute groups of tasks to initially prepare a test, as described in "Executing selected groups of tasks" on page 238. You might close the Client Simulator views you have open during the course of a series of tests, however, and then need to launch them and connect them again. You do not have to execute all of the tasks that have already been performed, however—you can execute only the **Start Client Simulators** task group.

Executing a single task

To execute a single task, right-click the task and choose **Run**. Only the task you right-clicked will execute. The checkbox for the task does not have to be enabled for it to execute.

You might use this approach to shut down the server. You might want to start the server again immediately afterwards, and not want to re-enable all the required checkboxes. You can just right-click the **Shutdown Server** task and execute it alone.

Stopping tasks

You can instruct Task Manager to stop its queue of currently executing tasks. It will allow whichever task is currently executing to finish, but will not execute any subsequent tasks. For example, if you choose to stop the tasks while the **Launch Server** task is executing, then Task Manager will allow the server to finish starting.

Do one of the following to stop the currently executing tasks:

- Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Stop Tasks** from the menu.
- Click the **Stop tasks** button in the title bar of the Task Manager view.

Killing tasks

You can instruct Task Manager to kill its currently executing tasks. It will terminate the currently executing task and will not execute any subsequent tasks either. For example, if you choose to kill the tasks while the **Launch Server** task is executing, then Task Manager does not allow the server to finish starting.

To kill the currently executing tasks, click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Kill Tasks** from the menu.

Pausing tasks

You can pause the currently executing tasks so that Task Manager allows the currently executing task to complete, but does not execute the remaining tasks until you unpause it. This allows you to start a series of tasks and perform an action which must be done before a later task is reached, but which you forgot about, and without stopping and restarting the remaining tasks. For example, if you want to start the server and deploy the dependents, but some of the components already exist in the repository and you do not realize that until you

have already executed the tasks, you could pause the tasks before the deployment stage begins. Then you can use repos_copy to delete the repository so that when the deployment begins it does not fail.

Do one of the following to pause the currently executing tasks:

- Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Pause Tasks** from the menu.
- Click the **Pause tasks** button in the title bar of the Task Manager view.

Do one of the following to cause the tasks to resume executing:

- Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Pause Tasks** from the menu.
- Click the **Pause tasks** button in the title bar of the Task Manager view.

Configuring your preferred task selections

You can save selections as a preference so that you can conveniently apply the selection scheme you use most commonly.

Do the following to configure your preferred task selections:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Expand the **Integrated Test Environment** node.
3. Select **Task Manager**.
4. Expand task groups and enable the checkboxes for the individual tasks and task groups that you want selected when you apply your preferences.
5. Click **OK**.

Figure 84 on page 241 shows the Task Manager preference selection interface.

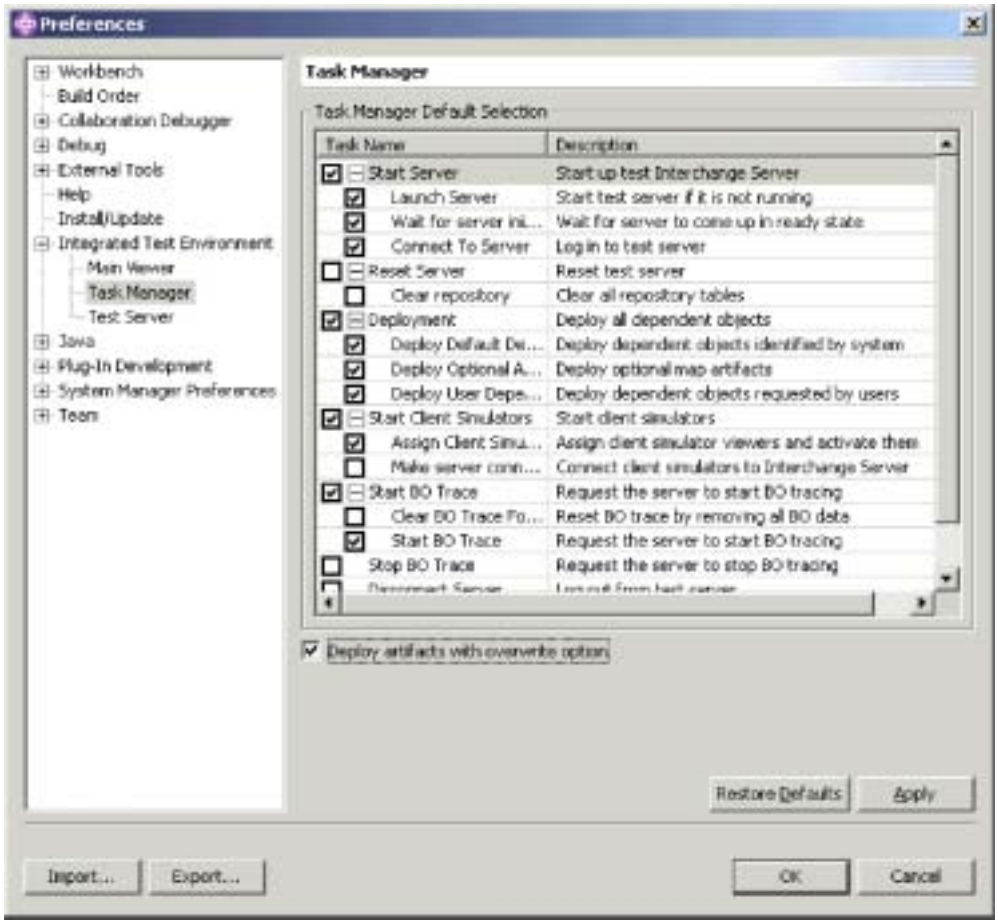

Figure 84. Integrated Test Environment task preferences

Configuring the default task selections

Do the following to configure the tasks and task groups that will be enabled by default each time you open a particular test unit:

1. Open the test unit for which you want to save the default selections.
2. Expand task groups and enable the checkboxes for the individual tasks and task groups that you want selected by default when you open a test unit.
3. Do one of the following to store the selections:
   - Click the down arrow in the right-hand corner of the title bar of the Task Manager view and choose **Store Selections** from the menu.
   - Click the **Store current selections** button in the title bar of the Task Manager view.

When you open the test unit the next time, the tasks you chose will be selected in the Task Manager view.

Using the Integrated Test Environment Console and InterChange Server Console views

As described in “Integrated Test Environment interface” on page 216, the Integrated Test Environment Console tab displays information about the execution of Integrated Test Environment and the InterChange Server Console displays information about the execution of InterChange Server.

You can use the Integrated Test Environment Console view to determine when Integrated Test Environment tasks have completed and if they were successful. For instance, messages are written when Integrated Test Environment connects successfully to InterChange Server, or when Integrated Test Environment launches Map Designer to debug a map.

If you started InterChange Server from within Integrated Test Environment, the InterChange Server Console view displays the logging and tracing output of the server, just as the MS-DOS console does. You can then use the view to troubleshoot the interfaces you are testing. It has the benefit that it is contained within the Integrated Test Environment perspective and can be conveniently searched.

The following sections describe some tasks you will frequently want to perform when using the Integrated Test Environment Console and InterChange Server Console views.

Finding text

You might want to search the Integrated Test Environment Console or InterChange Server Console views for particular phrases. For instance, you might want to confirm that the InterChange Server started successfully. You could search the InterChange Server Console view for the text “is ready” to do so.

Do the following to search the Integrated Test Environment Console or InterChange Server Console views for text strings:

1. Right-click in the Integrated Test Environment Console or InterChange Server Console view and select **Find/Replace** from the context menu.
2. Type the text string for which you want to search in the **Find** field.
3. Enable the **Forward** radio button to search the text after the cursor or enable the **Backward** radio button to search the text before the cursor.
4. Enable other options such as the **Case Sensitive** checkbox.
5. Click **Find Next**.

Copying text

You might want to copy text from the console views to show the text to other developers or Technical Support to get help with troubleshooting.

To copy text from a console view, select the desired text, then right-click it and choose **Copy** from the context menu.

Maximizing the view window

As with all workbench editors and viewers, you can maximize the Integrated Test Environment Console or InterChange Server Console views by double-clicking their title bar. Although this is not useful with most viewers, it can be very helpful when troubleshooting an interface by examining information in the InterChange Server Console view.

Clearing the view window

You might want to clear the Integrated Test Environment Console or InterChange Server Console views to eliminate the information displayed in them. This can be helpful when you want to view only information that is relevant to debugging a problem in the interface. You can clear the console view, reproduce the problem, and troubleshoot it without having to search through information that is not related to the problem.

Configuring Integrated Test Environment Console preferences

Do the following to configure your preferences for the Integrated Test Environment Console:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Select **Integrated Test Environment**.

   Figure 77 on page 225 shows the Integrated Test Environment preferences.
3. Click the button to the right of the "Normal text" label.
4. At the "Color" dialog, select the color you would like the Integrated Test Environment Console to use for messages that are not warnings or errors.
5. Click **OK**.
6. Repeat steps 3 through 5 for the "Warning text" and "Error text" labels to set the color for those types of messages.
7. Click **OK**.

Using the Test Unit view

As described in "Test Unit editor" on page 218, the Test Unit view provides a graphical view of the interface and provides mechanisms to manipulate the components in the interface.

Using the Server Context Overlay

The Server Context Overlay displays information about the components in the interface within the Test Unit view and enables context menu options for deploying components and manipulating their states.

To enable or refresh the Server Context Overlay as described in the following sections, Integrated Test Environment must be connected to the server. For more information, see the **Connect to Server** task in Table 25 on page 235.

When the Server Context Overlay is enabled, icons are added to the icons for each component shown in the Test Unit view to indicate the state of the component. This is useful because all the components that participate in an interface must be running in order to test it.

Figure 85 on page 244 shows a Test Unit view where the Server Context Overlay is enabled. Most of the components have green arrow icons, indicating they are in a started state. The **ClarifyConnector** component has an orange circle indicating that it is in a paused state. The collaboration object **CustomerSync_ClarifyToCaesar** has a red square icon, indicating that it is in a stopped state. Table 27 on page 244 describes the different component state icons and their meanings.

For information on component states, see the *System Administration Guide*.

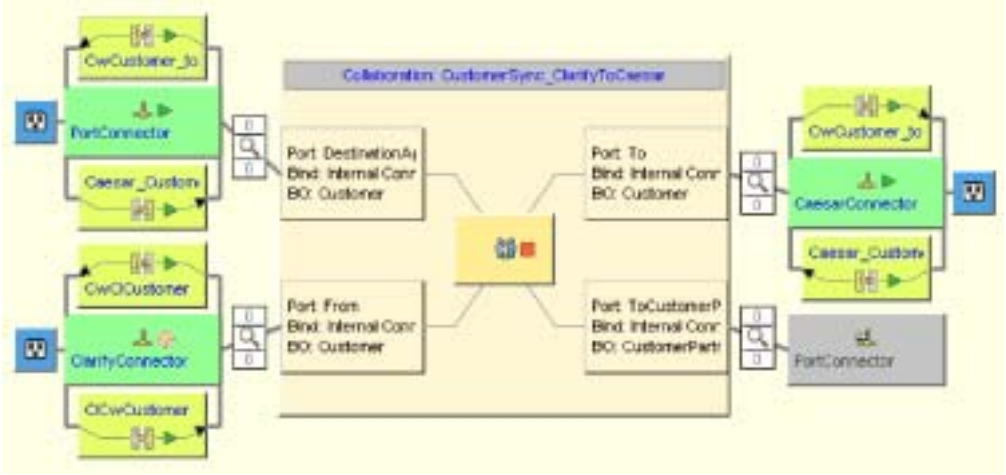

Figure 85. Test Unit view with Server Context Overlay enabled

Table 27. Component state icons and meanings

Icon	Meaning
Green arrow	The component is started.
Orange circle	The component is paused.
Red square	The component is stopped.
Question mark	The component is in an unknown state.
Red circle with white “X”	The component does not exist in the repository.
Blue checkmark	The component deployed successfully.
Red “X”	Integrated Test Environment is disconnected from the server, so the Server Context Overlay cannot display the current state of the component.

Enabling and disabling the Server Context Overlay

To enable the Server Context Overlay, right-click the background of the Test Unit view and select **Server Context Overlay** from the context menu.

When the Server Context Overlay is enabled, the component state icons appear in the Test Unit view and right-click menu items for the components are enabled.

To disable the Server Context Overlay, right-click the background of the Test Unit view and select **Server Context Overlay** from the context menu.

To have the Server Context Overlay always appear, enable the **Show Server Context Overlay always** option in the Integrated Test Environment preferences. Figure 77 on page 225 shows the Integrated Test Environment preferences.

Refreshing the Server Context Overlay

The state information in the Server Context Overlay is not dynamic, so if the state of a component changes after the Server Context Overlay is initially enabled due to actions external to Integrated Test Environment, then the state information the Server Context Overlay presents will be inaccurate. For instance, if you change the state of a component using the InterChange Server Component Management view in System Manager, the Server Context Overlay will still reflect the state of the component prior to the change. You must refresh the Server Context Overlay to update it with current state information.

To refresh the Server Context Overlay, right-click the background of the Test Unit view and select **Refresh Server Context Overlay** from the context menu.

Managing the repository using the Test Unit view

All of the components required for an interface must be deployed to the server repository to test the interface. The Outline view provides a way to work with the InterChange Server repository to ensure that it contains the required components.

To deploy a component when it already exists in the server repository, right-click the component in the Test Unit view and choose **Deploy with overwrite** from the context menu.

To deploy a component if it does not already exist in the server repository, right-click the component in the Test Unit view and choose **Deploy** from the context menu. If the selected components already exist in the server repository and you use the **Deploy** option then the deployment will fail.

To remove a component from the server repository using the Test Unit view, right-click the component in the view and choose **Remove from server** from the context menu.

Entries are written to the Integrated Test Environment Console and InterChange Server Console indicating whether the deployment was successful or not.

Managing component states using the Test Unit view

You can change the state of a component using the Test Unit view. This allows you to resolve problems you discover during testing—you can deactivate the component, modify it, re-deploy it, and re-activate it without using System Manager, repos_copy, or System Monitor.

To change the state of a component, right-click its icon in the Server Context Overlay and choose the desired state operation—**Start**, **Pause**, or **Stop**. Not all component types have all state operations available.

For information on component states, see the *System Administration Guide*.

Using, showing, and closing Client Simulator views using the Test Unit view

You use Client Simulator views to emulate connectors and access clients when testing an interface. Client Simulator views provide interfaces to create business object instances with test data, send them as business object requests, and reply to them as business object responses.

To load the profile for a connector into a particular Client Simulator view, right-click the icon for the connector in the Test Unit view and select **Use Client Simulator *x***, where ***x*** is the number of the Client Simulator view position you want to use for that connector.

Table 29 on page 249 shows the default interface positions in which the different Client Simulator views are displayed.

Views in Integrated Test Environment can be concealed by other views during different stages of a test; for example, a Client Simulator view might be concealed

by a BO Inspector view. To reveal a Client Simulator view that is concealed, right-click the connector icon and select **Show Client Simulator** to give focus to the Client Simulator view.

To close a Client Simulator view, right-click the connector icon and select **Close Client Simulator**.

Note: This option not only closes the Client Simulator view, but also causes the Client Simulator view to stop emulating the connector.

For more information on working with Client Simulator views, see "Using the Client Simulator view" on page 248.

Showing BO Inspector views

If you use business object tracing, Integrated Test Environment records business object data at certain points during the execution of the interface.

To show the business object data for a BO Marker into a particular BO Inspector view, right-click the BO Marker and select **Show in BO Viewer*x***, where ***x*** is the number of the BO Inspector view position you want to use for that marker. BO Marker icons are identified in Figure 85 on page 244.

Table 31 on page 263 shows the default interface positions in which the different BO Inspector views are displayed.

For more information on using business object tracing, see Table 25 on page 235.

For more information on working with BO Inspector views, see "Using the BO Inspector view" on page 262.

Refreshing BO Markers

If you are using business object tracing but have disabled auto-update of the trace information then you must explicitly refresh the BO Markers to update them.

To update BO Markers, right-click the Server Context Overlay and select **Refresh BO Markers** from the context menu.

Debugging maps using the Test Unit view

If you use business object tracing you can launch Map Designer in debug mode for maps that transform generic objects to application-specific objects, using the data produced during the execution of the test within Integrated Test Environment.

Do the following to debug maps using the Test Unit view:

1. Enable the Server Context Overlay as described in "Enabling and disabling the Server Context Overlay" on page 244.
2. Execute the Start BO Trace task by following the instructions in one of the following sections:
   - "Executing tasks using the Outline view" on page 231
   - "Using the Task Manager view" on page 234.
3. When you have sent a business object through the interface, right-click the icon for the map you want to debug and select **Debug Map** from the context menu.
4. At the "Select a BO Instance" window select the business object instance you want to use and click **OK**.

Map Designer launches in debug mode with the specified map opened and the specified test data supplied. For more information on how use the debugging features of Map Designer, see the *Map Development Guide*.

Note: Map Designer executes independently of Integrated Test Environment. Integrated Test Environment does not continue to communicate with Map Designer after you launch it.

Changing the Test Unit view scale

There are several options available for changing the scale of the Test Unit view.

To change the Test Unit view scale, right-click the Server Context Overlay and select the desired scale from the context menu. Table 28 lists and describes the scale options.

Table 28. Test Unit view scales and descriptions

Test Unit view scale type	Description
Fit Window	This scale deforms the interface illustration to fit the Test Unit view horizontally.
Fill Window	This scale deforms the interface illustration to fill the Test Unit view. Typically this results in the interface icons being stretched vertically.
View Normal Scale	This is the default scale.
View Double Scale	This scale increases the magnification of the interface illustration by two hundred percent.

Configuring Test Unit view preferences

Do the following to configure your preferences for the Test Unit view:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Expand **Integrated Test Environment**.
3. Select **Main Viewer**.

   Figure 77 on page 225 shows the Integrated Test Environment preferences.

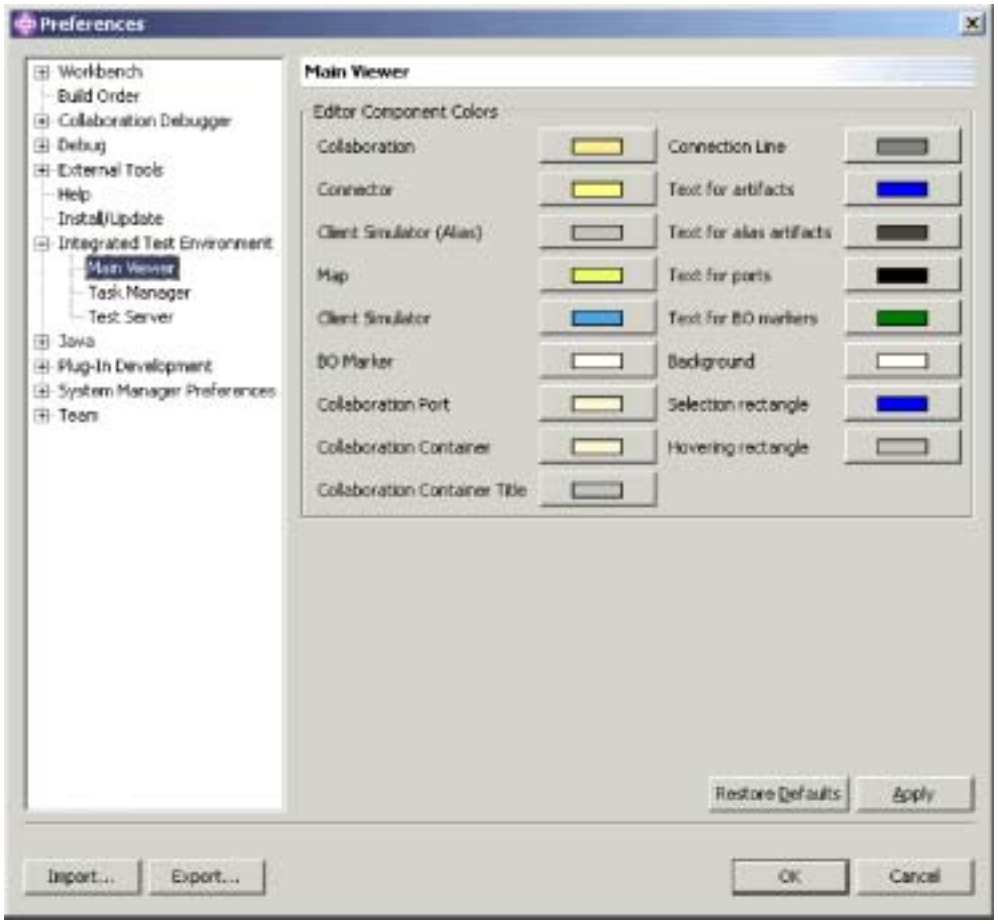

Figure 86. Test Unit view preferences

4. Click the button to the right of a component label in the “Edit Component Colors” pane.
5. At the “Color” dialog, select the color you would like the Test Unit view to use for the background of that component type.
6. Click **OK**.
7. Repeat steps 4 through 6 for the other component types.
8. Click **OK**.

Using the Client Simulator view

The Client Simulator view allows you to emulate the clients in IBM WebSphere InterChange Server interfaces—connector agents and access clients. The view provides interfaces to connect the clients to the server, create and send business object requests, and reply with business object responses.

Showing and closing Client Simulator views

The following sections describe the numerous ways you can show and close Client Simulator views.

The “Default interface position” column in Table 29 on page 249 lists the views whose default positions in the interface are overlaid with the Client Simulator views in the “Client Simulator view number” column.

Table 29. Default Client Simulator view positions

Client Simulator view number	Default interface position
Test Connector 1	ITE Navigator
Test Connector 2	Outline
Test Connector 3	Properties
Test Connector 4	Dependency
Test Connector 5	ITE Navigator
Test Connector 6	Outline
Test Connector 7	Properties
Test Connector 8	Dependency
Test Connector 9	ITE Navigator

Showing Client Simulator views

Use one of the following techniques to show Client Simulator views for the connectors and access clients in the interface you are testing:

- "Showing and closing Client Simulator views using the Outline view" on page 233 along with Table 25 on page 235
- "Using the Task Manager view" on page 234 along with Table 25 on page 235
- "Using, showing, and closing Client Simulator views using the Test Unit view" on page 245 along with Table 25 on page 235
- Select **Integrated Test Environment > Show All Client Simulator Views** from the menu bar to show all nine Client Simulator views.
- Select **Window > Show View > *x*:Client Simulator** from the menu bar to launch a specific Client Simulator view, where ***x*** is the number of the Client Simulator view position you want to use for that connector.

Table 29 shows the default interface positions in which the different Client Simulator views are displayed.

Closing Client Simulator views

Use one of the following techniques to close Client Simulator views:

- "Showing and closing Client Simulator views using the Outline view" on page 233 along with Table 25 on page 235.
- "Using, showing, and closing Client Simulator views using the Test Unit view" on page 245.
- Select **Integrated Test Environment > Close All Client Simulator Views** from the menu bar to close all open Client Simulator views.
- Either click the **Close** button in the right-hand side of a Client Simulator view title bar, or right-click the Client Simulator view title bar and select **Close** from the context menu.

Connecting a Client Simulator view to the server

You must connect Client Simulator views to the server to emulate a connector as part of testing an interface.

When using the Client Simulator view, you can either emulate a connector using the definition in the repository or using a configuration file. You can also use the Client Simulator view to emulate an access client.

Note that you can also use the Task Manager view to connect agents to the server. For more information, see Table 25 on page 235.

Emulating a connector using the repository definition

Do one of the following to connect to the server using the repository definition of the connector:

- Click the **Connect** button in the toolbar.
- Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Connect**.

Note: The `DeliveryTransport` property of the connector you are emulating must be set to the value `IDL` to connect to the server using the repository definition.

Emulating a connector using a configuration file

Do the following to connect to the server using a configuration file:

1. Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Connect with *.cfg**.
2. Navigate to and open the connector configuration file.

Note: If the `DeliveryTransport` property of the connector you are emulating is not set to the value `IDL` then and you select **Server > Connect** from the menu, then Integrated Test Environment presents the dialog to specify the configuration file to be used.

Emulating an access client

To emulate an access client, click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Connect**.

Confirming that a client has connected to the server

Use the following indicators to determine if a Client Simulator has successfully connected to the server:

- In the Outline view, the node for the port through which the client communicates turns green and reads “Ready”.
- If the client is a connector, the message “[AppConnector: Connector has recovered]” is written to the Status Pane of the Client Simulator view.
- If the client is an access client, the message “Ready” is written to the Status Pane of the Client Simulator view.
- The **BO Type** and **BO Instance** drop-down menus become enabled on the Input Pane of the Client Simulator view.

For more information about the Status Pane, see “Using the Status Pane” on page 252.

For more information about the Input Pane, see “Using the Input Pane” on page 251.

Disconnecting from the server

To disconnect a Client Simulator that is emulating a connector from the server, click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Disconnect**.

A Client Simulator view also disconnects from the server when you close it. The only way to disconnect a Client Simulator view that is emulating an access client from the server is to close the Client Simulator view. For more information on closing Client Simulator views, see “Closing Client Simulator views” on page 249.

Client Simulator view panes and pane arrangements

The Client Simulator view has several panes you use to simulate the behavior of a connector or access client.

Using the Input Pane

You use the Input Pane in the Client Simulator view to work with and send business object requests. You can create business object instances, add data to their attributes, and send them as requests. You typically use the Input Pane when working with a Client Simulator view that is emulating a source client in an interface.

Figure 87 shows the Input Pane for a Client Simulator view that is emulating a source connector.

Figure 87. Client Simulator view Input Pane

Using the Result Pane

You use the Result Pane in the Client Simulator view to work with business object requests that have been received and to send business object responses. You typically use the Result Pane when working with a Client Simulator view that is emulating a destination client in an interface.

Figure 88 on page 252 shows the Result Pane for a Client Simulator view that is emulating a destination connector.

Figure 88. Client Simulator view Result Pane

Using the Status Pane

The Status Pane contains messages about the execution of the Client Simulator view. For instance, the message "[AppConnector: Connector has recovered]" is written to the Status Pane when a connector has successfully connected to the server.

You use the Status Pane primarily to confirm that Client Simulator has successfully connected to the server and to perform troubleshooting. For instance, if you send a business object request from a Client Simulator view and there is no subscription for the business object type, then a message is written to the Status Pane. This can help you to determine why a business object request does not appear in the Result Pane of a Client Simulator view emulating a destination client in an interface.

Figure 89 on page 253 shows the Status Pane for a Client Simulator view.

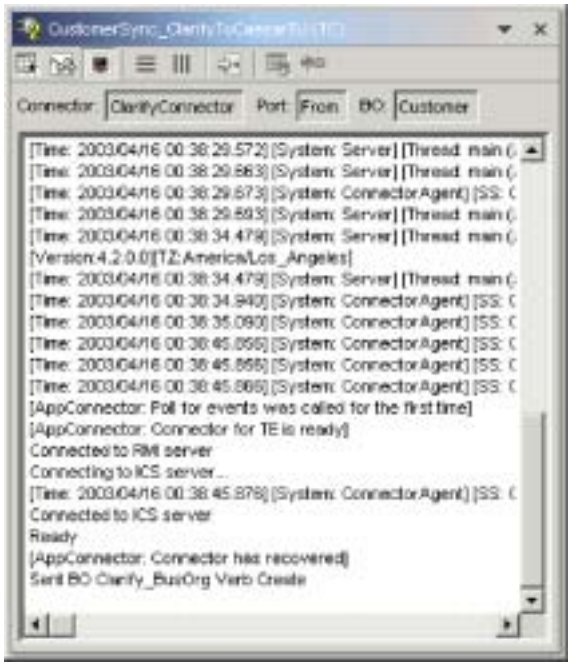

Figure 89. Client Simulator view Status Pane

Arranging all panes vertically

You can arrange a Client Simulator view so that it displays the Input, Result, and Status panes simultaneously, stacked on top of one another. This view can be helpful when a Client Simulator view is emulating a client that has to handle both event notification and request processing responsibilities.

Do one of the following to arrange the Client Simulator view panes vertically:

- Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and choose **Window > Arrange Vertical** from the menu.
- Click the **Show all vertically** button in the title bar of the Client Simulator view.

Figure 90 on page 254 shows the Client Simulator view panes when they have been arranged vertically.

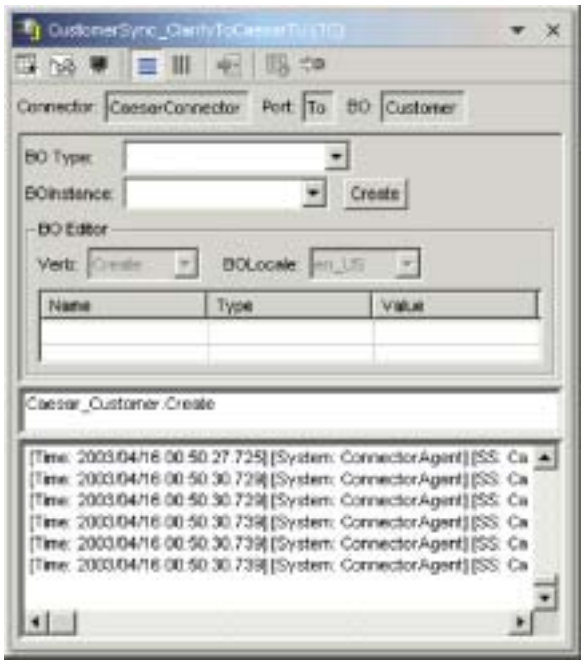

Figure 90. Arranging Client Simulator view panes vertically

Arranging all panes horizontally

You can arrange a Client Simulator view so that it displays the Input, Result, and Status panes simultaneously from left to right. This view can be helpful when a Client Simulator view is emulating a client that has to handle both event notification and request processing responsibilities.

Do one of the following to arrange the Client Simulator view panes horizontally:

- Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and choose **Window > Arrange Horizontal** from the menu.
- Click the **Show all horizontally** button in the title bar of the Client Simulator view.

Figure 90 shows the Client Simulator view panes when they have been arranged horizontally.

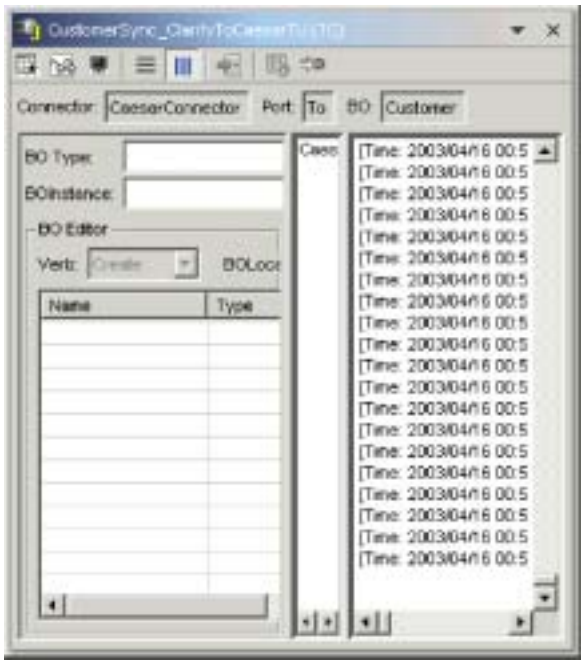

Figure 91. Arranging Client Simulator view panes horizontally

Working with business object requests in the Client Simulator view

Request business objects are those that you send from a Client Simulator view when it is emulating a connector that is the source of the events that trigger an interface. Working with request business objects consists of creating a business object instance, populating it with data, and sending the request.

Creating request business objects

Do the following to create a new business object instance:

1. In the Input Pane, select the name of the business object you want to create from the **BO Type** drop-down menu.
2. Do one of the following:
   - Click **Create** next to the **BO Instance** field.
   - Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Edit > Create BO**.
3. When presented with the "New Instance" dialog, type a name for the instance in the **Enter Name** field.
4. Select the desired verb from the **Verb** drop-down menu.
5. Select the desired locale from the **BO Locale** drop-down menu.
6. Provide values for the simple attributes and child business objects within the top-level object, as described in "Setting values for business object attributes" on page 257.
7. Click **OK**.

Sending request business objects asynchronously

When a source client sends a request business object in asynchronous mode, it does not expect to get back a response business object. Once the request business

object is dispatched, the source client's role in the transaction is finished. The response business object is typically processed by the integration broker. The default mode for a Client Simulator view is asynchronous.

To send a business object asynchronously, do the following:

1. Create the request business object instance, as described in "Creating request business objects" on page 255.
2. Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Mode > Asynchronous**.

   Note: The Client Simulator view operates in "Asynchronous" mode by default, so you only have to perform this step if you previously were sending synchronous requests from the view. You do not have to set the mode before sending each request.
3. Do one of the following to send the request:
   - Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Send**.
   - Click the **Send BO** button in the toolbar of the Client Simulator view.

Sending business object requests synchronously

When a source client sends a request business object synchronously, it expects to get back a response business object from the integration broker after any destination applications have processed the request.

1. Create the request business object instance, as described in "Creating request business objects" on page 255.
2. Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Mode > Synchronous** to set the Client Simulator view to synchronous mode.
3. Do one of the following to send the request:
   - Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Send**.
   - Click the **Send BO** button in the toolbar of the Client Simulator view.
4. When presented with the "Select Collaboration" dialog, select the collaboration to which the business object should be sent from the **Collaboration** drop-down menu and click **OK**.

   The business object request is sent to the configured port of the collaboration object chosen for processing.

Sending business object requests in batch mode

In batch mode, the Client Simulator view lets you specify the number of instances of a particular business object you want to send, as well as one attribute in the top-level object —a primary key attribute, for example—that you want set to a unique value for each instance. The Client Simulator view copies the business object as many times as you have specified, incrementing the value of the single attribute you specified for each instance, and then sends each instance. This option allows you to create a large number of business objects quickly and easily.

If the selected attribute is a key field that participates in dynamic cross-referencing as part of an identity relationship, then you must guarantee that the initial value and all those that follow it are unique. Otherwise, the cross-referencing logic will fail, causing the request business objects to fail.

To ensure that the values are unique, you can use Relationship Manager or execute SQL statements against the table for the relationship participant as follows.

- Determine the highest current value for the participant and set the Initial Value field to an even higher value. The first business object instance in the batch and all those that follow will then be unique.
- Delete the existing table entries for the participant, thus guaranteeing that no entries have the same attribute value as any of the batch business objects.

To send business objects in batch mode, do the following:

1. Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Server > Send Batch**.
2. In the "Batch Mode" window, select the desired verb from the **Verb** drop-down menu.
3. Select the desired locale from the **BO Locale** drop-down menu.
4. Select from the **Attribute** list the attribute in the top-level business object that you want incremented with each business object request in the batch.

   The selected attribute should typically be an attribute that uniquely identifies the business object, such as a primary key.
5. In the **Initial Value** field, type the starting value for the attribute to be incremented.
6. In the **No. of BO's** field, type the number of business object instances you want generated and sent.
7. Click **OK**.

   The Client Simulator view generates the number of business objects you specified, all identical with the exception of the one specified attribute, whose value is incremented for each instance.

Figure 92 shows a batch mode configuration in which:

- Fifty business objects are to be sent.
- The value of the attribute `OrgObjid` is to be incremented.
- The starting value for the attribute is 10000.

Figure 92. Sending business object requests in batch mode

Setting values for business object attributes

You can set values for business object attributes when creating a business object request and when editing a business object request that has been received, so that it can be returned as a response.

To set values for the attributes of a business object you intend to send as a request, you must use the Input Pane of the Client Simulator view. For more information, see “Using the Input Pane” on page 251.

To set values for the attributes of a business object you are editing to send as a response, you use the “Response BO” window. For more information, see “Editing response business objects” on page 259.

Setting values for simple attributes

To provide a value for a simple attribute, click its cell in the **Value** column of the Input Pane or the “Response BO” window and enter a value.

Adding child business objects

To add an instance of a child business object, right-click the attribute that represents the child object in the Input Pane or the “Response BO” window and select **Add Instance** from the context menu.

A plus sign (+) is added next to the attribute that represents the child business object to show that there is at least one child business object instance. If you expand the child object attribute, numbered entries are displayed for each instance. The individual instances also have plus signs (+) next to them, so you can expand them and set values for their attributes.

To add more child business object instances, right-click the attribute that represents the child object and select **Add Instance** from the context menu.

Note: If the **Card** property of the attribute that references the child business object is set to the value 1 (indicating it is of single-cardinality), then you will only be able to add one instance of the child object.

Removing child business objects

To remove an instance of a child business object, right-click the instance and select **Remove Instance** from the context menu.

To remove all instances of a child business object, right-click the attribute that represents the child business object and select **Delete All Instances** from the context menu.

Setting the verb of a child business object

You can set the verb of a child business object to test the effect that value has on the business process. This can be helpful when you are troubleshooting logic that involves the cross-referencing of child objects.

To set the verb of a child business object instance, right-click it and choose **Set Verb** from the context menu. When presented with the “Select Verb” prompt, selected the desired verb and click **OK**.

Resetting business object attributes to their default values

To reset the attributes of a business object request to their default values, click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Edit > Reset BO**.

To reset the attributes of a business object response to their default values, click the **Reset BO to default** button in the toolbar of the “Response BO” window.

Clearing business object attribute values

To clear the attributes of a business object request of their values, click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Edit > Clear BO**.

To clear the attributes of a business object response of their values, click the **Clear BO values** button in the toolbar of the "Response BO" window.

Working with response business objects

Response business objects are those that you send from a Client Simulator view when it is emulating a connector that is the recipient of business object requests in an interface. Working with request business objects consists of editing the values in the business object instance and sending the response back to the server.

If you create and send a business object request as described in "Working with business object requests in the Client Simulator view" on page 255 and it is successfully processed by the interface, then the business object appears in the Result Pane of any destination connectors in the interface. Figure 88 on page 252 shows a Result Pane in which a business object request has been received.

"Editing response business objects" describes how to edit a business object request that has been received, and "Sending response business objects" describes how to send the instance as a response.

Editing response business objects

When you receive a business object request in a Client Simulator view that is emulating a destination client in an interface, you frequently want to edit the values of the attributes. For instance, you will want to provide unique values for primary key attributes that participate in relationships, or you will want to modify the value of other attributes to test map or collaboration logic that responds differently depending on the exact values in the business object. Do the following to set the values of business object attributes:

1. Do one of the following to edit the business object request in the "Response BO" window:
   - Double-click the business object instance in the Result Pane.
   - Select the business object instance in the Result Pane, then click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Reply > Edit Response**.
2. Do one of the following to edit the attributes of the business object:
   - Use one of the techniques described in "Setting values for business object attributes" on page 257 to modify the values of the business object attributes.
   - Import business object data from a file as described in "Importing a response business object" on page 261.

     The ability to import saved data into a business object request is very useful in situations where you have to populate a response business object with data before sending it as a reply. Instead of manually typing a value for each attribute that requires response data, you can type the values once, export the business object (as described in "Exporting a response business object" on page 261), and then import the saved data on subsequent tests.

Sending response business objects

After you edit a request business object (if editing is needed), you send it back to the server as a reply.

Table 30 lists the reply options and shows their corresponding connector return codes for both C++ and Java connectors. For more detailed information about C++ or Java Connector return codes, see the *Connector Development Guide for Java or C++*.

Table 30. Client Simulator view reply types and connector return codes.

Client Simulator view reply type	C++ connector return code	Java connector return code
Success	BON_SUCCESS	SUCCESS
Fail	BON_FAIL	FAIL
Multiple Hits	BON_MULTIPLE_HITS	MULTIPLE_HITS
Retrieve By Content Fail	BON_FAIL_RETRIEVE_BY_CONTENT	RETRIEVEBYCONTENT_FAILED
Not Found	BON_BO_DOES_NOT_EXIST	BO_DOES_NOT_EXIST
Value Duplicate	BON_VALDUPES	VALDUPES

To reply to a request business object, do the following:

1. Select the business object instance in the Request Pane.
2. Do one of the following:
   - Click the **Reply Success** or **Reply Fail** buttons in the toolbar of the Client Simulator view.
   - Click the down arrow in the right-hand corner of the title bar of the Client Simulator view, select the **Reply** submenu, and then select the desired type of reply.

Saving business objects

You can save a business object instance that you have created in the Input Pane so that it is available whenever the Client Simulator view is opened to emulate that particular connector definition. You can select the instance by name from **BO Instance** field instead of having to go through the process of creating a new request.

To save the business object instances currently listed in the **BO Instance** field, click the down arrow on the right-hand side of the title bar of the Client Simulator view and select **Edit > Save All BO's**.

The business object is saved to a file with the name of the business object instance, without any extension, in a directory named after the business object type in the `.bos` directory of the directory for the test project in which the test unit is defined. The file is saved in the same format used by Test Connector and Map Designer, so it can be used for testing in those tools as well.

Deleting business objects

To delete a business object instance from a Client Simulator view, click the down arrow on the right-hand side of the title bar of the Client Simulator view and select **Edit > Delete BO**.

Exporting business objects

You can export a business object instance to a file to conveniently archive data and to share with Technical Support. The test data file is saved with a `.bo` extension and can be used for testing with Test Connector and Map Designer as well as the Integrated Test Environment.

Exporting a request business object

Do the following to export a business object instance from the Input Pane of the Client Simulator view:

1. Select the business object instance you want to export in the **BO Instance** field.
2. Click the down arrow on the right-hand side of the title bar of the Client Simulator view and select **Edit > Export BO**.
3. Navigate to the directory in which you want to save the file and specify a name in the **File name** field.
4. Click **Save**.

Exporting a response business object

Do the following to export a business object instance from the "Response BO" window when editing a business object response in the Result Pane:

1. Click the **Save BO** button in the toolbar of the "Response BO" window.
2. Navigate to the directory in which you want to save the file and specify a name in the **File name** field.
3. Click **Save**.

Importing business objects

You can import a business object test data file that was saved from Map Designer or Test Connector, or exported from Integrated Test Environment.

Importing a request business object

Do the following to import a business object instance into the Input Pane of the Client Simulator view:

1. Click the down arrow on the right-hand side of the title bar of the Client Simulator view and select **Edit > Import BO**.
2. Navigate to and open the test data file.

Importing a response business object

Do the following to import a business object instance into the "Response BO" window when editing a business object response in the Result Pane:

1. Click the **Load BO** button in the toolbar of the "Response BO" window.
2. Navigate to and open the test data file.

Comparing business object instances

A Client Simulator view can compare two business objects of the same type and display the attributes that differ in value. You can use this function to view changes to a business object at different points in the execution of a transaction. For instance, you could compare a business object that has been sent to the server with the same business object after the server has processed it and returned it to the source connector. Do the following to compare two business object instances:

1. Select the request business object instance to be compared on the Input Pane.
2. Select the response business object to be compared on the Response Pane.
3. Click the down arrow in the right-hand corner of the title bar of the Client Simulator view and select **Reply > Compare BO's**.

Using the BO Inspector view

If you use business object tracing, Integrated Test Environment gathers information about a business object while the system processes the business object. Integrated Test Environment captures an image of the business object after it has been processed by a map and after it has been processed by a collaboration.

For example, if you are testing an interface in which a connector sends a business object request to a collaboration object which sends it to a destination connector, which processes it and then returns a response, then Integrated Test Environment captures the following business object data:

- The generic business object produced by the map that is called by the source connector when sending the business object request to InterChange Server.
- The generic business object supplied as input to the map that is called by the destination connector.
- The generic business object produced by the map that is called by the destination connector when returning the business object response to InterChange Server.
- Exception messages related to failed flows.

Figure 93 shows a BO Inspector view that has captured three input business objects, two result business objects, and an exception message.

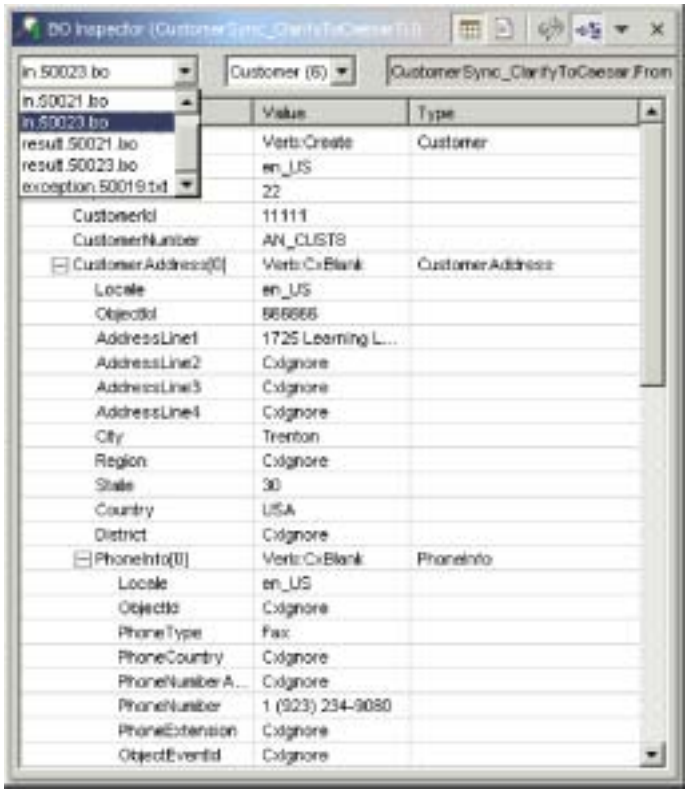

Figure 93. Business object inspector view

This section describes how to use business object tracing and BO Inspector views.

Business object trace behavior

The following statements describe the behavior related to BO Markers in the Test Unit editor and BO Inspector views as a business object is processed by an interface:

- When you send a business object request from a source connector, the BO Marker attached to the connector icon in the Test Unit editor is filled with green and the number 1 is displayed above the magnifying glass illustration in the BO Marker. This indicates that it has processed one business object.

  You can show the BO Inspector view for this BO Marker at this point and see the generic business object produced by the inbound map. It is identified with a prefix of “in” in the drop-down menu in the upper-left of the BO Inspector view.
- The business object is processed by subsequent components in the interface (such as the collaboration object) and if there are no errors then it is processed by the outbound map associated with the destination connector in the interface.

  When this occurs the BO Marker attached to the destination connector icon in the Server Context Overlay is filled with green and the number 1 is displayed above the magnifying glass illustration in the BO Marker, indicating that it, too, has processed the business object.

  You can show the BO Inspector view for this BO Marker at this point and see the generic business object that was processed by the outbound map. It is identified with a prefix of “out” in the drop-down menu in the upper-left of the BO Inspector view.
- If you reply to the request business object in the destination connector at this point the interface completes its processing. The number 1 is displayed below the magnifying glass illustration in the BO Marker, indicating that the generic object produced by the inbound map of the destination connector has been processed.

  At this point you can view the generic business object in the first BO Inspector view. It is identified with a prefix of “result” in the drop-down menu in the upper-left of the BO Inspector view. Because the same BO Inspector view also contains data for the “in” business object instance, you must select the “result” instance from the drop-down menu.
- If an error occurs during the execution of the interface a red circle is drawn around the magnifying glass of the BO Marker. You can select the error—identified with a prefix of “exception”—in the BO Inspector view associated with the BO Marker. You must change the BO Inspector view to Text View in order to view the error text.

Showing and closing BO Inspector views

The following sections describe the numerous ways you can show and close BO Inspector views.

The “Default interface position” column in Table 31 lists the views whose default positions in the interface are overlaid with the BO Inspector views in the “BO Viewer number” column.

Table 31. Default BO Inspector view positions

BO Viewer number	Default interface position
BO Viewer 1	Outline
BO Viewer 2	Properties
BO Viewer 3	Dependency

Showing BO Inspector views

Use one of the following techniques to show BO Inspector views:

- Follow the instructions in “Showing and closing BO Inspector views using the Outline view” on page 233 to show a BO Inspector view in the Outline view.
- Follow the instructions in “Showing BO Inspector views” on page 246 to show a BO Inspector view in the Test Unit view.
- Select **Integrated Test Environment > Show All BO Views** from the menu bar to show all three BO Inspector views.
- Select **Perspective > Show View > *x*:BO Inspector** from the menu bar to launch a specific BO Inspector view, where ***x*** is the number of the BO Inspector view instance you want to show.

Closing BO Inspector views

Use one of the following techniques to close all BO Inspector views:

- Follow the instructions in “Showing and closing BO Inspector views using the Outline view” on page 233 to close a BO Inspector view in the Outline view.
- Select **Integrated Test Environment > Close All BO Views** from the menu bar to close all open BO Inspector views.
- Either click the **Close** button in the right-hand side of the BO Inspector view title bar, or right-click the BO Inspector view title bar and select **Close** from the context menu.

Refreshing business object data

As the components of an interface process a business object, the counters of the BO Markers in the Test Unit editor get updated. This indicates that you can update the business object data in the BO Inspector view associated with that BO Marker to view the most recently produced business object.

Do one of the following to update the business object data in a BO Inspector view:

- Click the **Refresh BO data** button in the toolbar of the BO Inspector view.
- Click the down arrow in the right-hand corner of the title bar of the BO Inspector view and select **Refresh**.

After the business object data is refreshed, you can select the most current business object instance from the drop-down menu in the upper-left-hand corner of the BO Inspector view.

Refreshing business object markers

For information on refreshing BO Markers in the Server Context Overlay, see “Refreshing BO Markers” on page 246.

BO Inspector view styles

There are several view styles you can use with the BO Inspector view.

Using the Table view style

The Table view style presents the business object data in a table, as it appears in the Input Pane of the Client Simulator view or in the “Response BO” window.

Figure 93 on page 262 shows the BO Inspector view with the Table view style.

Do one of the following to use the Table view style:

- Click the **Show table view** button in the toolbar of the BO Inspector view.

- Click the down arrow in the right-hand side of the title bar of the BO Inspector view and select **Show Table View**.

Using the Text view style

The Text view style presents the business object data in the text markup format in which it is stored when exported from the Client Simulator view or “Response BO” window.

Figure 94 shows the BO Inspector view with the Text view style.

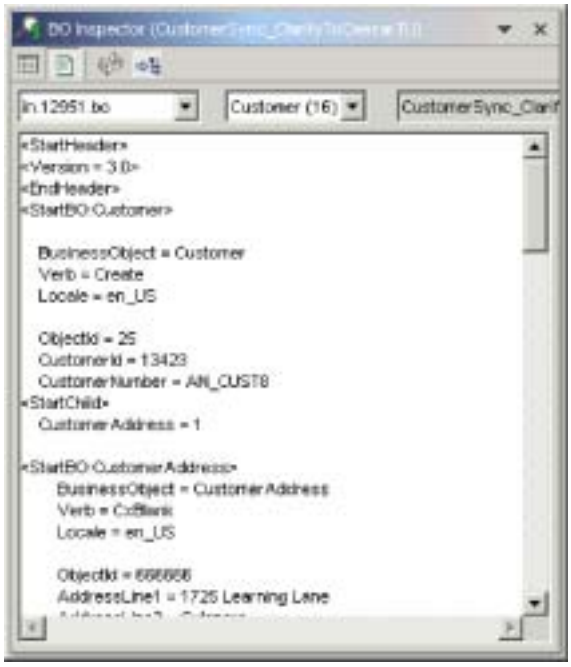

Figure 94. BO Inspector view with business object data in text view style

To view an exception message in a BO Inspector view, you must change it to the Text view style; the BO Inspector view cannot display an exception message in the Table view style.

Figure 95 on page 266 shows an exception message in the BO Inspector view with the Text view style.

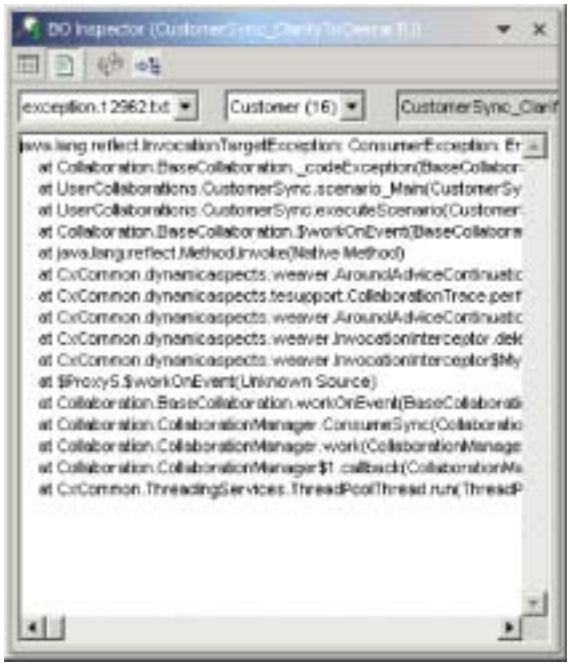

Figure 95. BO Inspector view with exception information in text view style

Do one of the following to use the Text view style:

- Click the **Show text view** button in the toolbar of the BO Inspector view.
- Click the down arrow in the right-hand side of the title bar of the BO Inspector view and select **Show Text View**.

Arranging the views vertically

You can arrange a BO Inspector view so that it displays the Text view beneath the Table view at the same time.

To arrange the BO Inspector view panes vertically, click the down arrow in the right-hand corner of the title bar of the BO Inspector view and select **Arrange Vertical** from the menu.

Figure 96 on page 267 shows the BO Inspector view panes when they have been arranged vertically.

Figure 96. BO Inspector view panes arranged vertically

Arranging the views horizontally

You can arrange a BO Inspector view so that it displays the Text view to the right of the Table view at the same time.

To arrange the BO Inspector view panes horizontally, click the down arrow in the right-hand corner of the title bar of the BO Inspector view and select **Arrange Horizontal** from the menu.

Figure 97 on page 268 shows the BO Inspector view panes when they have been arranged horizontally.

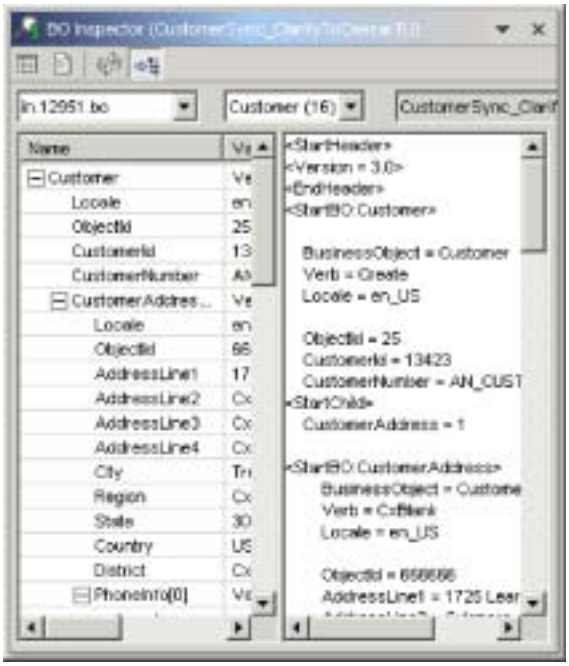

Figure 97. BO Inspector view panes arranged horizontally

Saving test data from a BO Inspector view

Although there is no built-in mechanism for saving test data in a BO Inspector view, you can use the following workaround:

1. Switch the BO Inspector view to the Text view style as described in "Using the Text view style" on page 265.
2. Select all of the text in the Text view pane.
3. Use the keyboard shortcut **Ctrl+C** to copy the text.
4. Paste the copied text into a text file and save it with a `.bo` extension.

   You can use the saved text file in Map Designer, Test Connector, and Integrated Test Environment.

Configuring business object tracing preferences

Business object tracing can be a very resource intensive task. By default it is configured to turn off after two hours, though you can change this timeout value. Furthermore, you can configure Integrated Test Environment so that it does not automatically update the counters of BO Markers as business objects are processed, which can be a resource-intensive task in high-volume transactions. Do the following to configure preferences related to business object tracing:

1. Select **Window > Preferences** from the menu bar.
2. Expand **Integrated Test Environment** and then select **Test Server**.
3. Do the following in the "BO Trace" pane to configure your business object tracing preferences:
   a. Type the number of minutes after which you want business object tracing to turn off in the **BO trace timeout** field.
   b. To have Integrated Test Environment automatically update the counters of BO Markers as business objects are processed, leave the **Enable auto update** checkbox enabled.

4. Click **OK**.

Performing a test using Integrated Test Environment

Whereas most of this chapter described subtasks you perform as part of a test or specific interface elements of the Integrated Test Environment perspective, this section describes the workflow you typically follow to perform a test. It incorporates most of the subtasks and involves most of the interface elements. It recommends the most efficient and effective techniques in situations where there are multiple ways of accomplishing a subtask. Do the following to test an interface using Integrated Test Environment:

1. Although you can deploy components to an InterChange Server instance by using Integrated Test Environment, it is recommended that you perform all deployment activities beforehand for the following reasons:
   - You avoid having to compile maps and collaboration templates as part of the testing process.
   - You can start the components prior to the testing stage as well; components must be deployed before they can be started. When you deploy connectors you must restart the server to start them, and almost any interface involves a connector, so it is typically not efficient to deploy the components for an interface as part of the testing process.
   - If you have to test several interfaces you can do a single deployment prior to testing, rather than having to make sure you oversee the proper deployment for each interface during testing.

   For information on deploying components using System Manager, see “Deploying components to a server” on page 65.

   For information on deploying components using repos_copy, see Chapter 6, “Using repos_copy,” on page 113.

   For information on deploying components by using Integrated Test Environment (which is not recommended), reference one of the following sections:
   - “Managing the repository using the Outline view” on page 231
   - “Using the Task Manager view” on page 234
   - “Managing the repository using the Test Unit view” on page 245
2. Ensure that all of the components required to test the interface are in an active state.

   To start components, use one of the following interfaces:
   - System Monitor, as described in the *System Administration Guide*
   - InterChange Server Component Management view, as described in “Managing component states in the repository” on page 74
   - Outline view, as described in “Managing component states using the Outline view” on page 231
   - “Managing component states using the Test Unit view” on page 245
3. Register the InterChange Server you want to test with as a “Local Test Server”, as described in “Registering an InterChange Server instance” on page 38.
4. Start the Integrated Test Environment perspective as described in “Starting Integrated Test Environment” on page 216.
5. Select the server you want to test with as described in “Selecting a server configuration” on page 219.

If the server instance you want to use is not listed in the dialog, try deleting it from the Server Instances view and re-registering it.

6. Configure RMI settings for your environment as described in “Configuring RMI settings” on page 220. If you do not do this properly, you will not be able to test interfaces using Integrated Test Environment.
7. Create a test project to contain the test unit. For more information, see “Creating a test project” on page 226.
8. Create a test unit for the interface you want to test. For more information, see “Creating a test unit within Integrated Test Environment” on page 227.
9. If you plan to deploy the components in the interface you are testing using Integrated Test Environment, do the following:
   - Add the user dependents for the interface as described in “Adding user dependents” on page 230.
   - Follow the instructions in “Configuring InterChange Server to start in design mode” on page 223.
10. Make sure the IBM Java Object Request Broker is started. For more information, see the *System Installation Guide for Windows or Unix*.
11. Use the Task Manager view to start the server, bind the Integrated Test Environment agent to it, and connect the Integrated Test Environment to it as described in “Using the Task Manager view” on page 234.
12. Enable the Server Context Overlay as described in “Enabling and disabling the Server Context Overlay” on page 244.
13. Show Client Simulator views for the clients in the interface.

    It is recommended that you organize the Client Simulator views in a way that makes sense to you. For instance, you might find it easiest to have the view for the source connector in position 1 in the perspective (shared with the Integrated Test Environment Navigator view), and to have the view for the destination connector in position 4 in the perspective (shared with the Properties view).

    You can do one of the following to show and organize Client Simulator views for the interface:
    - Execute the “Start connector views” task as described in Table 25 on page 235 and then drag-and-drop the views to the positions you prefer.
    - Follow the instructions in “Using, showing, and closing Client Simulator views using the Test Unit view” on page 245 to choose which Client Simulator view you want to use for each connector definition.
14. Connect the Client Simulator views to the server as described in “Connecting a Client Simulator view to the server” on page 249.

    Confirm that the clients connect to the server successfully as described in “Confirming that a client has connected to the server” on page 250.

    After you have confirmed that the clients connected to the server successfully, configure the Client Simulator view for the source connector to use the Input Pane as described in “Using the Input Pane” on page 251 and configure the Client Simulator view for the destination connector to use the Result Pane as described in “Using the Result Pane” on page 251.
15. If you want to use business object tracing, start it at this point so that the data is captured when you begin to send business objects in the next steps. For more information on starting the business object tracing task, see Table 25 on page 235.
16. Use the Outline view to confirm that the interface is ready for testing, as described in “Verifying test readiness using the Outline view” on page 232.

17. Do the following to create and send a business object request from the source connector:
    a. Create a business object instance to send as a request as described in “Creating request business objects” on page 255.
    b. Set values for the attributes of the business object instance as described in “Setting values for business object attributes” on page 257.
    c. Save the business object instance to a file to be used in subsequent tests as described in “Saving business objects” on page 260.
    d. Send the business object instance as a request as described in “Sending request business objects asynchronously” on page 255 or “Sending business object requests synchronously” on page 256 as appropriate.
18. Use the InterChange Server Console view to observe the processing of the business object as described in “Using the Integrated Test Environment Console and InterChange Server Console views” on page 242.
19. Do the following to examine the business object as different components finish processing it:
    - Show BO Inspector views for the BO Markers in the interface as described in “Showing BO Inspector views” on page 264.
    - Examine the business object instances in the BO Inspector views as described in “Using the Table view style” on page 264.
20. Edit the response business object in the Result Pane of the destination Client Simulator view as described in “Editing response business objects” on page 259.
21. Send the business object response as a reply as described in “Sending response business objects” on page 259.
22. Repeat steps 17 through 21 to test the interface again, or repeat steps 7 on page 270 through 21 to test another interface.

Chapter 13. Using Collaboration Debugger

Collaboration Debugger is a workbench perspective that allows you to specify points at which a collaboration process should pause while executing, and allows you to inspect the data of the flow being processed at those points.

To debug a collaboration, you must do a number of tasks such as attach Collaboration Debugger to a collaboration object, configure breakpoints, manage the progress of the flow through the collaboration, and inspect the data of the flow. The following characteristics describe the use of Collaboration Debugger:

- You only have to perform some of the tasks a single time. For instance, you typically only have to attach Collaboration Debugger to the collaboration object once.
- You typically perform some tasks multiple times. For instance, you usually manage the progress of the flow multiple times during the course of its processing.
- You might not perform some tasks during the processing of every flow.

Due to the non-linear nature of debugging a collaboration, the sections of this chapter describe the interfaces of the perspective or certain tasks that you perform in the order in which you most commonly use the interfaces or perform the tasks. "Debugging a collaboration" on page 291 describes the workflow of debugging a collaboration to provide a more linear guide to the interfaces and tasks.

This chapter contains the following sections:

Starting Collaboration Debugger

To start Collaboration Debugger, do the following:

1. Select **Start > Programs > IBM WebSphere InterChange Server > IBM WebSphere Business Integration Toolset > Administrative > System Manager**.
2. Select **Window > Open Perspective > Other** from the menu bar.
3. Select `Collaboration Debugger` from the list of perspectives and then click **OK**.

   The workbench starts and appears. Figure 2 on page 35 shows the Collaboration Debugger perspective and "Collaboration Debugger interface" on page 275 describes the interface and its elements.

Configuring Collaboration Debugger view preferences

You can configure Collaboration Debugger so that its interfaces either open in the perspective you are currently using, or in Collaboration Debugger as designed. The benefit of having the interfaces open in the perspective you are currently using is that you do not have to navigate between perspectives to use the Collaboration Debugger views while also performing tasks like deploying components or sending events. The benefit of having the interfaces open in Collaboration Debugger is that you reduce the number of interfaces that have to share the same positions in the workbench.

Do the following to configure your preferences for the Collaboration Debugger views:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Select **Collaboration Debugger**.

   Figure 98 shows the view preferences for Collaboration Debugger.

Figure 98. Collaboration Debugger view preferences

3. Enable the **Display in Debugger Perspective** radio button to have the Collaboration Debugger views open in Collaboration Debugger itself.

Enable the **Display in Current Perspective** radio button to have the Collaboration Debugger views open in the perspective you currently have active.

4. Click **OK**.

Collaboration Debugger interface

The Collaboration Debugger perspective has several views and one editor in the default configuration it opens with. Figure 99 shows the default Collaboration Debugger perspective.

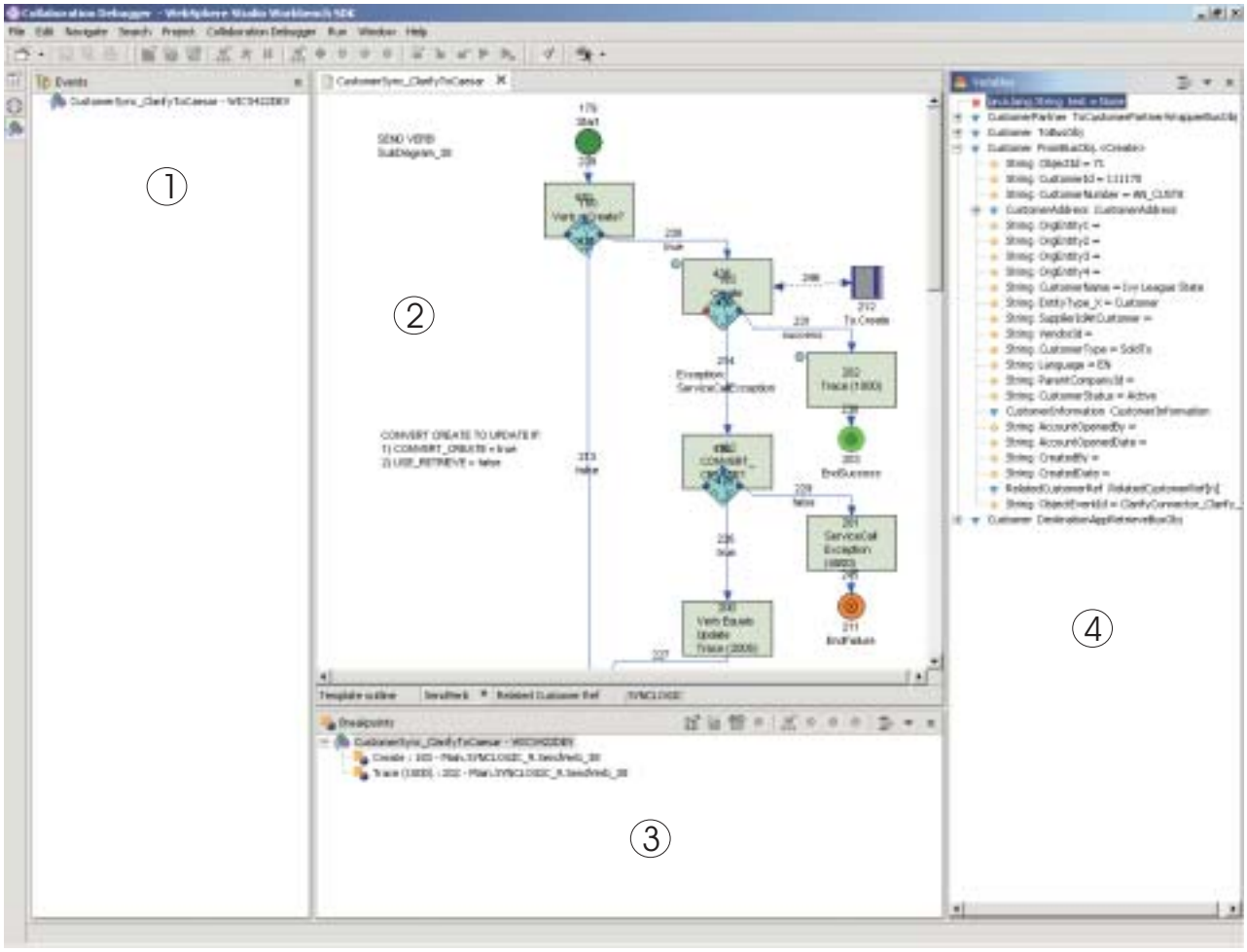

Figure 99. Collaboration Debugger perspective

Table 32 on page 275 identifies the interface elements of the Collaboration Debugger perspective, indicated by the numbers in Figure 99.

Table 32. Collaboration Debugger perspective interface elements

Interface element number	Interface element name
1	“Events view” on page 276
2	“Collaboration template editor” on page 276
3	“Breakpoints view” on page 277

Table 32. Collaboration Debugger perspective interface elements (continued)

Interface element number	Interface element name
4	"Variables view" on page 278

Events view

The "Events" view lists the flows that are delivered for processing to the collaboration object you are debugging. It provides operations to affect the processing of the flow and to locate the node that is currently executing.

Figure 100 shows the "Events" view.

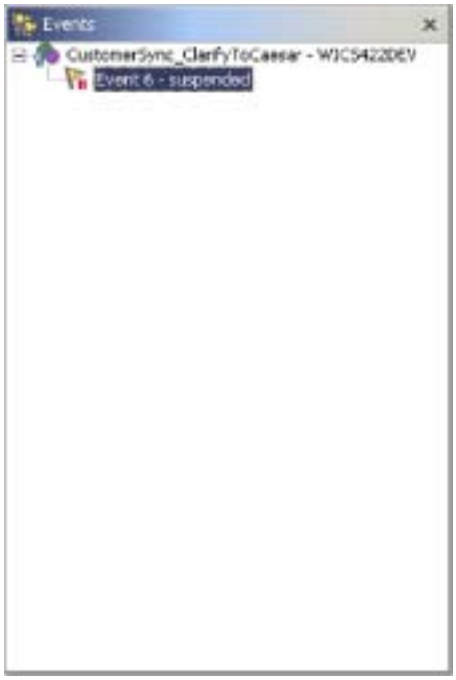

Figure 100. Events view

For more information about working with events and using the "Events" view, see "Working with events" on page 284.

Collaboration template editor

The "Collaboration template" editor is a read-only editor that displays the diagrams of the collaboration template upon which the collaboration object you are debugging is based. It allows you to view the design of the business process and perform many of the tasks associated with debugging a collaboration.

Figure 101 on page 277 shows the "Collaboration template" editor.

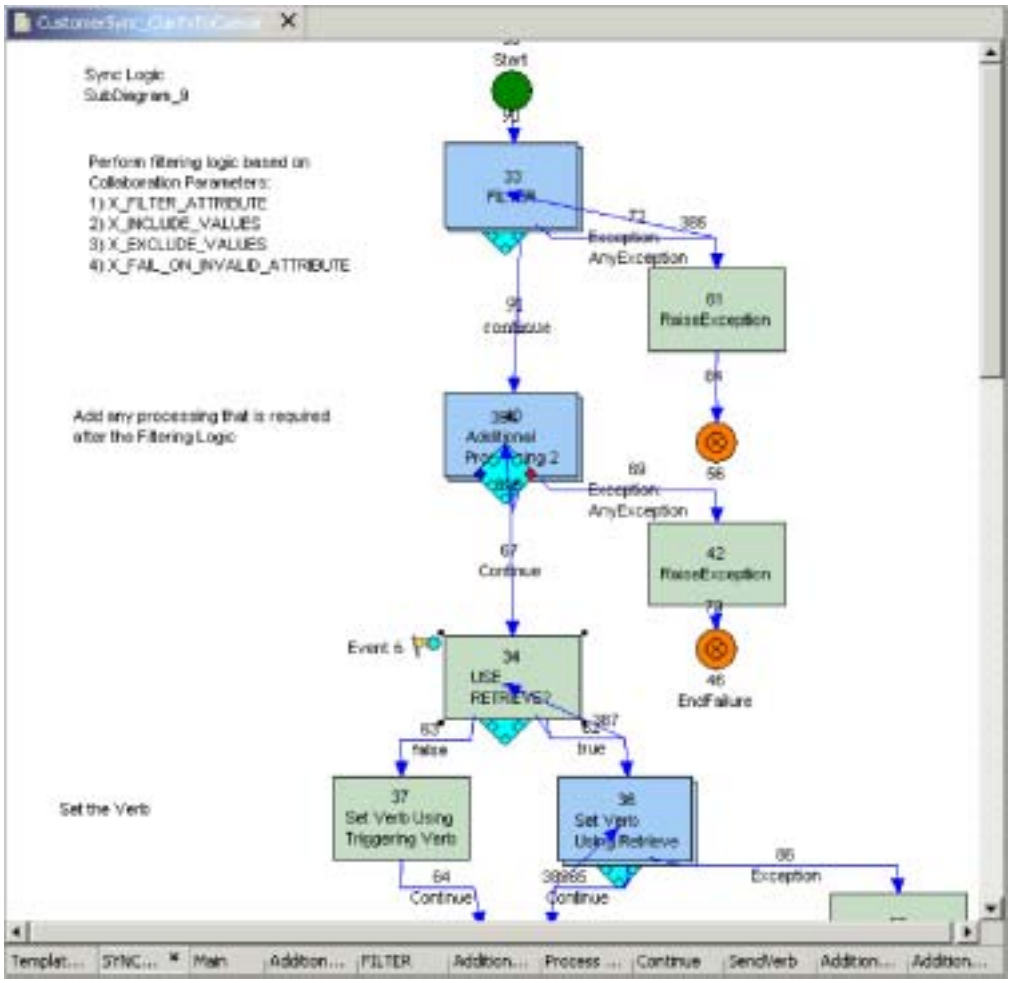

Figure 101. Collaboration template editor

The following sections contain more information about the “Collaboration template” editor and the tasks you can perform using it:

- “Using the collaboration template editor” on page 282
- “Working with breakpoints” on page 285
- “Performing debugging operations” on page 288

Breakpoints view

The “Breakpoints” view lets you manage breakpoints in the collaboration object and manage the processing of a flow as it interacts with those breakpoints.

Figure 102 on page 278 shows the “Breakpoints” view.

Figure 102. Breakpoints view

For more information about the tasks you typically perform using the "Breakpoints" view, see the following sections:

- "Working with breakpoints" on page 285
- "Performing debugging operations" on page 288

Variables view

The "Variables" view displays information about the flow being processed as it pauses in response to debugging operations. It lists all the variables of `BusObj` or Java primitive types that are declared in the "Declarations" section of the collaboration template. As you perform debugging operations to manage a flow through the collaboration, the "Variables" view updates the variables it lists with their values at the node being executed at the time.

Figure 103 on page 279 shows the "Variables" view.

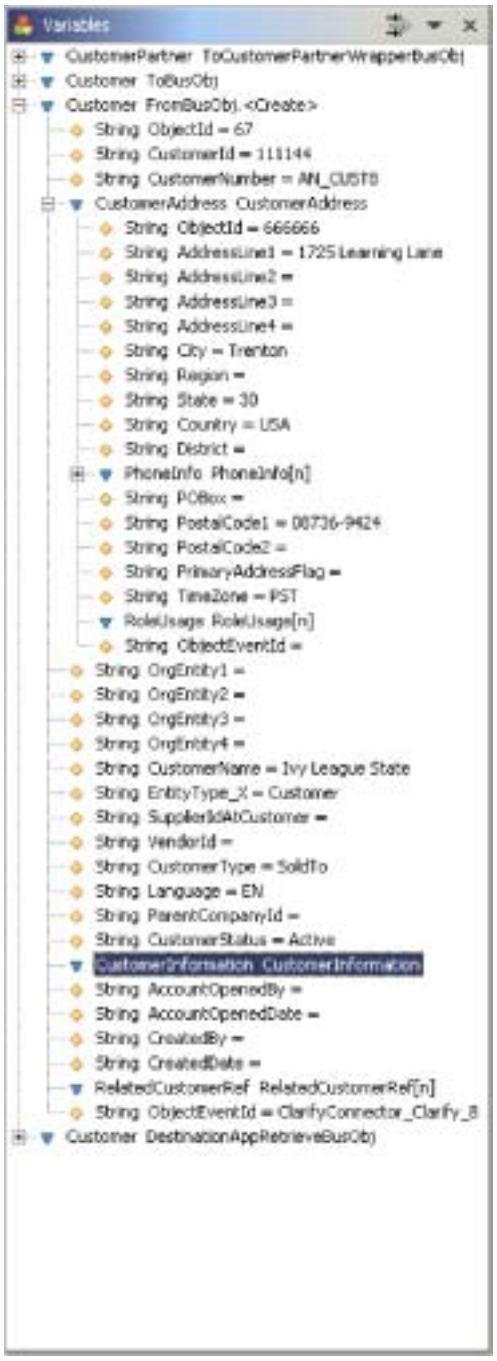

Figure 103. Variables view

For more information about the “Variables” view and the tasks that affect its contents, see the following sections:

- “Working with events” on page 284
- “Performing debugging operations” on page 288
- “Working with variables” on page 290

Attaching and detaching Collaboration Debugger

You must attach Collaboration Debugger to a collaboration object to start a debugging session, set breakpoints, and manage a flow through the business process.

The collaboration object must be active to attach Collaboration Debugger to it. For more information about component states and how to change them, see the *System Administration Guide*.

When you are finished debugging a collaboration object, detach Collaboration Debugger to resume the normal processing of flows. If you stop a collaboration object then Collaboration Debugger automatically detaches from it.

Attaching Collaboration Debugger

You can attach Collaboration Debugger to a collaboration object in either System Manager or Integrated Test Environment, as described in the following sections:

- "Attaching Collaboration Debugger in System Manager"
- "Attaching Collaboration Debugger in Integrated Test Environment"

Attaching Collaboration Debugger in System Manager

Do the following to attach Collaboration Debugger to a collaboration object in System Manager:

1. In the InterChange Server Component Management view, expand the server instance in which the collaboration object exists, then expand the Collaboration Objects folder.
2. Select the collaboration objects that you want to attach Collaboration Debugger to. You can use standard Windows selection techniques to select multiple user projects for deployment at the same time, such as the following:
   - Hold down **Shift** to select contiguous items.
   - Hold down **Ctrl** to select non-contiguous items.
3. Right-click one of the selected collaboration objects and choose **Attach Debugger** from the context menu

Attaching Collaboration Debugger in Integrated Test Environment

To attach Collaboration Debugger to a collaboration object in Integrated Test Environment, right-click the center icon for the collaboration object in the Test Unit view and choose **Attach Debugger** from the context menu.

For more information about the Test Unit view, see "Using the Test Unit view" on page 243.

Attaching Collaboration Debugger in Collaboration Debugger

Do one of the following to attach Collaboration Debugger to a collaboration object in Collaboration Debugger itself:

- Right-click anywhere in the "Collaboration template" editor and choose **Attach Debugger** from the context menu
- Right-click the collaboration object in the "Breakpoints" view or the "Events" view and choose **Attach Debugger** from the context menu
- Select the collaboration object in the "Breakpoints" view or the "Events" view and select **Collaboration Debugger > Attach Debugger** from the menu bar

- Select the collaboration object in the “Breakpoints” view and click **Attach Debugger** in the toolbar of the “Breakpoints” view
- Select the collaboration object in the “Breakpoints” view, click the drop-down arrow in the upper-right-hand corner of the title bar, and choose **Attach Debugger**

Detaching Collaboration Debugger

You can detach Collaboration Debugger from a collaboration object in either System Manager or Integrated Test Environment, as described in the following sections:

- “Detaching Collaboration Debugger in System Manager”
- “Detaching Collaboration Debugger in Integrated Test Environment”

Detaching Collaboration Debugger in System Manager

Do the following to detach Collaboration Debugger from a collaboration object in System Manager:

1. In the InterChange Server Component Management view, expand the server instance in which the collaboration object exists, then expand the Collaboration Objects folder.
2. Select the collaboration objects that you want to detach Collaboration Debugger from. You can use standard Windows selection techniques to select multiple user projects for deployment at the same time, such as the following:
   - Hold down **Shift** to select contiguous items.
   - Hold down **Ctrl** to select non-contiguous items.
3. Right-click one of the selected collaboration objects and choose **Detach Debugger** from the context menu

Detaching Collaboration Debugger in Integrated Test Environment

To detach Collaboration Debugger from a collaboration object in Integrated Test Environment, right-click the center icon for the collaboration object in the Test Unit view and choose **Detach Debugger** from the context menu.

For more information about the Test Unit view, see “Using the Test Unit view” on page 243.

Detaching Collaboration Debugger in Collaboration Debugger

Do one of the following to detach Collaboration Debugger from a collaboration object in Collaboration Debugger itself:

- Right-click anywhere in the “Collaboration template” editor and choose **Detach Debugger** from the context menu
- Right-click the collaboration object in the “Breakpoints” view or the “Events” view and choose **Detach Debugger** from the context menu
- Select the collaboration object in the “Breakpoints” view or the “Events” view and select **Collaboration Debugger > Detach Debugger** from the menu bar
- Select the collaboration object in the “Breakpoints” view and click **Detach Debugger** in the toolbar of the “Breakpoints” view
- Select the collaboration object in the “Breakpoints” view, click the drop-down arrow in the upper-right-hand corner of the title bar, and choose **Detach Debugger**

Removing detached collaboration objects from Collaboration Debugger

Do one of the following to remove all collaboration objects you have detached from Collaboration Debugger so that they are not displayed in the views:

- Right-click anywhere in the "Events" or "Breakpoints" views and choose **Remove All Detached** from the context menu
- Click **Remove All Detached** in the toolbar of the "Breakpoints" view
- Click the drop-down arrow in the upper-right-hand corner of the title bar of the "Breakpoints" view, and choose **Remove All Detached**

Using the collaboration template editor

The "Collaboration template" editor is a read-only editor that displays the diagrams of the collaboration template upon which the collaboration object you are debugging is based. It allows you to view the design of the business process and perform many of the tasks associated with debugging a collaboration.

The "Collaboration template" editor allows you to work with diagrams as described in "Opening and closing diagrams" and node labels and IDs as described in "Showing and hiding node IDs and labels."

In addition to these sections, which describe tasks that are specific to the "Collaboration template" editor, the following sections describe tasks you can perform using multiple interfaces:

- "Using the collaboration template editor"
- "Working with breakpoints" on page 285
- "Performing debugging operations" on page 288

Opening and closing diagrams

To manage breakpoints for nodes that belong to subdiagram and iterator nodes in a collaboration template, you must open the diagram for the node.

Do one of the following to open the diagram for a subdiagram or iterator node:

- Select the node and press **Enter**
- Double-click the node
- Right-click the node and choose **Open Diagram** from the context menu.

To close the diagram for a subdiagram or iterator node, click the **X** in the upper-right-hand corner of the tab for the diagram.

Showing and hiding node IDs and labels

The unique IDs and labels assigned to the nodes and links in a collaboration template diagram make it easy to identify them and to understand the business process defined by the template. Having them displayed in the Collaboration Debugger can therefore facilitate your troubleshooting experience.

To show the unique identifiers in a diagram, right-click anywhere in the editor and select **Show uniqueIDs** from the context menu.

To hide the unique identifiers in a diagram, right-click anywhere in the editor and select **Hide uniqueIDs** from the context menu.

To show the labels in a diagram, right-click anywhere in the editor and select **Show labels** from the context menu.

To show the unique identifiers in a diagram, right-click anywhere in the editor and select **Show labels** from the context menu.

Configuring Collaboration template editor preferences

Do the following to configure your preferences for the Collaboration template editor:

1. Select **Window > Preferences** from the menu bar of the workbench.
2. Expand **Collaboration Debugger**.
3. Select **Diagram Colors**.

   Figure 104 shows the Collaboration template diagram color preferences.

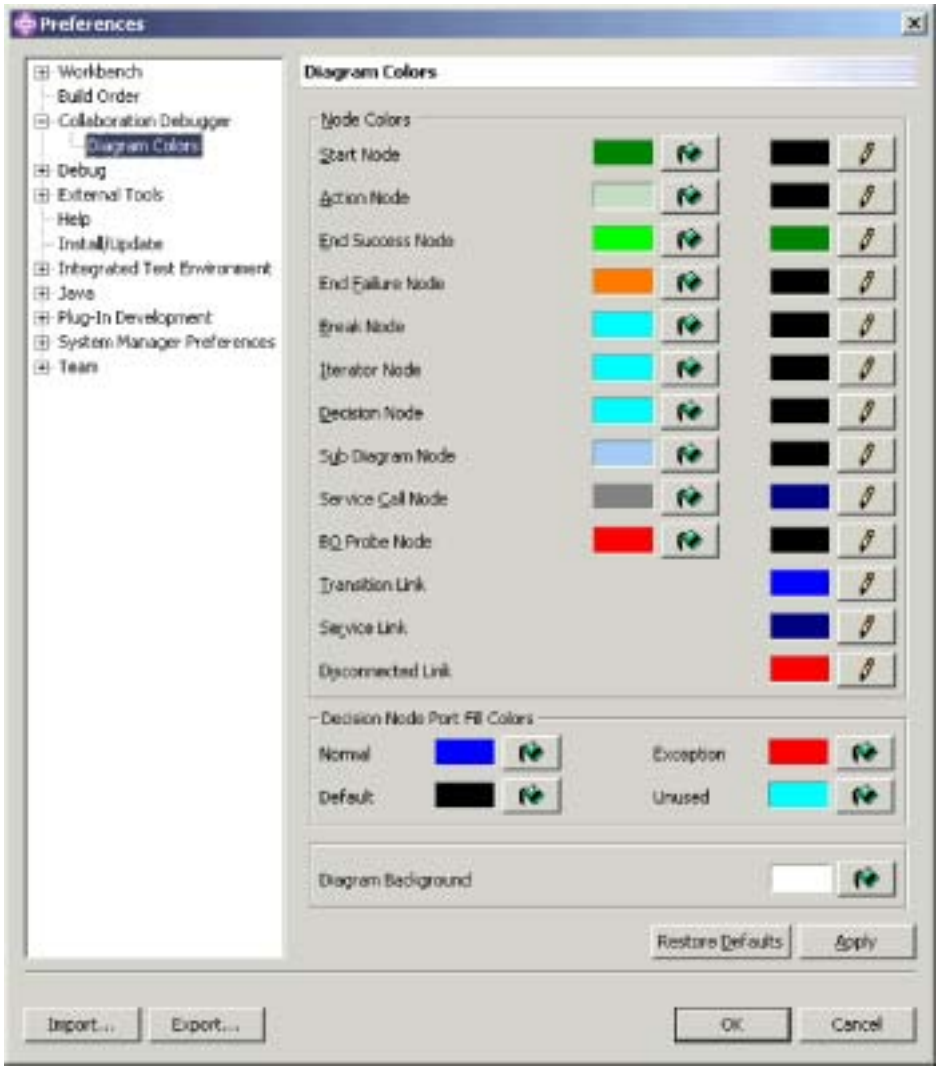

Figure 104. Collaboration template diagram color preferences

4. Click the paint bucket icons associated with the different interface elements to set the desired fill color for those elements.

Click the pencil icons associated with the different interface elements to set the desired line color for those elements.

At the “Color” dialog, select the color you would like the Test Unit view to use for the background of that component type, then click **OK**.

5. Click **OK**.

Working with events

Events are the flows that trigger the execution of a collaboration object. Collaborations execute business process logic to manage event data, manipulating it and coordinating it among enterprise software resources. You use Collaboration Debugger primarily to pause the processing of an event so that you can inspect its data at significant points in the business logic.

Displaying an event

When a new flow is delivered for processing to a collaboration object that you are debugging it is assigned a number and is listed beneath the collaboration object in the “Events” view.

Figure 105 shows an event that has been received by a collaboration object and is displayed in the “Events” view.

Figure 105. An event displayed in the Events view

When you display an event, Collaboration Debugger gives focus to the node where processing is paused in the “Collaboration template” editor; black squares appear in each corner of the node, indicating that it is selected, and a flag icon labeled with the event instance is displayed in the upper-left-hand corner of the node. Figure 106 shows a node where processing of the event is paused in the Collaboration template editor.

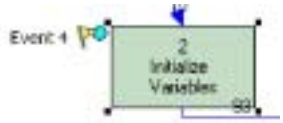

Figure 106. An event displayed in the Collaboration template editor

If there are active breakpoints set in the collaboration, the flow suspends processing while it waits for you to perform a debugging operation, and a flag icon is drawn on the node where processing is broken to represent the event as well.

You display an event to locate it in the collaboration diagram and to populate the “Variables” view with the current data in the flow.

The “Variables” view dynamically updates as you perform debugging operations, so you typically only need to display an event in the following situations:

- When the collaboration object first receives the flow for processing

- If you have navigated elsewhere in Collaboration Debugger (to display a breakpoint, for example) and need to locate the event again

Do one of the following to display an event:

- Right-click the event in the "Events" view and choose **Display** from the context menu
- Select the event in the "Events" view and select **Collaboration Debugger > Display** from the menu bar

Suspending an event

When you run an event, as described in "Running an event" on page 290, the flow of execution proceeds as it would normally, without pausing on any breakpoints. If you decide that you want to resume debugging the flow, you can suspend the event.

Once you have instructed Collaboration Debugger to run an event, it typically processes too quickly for you to suspend it, though service calls provide this opportunity. When a business object is sent out a service call, the business process blocks until a response is received; this provides enough time to suspend the flow prior to replying to the request, so that debugging resumes when the response is received by the collaboration.

Do one of the following to suspend an event:

- Right-click the event in the "Events" view and choose **Suspend** from the context menu
- Select the event in the "Events" view and select **Collaboration Debugger > Suspend** from the menu bar

Running an event

For more information on running an event, see "Running an event" on page 290.

Working with breakpoints

Breakpoints are instructions that Collaboration Debugger uses to pause the execution of a collaboration object. You set breakpoints at node in the business process where you want to inspect the data of a flow to observe its transformation and troubleshoot problems.

You can set breakpoints on the following types of nodes:

- Action node
- Subdiagram node
- Iterator node
- End success node
- End failure node
- Iterator node
- Break node

You can not set a breakpoint on a decision node.

For more information about the different types of nodes in a collaboration template, see the *Collaboration Development Guide.*

The following sections describe how you can arrange breakpoints in a collaboration you are debugging:

- "Setting breakpoints"
- "Displaying breakpoints"
- "Removing breakpoints"
- "Disabling breakpoints" on page 287
- "Enabling breakpoints" on page 287

After you have arranged breakpoints in the collaboration object, use the instructions in "Performing debugging operations" on page 288 to manage the processing of the flow.

Setting breakpoints

Do one of the following to set a breakpoint on a supported node:

- Right-click the node and choose **Set breakpoint(s)** from the context menu
- Select the node and click **Set breakpoint(s)** in the toolbar of Collaboration Debugger
- Select the node and select **Collaboration Debugger > Set breakpoint(s)** from the menu bar
- Double-click a node that does not have a breakpoint set on it already

Displaying breakpoints

Do one of the following to select and give focus to a node on which a breakpoint has been set:

- Right-click the breakpoint in the "Breakpoints" view and choose **Display** from the context menu
- Double-click the breakpoint in the "Breakpoints" view
- Select the breakpoint in the "Breakpoints" view and press **Enter**
- Select the breakpoint in the "Breakpoints" view and click **Display** in the toolbar of the "Breakpoints" view
- Select the breakpoint in the "Breakpoints" view, click the drop-down arrow in the upper-right-hand corner of the title bar, and choose **Display**
- Select the breakpoint in the "Breakpoints" view and click **Display** in the toolbar of Collaboration Debugger
- Select the breakpoint in the "Breakpoints" view and select **Collaboration Debugger > Display breakpoint(s)** from the menu bar

Removing breakpoints

Do one of the following to remove a breakpoint:

- Right-click the node on which the breakpoint is set in the "Collaboration template" editor and choose **Remove breakpoint(s)** from the context menu
- Right-click the breakpoint in the "Breakpoints" view and choose **Remove** from the context menu
- Select the breakpoint in the "Breakpoints" view and click **Remove** in the toolbar of the "Breakpoints" view
- Select the breakpoint in the "Breakpoints" view, click the drop-down arrow in the upper-right-hand corner of the title bar, and choose **Remove**
- Select the node on which the breakpoint is set in the "Collaboration template" editor and click **Remove breakpoint(s)** in the toolbar of Collaboration Debugger

- Select the node on which the breakpoint is set in the “Collaboration template” editor and select **Collaboration Debugger > Remove breakpoint(s)** from the menu bar
- Double-click a node that has a breakpoint set on it

Disabling breakpoints

Do one of the following to disable a breakpoint so that execution no longer pauses at the specified node, but so that you can re-activate it if necessary:

- Right-click the node on which the breakpoint is set in the “Collaboration template” editor and choose **Disable breakpoint(s)** from the context menu
- Right-click the breakpoint in the “Breakpoints” view and choose **Disable** from the context menu
- Select the breakpoint in the “Breakpoints” view and click **Disable** in the toolbar of the “Breakpoints” view
- Select the breakpoint in the “Breakpoints” view, click the drop-down arrow in the upper-right-hand corner of the title bar, and choose **Disable**
- Select the node on which the breakpoint is set in the “Collaboration template” editor and click **Disable breakpoint(s)** in the toolbar of Collaboration Debugger
- Select the node on which the breakpoint is set in the “Collaboration template” editor and select **Collaboration Debugger > Disable breakpoint(s)** from the menu bar

Enabling breakpoints

Do one of the following to enable a breakpoint that has been disabled:

- Right-click the node on which the breakpoint is set in the “Collaboration template” editor and choose **Enable breakpoint(s)** from the context menu
- Right-click the breakpoint in the “Breakpoints” view and choose **Enable** from the context menu
- Select the breakpoint in the “Breakpoints” view and click **Enable** in the toolbar of the “Breakpoints” view
- Select the breakpoint in the “Breakpoints” view, click the drop-down arrow in the upper-right-hand corner of the title bar, and choose **Enable**
- Select the node on which the breakpoint is set in the “Collaboration template” editor and click **Enable breakpoint(s)** in the toolbar of Collaboration Debugger
- Select the node on which the breakpoint is set in the “Collaboration template” editor and select **Collaboration Debugger > Enable breakpoint(s)** from the menu bar

Filtering breakpoints

You can filter the “Breakpoints” view to only display breakpoints set on certain types of nodes. Do the following to filter the “Breakpoints” view:

1. Do one of the following to display the “Breakpoints Filter” dialog:
   - Click **Filter** in the title bar of the “Breakpoints” view
   - Click the down-facing arrow in the upper-right-hand corner of the title bar of the “Breakpoints” view and select **Filter** from the drop-down menu

   Collaboration Debugger displays the “Breakpoints Filter” dialog, as shown in Figure 108 on page 291:

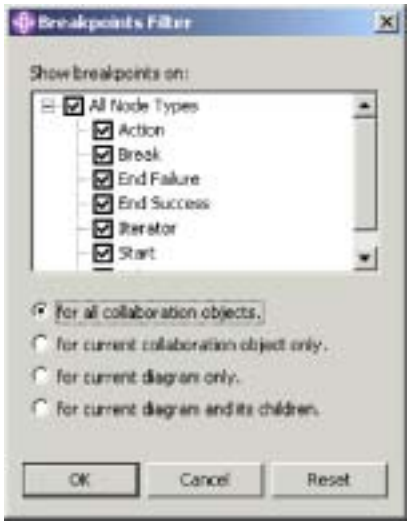

Figure 107. Breakpoints filter

2. Use the following techniques to configure your filter settings:
   - Enable the **All Node Types** checkbox to display all supported node types
   - Enable only the checkboxes for specific node types to display only those types
   - Click **Reset** to apply the default filter settings
3. Enable one of the following radio buttons to specify how broadly the filter should be applied:
   - **for all collaboration objects**
   - **for current collaboration object only**
   - **for current diagram only**
   - **for current diagram and its children**
4. Click **OK**.

Performing debugging operations

Once you have arranged breakpoints in the collaboration object you are debugging, you perform debugging operations to manage the flow of the event through the breakpoints. The following operations are available:

- "Stepping over"
- "Stepping into" on page 289
- "Stepping out" on page 289
- "Running" on page 289
- "Running to a specific node" on page 290
- "Running an event" on page 290

Stepping over

When you perform a "step over" operation, InterChange Server resumes the paused flow so that the node on which the breakpoint is set executes, and then pauses the flow at the next node.

Do one of the following to step over a node where execution is broken:

- Right-click anywhere in the “Collaboration template” editor and choose **Step Over** from the context menu
- Give focus to the “Collaboration template” editor and click **Step Over** in the toolbar
- Give focus to the “Collaboration template” editor and select **Collaboration Debugger > Step Over** from the menu bar
- Give focus to the “Collaboration template” editor and use the keyboard shortcut **F6**

Stepping into

When you perform a “step into” operation on a subdiagram or iterator node, InterChange Server resumes the paused flow. Execution proceeds until the flow reaches the start node in the subdiagram or iterator and then pauses.

Do one of the following to step into a subdiagram or iterator where execution is broken:

- Right-click anywhere in the “Collaboration template” editor and choose **Step Into** from the context menu
- Give focus to the “Collaboration template” editor and click **Step Into** in the toolbar
- Give focus to the “Collaboration template” editor and select **Collaboration Debugger > Step Into** from the menu bar
- Give focus to the “Collaboration template” editor and use the keyboard shortcut **F5**

Stepping out

When you perform a “step out” operation on a node within a subdiagram or iterator, InterChange Server resumes the paused flow. Execution proceeds until the subdiagram or iterator completes, or until the flow reaches an active breakpoint within the subdiagram or iterator.

Do one of the following to step out of a subdiagram or iterator where execution is broken:

- Right-click anywhere in the “Collaboration template” editor and choose **Step Out** from the context menu
- Give focus to the “Collaboration template” editor and click **Step Out** in the toolbar
- Give focus to the “Collaboration template” editor and select **Collaboration Debugger > Step Out** from the menu bar
- Give focus to the “Collaboration template” editor and use the keyboard shortcut **F7**

Running

When you perform a “run” operation, InterChange Server resumes the paused flow and allows processing to continue either until completion of the business process or to a node with an active breakpoint set on it.

Do one of the following to run a paused flow:

- Right-click anywhere in the “Collaboration template” editor and choose **Run** from the context menu
- Give focus to the “Collaboration template” editor and click **Run** in the toolbar

- Give focus to the “Collaboration template” editor and select **Collaboration Debugger > Run** from the menu bar
- Give focus to the “Collaboration template” editor and use the keyboard shortcut **F8**

Running to a specific node

When you perform a “run to this node” operation on a flow, InterChange Server resumes the paused flow and allows processing to continue either to the node on which you performed the operation, or to a node with an active breakpoint set on it.

Do one of the following to run a paused flow to a specific node:

- Right-click the node to which you want execution to run and choose **Run to this node** from the context menu
- Select the node to which you want execution to run and click **Run to** in the toolbar
- Select the node to which you want execution to run and select **Collaboration Debugger > Run to** from the menu bar

Running an event

When you perform a “run event” operation, InterChange Server resumes the paused flow and allows processing to continue, bypassing any breakpoints in the business process even if the breakpoints are active.

Do one of the following to run an event:

- Right-click the event in the “Events” view and choose **Run** from the context menu
- Select the event in the “Events” view and select **Collaboration Debugger > Run Event(s)** from the context menu
- Select the event in the “Events” view and click **Run Event(s)** in the toolbar

Working with variables

The “Variables” view displays the value of any variable that is declared in the “Declarations” tab of the collaboration template definition. This includes the business objects that are automatically declared by Process Designer to be associated with each port that you create on the “Ports and Triggering Events” tab, as well as any variables you add to the “Declarations” tab yourself.

When you display an event as described in “Displaying an event” on page 284 or manage a flow as described in “Performing debugging operations” on page 288, the “Variables” view updates the variable information it shows. You can expand any variables of `BusObj` type to view their attributes.

There is currently no way to display the `triggeringBusObj` variable. This is a system-declared variable that references the original flow that triggered execution of the collaboration object, and is typically very significant. If you want to maintain a record of the values of the `triggeringBusObj` variable, you can declare a new `BusObj` variable in the “Declarations” section of the collaboration template, then initialize it to the value of the `triggeringBusObj` variable in the first node of the scenario.

Furthermore, many delivered collaboration templates manipulate a variable named `processingBusObj` and leave the `triggeringBusObj` variable untouched. The `processingBusObj` variable is typically declared at the scenario level rather than the template level, though, so it is not displayed in the "Variables" view. To have the `processingBusObj` variable available in the "Variables" view, move the declaration for the variable from the scenario definition to the template definition.

You can filter the "Variables" view to only display variables of certain types. Do the following to filter the "Variables" view:

1. Do one of the following to display the "Variables Filter" dialog:
   - Click **Filter** in the title bar of the "Variables" view
   - Click the down-facing arrow in the upper-right-hand corner of the title bar of the "Variables" view and select **Filter** from the drop-down menu

   Collaboration Debugger displays the "Variables Filter" dialog, as shown in Figure 108:

Figure 108. Variables filter

2. Use the following techniques to configure your filter settings:
   - Enable the **All Types** checkbox to display all supported variable types
   - Enable only the checkboxes for specific variable types to display only those types
   - Click **Reset** to apply the default filter settings
3. Click **OK**.

Debugging a collaboration

Use the following workflow as a guideline to efficiently debug a collaboration object:

1. Create the collaboration template, create a collaboration object based on it, deploy both to the InterChange Server instance, and make sure that the collaboration object is started.
2. Start the Collaboration Debugger perspective as described in "Starting Collaboration Debugger" on page 273.
3. Attach Collaboration Debugger to the collaboration object as described in "Attaching Collaboration Debugger" on page 280.
4. Set breakpoints in desired locations, as described in "Working with breakpoints" on page 285.

You will usually find it most useful to set breakpoints on nodes where the flow is being transformed, where the flow is being sent out of the collaboration, and where problematic pieces of logic are located.

5. Create and send an event such that the collaboration object will be triggered, using either Test Connector or Integrated Test Environment. For more information on Test Connector, see Chapter 11, "Using Test Connector," on page 201.

   For more information on Integrated Test Environment, see Chapter 12, "Using Integrated Test Environment," on page 215.
6. Display the submitted event as described in "Displaying an event" on page 284.
7. Manage the flow across the breakpoints you have configured by using the techniques described in "Performing debugging operations" on page 288.
8. Use the "Variables" view to watch the data of the event change as it is processed by the collaboration, as described in "Working with variables" on page 290.
9. Repeat steps 4 on page 291 through 8 to modify the breakpoints and process flows with different data.

Chapter 14. Performance tuning

This chapter describes how to implement various techniques and configurations to improve system performance. It contains the following sections:

Implementing concurrent processing of event-triggered flows

You can configure collaboration objects and connector controllers to process multiple event-triggered flows at the same time. This can significantly improve the performance for event-triggered interfaces.

Concurrent event-triggered flow processing in Collaborations

Collaborations can be configured to process multiple event-triggered flows concurrently. System throughput and response time to event processing improve when concurrent event processing is properly used (see tip below). By default, a collaboration processes one event-triggered flow at a time.

When a collaboration is processing concurrent event-triggered flows, the collaboration identifies dependencies among those flows and processes them in the order that they were sent from the connector controller. Concurrent processing of event-triggered flows is performed on flows that do not have data conflicts, while flows with data conflicts are processed in the order received.

To configure a collaboration to handle multiple event-triggered flows, see "Maximum number of concurrent events" on page 178.

Tip: Concurrent processing of event-triggered flows in collaborations requires additional system resources. To maximize performance, ensure that system resources used to handle concurrent events are not idle. For example, do not set the value for the **Maximum number of concurrent events** to 10 if the number of events in the collaboration queue never reaches 10.

If a collaboration's inbound ports are bound only to receive external calls through the Access Interface, and are not bound to any connectors, you can improve performance by setting the value of **Maximum number of concurrent events** to zero. But do not set this value to zero if the collaboration is being used for bidirectional exchanges with connectors.

Concurrent event-triggered flow processing in collaboration-object groups

Each collaboration in a collaboration-object group can be configured independently for processing a number of concurrent event-triggered flows. It is recommended

that you set the same value for the number of concurrent event-triggered flows for all of the collaborations in a group, so that a collaboration with a low concurrency rate does not become a bottleneck.

Concurrent event-triggered flow processing in connector controllers

Connector controllers can be configured to process multiple event-triggered flows concurrently.

Event-flow processing performance improves when a connector is configured to process triggered event flows concurrently. This is because multiple business objects can be transformed in mapping at the same time.

To configure concurrent event-triggered flow processing for connector controllers, you set the **ConcurrentEventTriggeredFlows** property to the maximum number of flows you want processed at the same time. For more information on this property, see “ConcurrentEventTriggeredFlows” on page 145. For more information on using Connector Configurator to set connector properties, see in Chapter 7, “Configuring connectors,” on page 125 general.

Tip: If the **ConcurrentEventTriggeredFlows** property is configured to a value greater than 1, the connector controller maintains the same order of events that it received from the application. Concurrent processing of event-triggered flows in connectors requires additional system resources. To maximize performance, ensure that the system resources that handle concurrent events are not idle. For example, do not set the **ConcurrentEventTriggeredFlows** property to a value of 10 if the number of events arriving at InterChange Server for the connector never reaches 10. Use the server statistics window to determine the number of events that are in connector’s queue by monitoring the MQ queue depth for the connector. Monitoring this statistic can help you set the value for the **ConcurrentEventTriggeredFlows** property.

If a connector is being used only as a destination, you can improve performance by setting the value of **ConcurrentEventTriggeredFlows** to zero. But do not set this value to zero if the connector is being used in bidirectional exchanges with a collaboration.

Implementing concurrent processing of requests by connector agents

Connector agents that were designed in Java to be multi-threaded can perform concurrent processing of multiple threads without additional configuration of the connector agent. Additionally, some connectors that were not designed for multi-threading can be configured to use a feature called Connector Agent Parallelism (CAP), which enables a connector to run as multiple connector processes to handle multiple requests. Connector Agent Parallelism instantiates multiple processes in parallel from a single connector agent master process, making it possible to process flows concurrently rather than serially.

Use of Connector Agent Parallelism in the IBM WebSphere InterChange Server business integration system is configurable through the “Resources” tab in Connector Configurator. Because the use of concurrent processing for connector agents may bring performance benefits in some circumstances but not in others, the default behavior for connector agents is to use serial processing. Before changing the default setting, you should consult the IBM Business Integration Adapters documentation for the specific connector, analyze the application itself to

determine whether its architecture or requirements would prevent effective use of concurrent processing, and consider how you can most effectively utilize the resources available in your specific environment. You should test any change you make in this setting before implementing it in a production environment.

Reasons to use connector agent parallelism

Typically, you should consider using Connector Agent Parallelism in the following circumstances:

- The connector agent was written in C++. Connector agents that use the C++ CDK are inherently single-threaded, but they can be configured to use Connector Agent Parallelism to run processes concurrently.
- The connector agent was written in Java, but needed to hold synchronization locks in calling the application APIs, and so could not make use of the Java multi-threading capabilities. Such a connector can be configured to use Connector Agent Parallelism to run processes concurrently.

Reasons not to use connector agent parallelism

Even if a connector falls into one of the above categories, there may be circumstances where you would probably not want to use Connector Agent Parallelism:

- If the connector is Java-based and designed for multi-threaded operation, you would probably not want to use Connector Agent Parallelism, because running multiple threads concurrently—which the connector can do without using Connector Agent Parallelism—consumes fewer resources.
- An application might have an architecture or requirements that prohibit concurrent processing, regardless of the capabilities of the connector. For example, for TPI connectors, only one instance of the Cyclone server can be instantiated.
- A connector might need to make exclusive use of global resources. In such a case, you should examine the behavior that will result if multiple processes attempt to exclusively access the same global resources concurrently. You might or might not be able to make effective use of Connector Agent Parallelism in such a situation.

Configuring connector agents to process requests concurrently

To change settings for use of connector agent parallelism, start Connector Configurator and choose the “Resources” tab. The “Resources” tab is shown in Figure 109 on page 296:

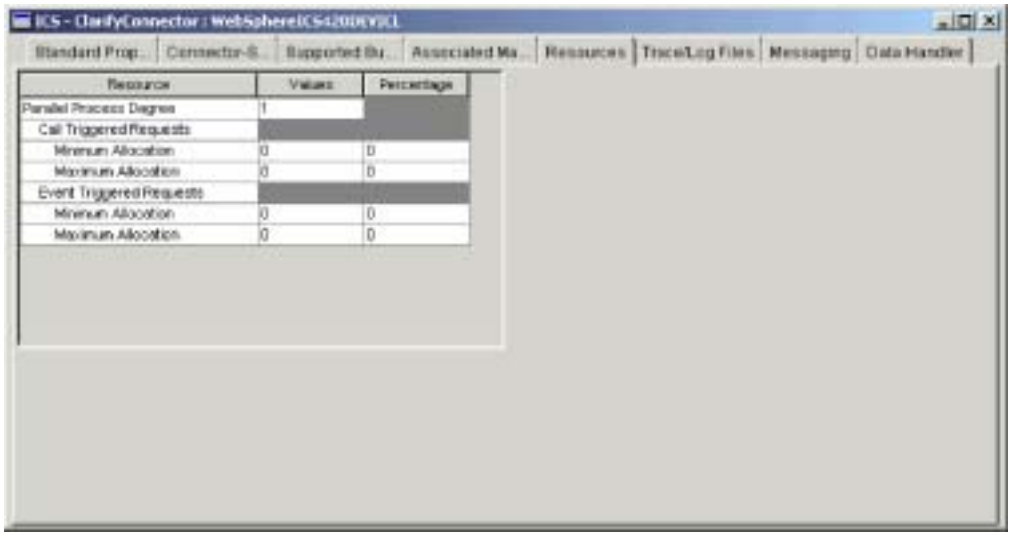

Figure 109. Configuring values on the "Resources" tab

Set the values of the Resources tab for either serial or concurrent processing.

If you change the value from 1 to a higher value, or from a higher value to one, you must reboot the connector agent for the change to take effect. If the value is already higher than one, you can change to another value higher than one dynamically.

For serial processing

By default, connector definitions are configured with the **Parallel Process Degree** property set to the value 1, causing them to use serial processing. When the **Parallel Process Degree** property is set to 1, all other values in the window are set to 0.

For concurrent processing

Do the following to configure a connector to use concurrent processing:

1. Set the **Parallel Process Degree** property to the value 2 or higher. The connector will run the specified number of slave processes concurrently.

   Note: The value represents the number of slave processes. Each connector agent has a polling process, a master process, and a configurable number of slave processes. So for a value of 2, for example, the connector uses a total of four processes—the connector's polling process, the connector's master process, and 2 slave processes.
2. Type the minimum number of slave processes you want spawned for call-triggered flows in the **Values** column for the `Minimum Allocation` property indented under the `Call Triggered Requests` property.
3. Type the maximum number of slave processes you want spawned for call-triggered flows in the **Values** column for the `Maximum Allocation` property indented under the `Call Triggered Requests` property.
4. Type the minimum number of slave processes you want spawned for event-triggered flows in the **Values** column for the `Minimum Allocation` property indented under the `Event Triggered Requests` property.
5. Type the maximum number of slave processes you want spawned for event-triggered flows in the **Values** column for the `Maximum Allocation` property indented under the `Event Triggered Requests` property.

Call-triggered flows are synchronous and are initiated by access clients, which expect a response to be returned for their request. If the connector will be processing call-triggered flows and event-triggered flows, you might want to dedicate more slave processes to the call-triggered flows so that the initiating programs receive as quick a response as possible.

You can either accept the default values, or you can enter the minimum number and maximum number of processes that you want to devote to each type of request. Note that the sum of the maximums for both types of request cannot exceed the value of the Parallel Process Degree property. Also, the minimums cannot exceed the maximums.

To make effective use of Connector Agent Parallelism, coordinate the Parallel Process Degree value of the connector agent with the Maximum Number of Concurrent Events values in the collaboration objects that send requests to the connector agent. If only one collaboration sends requests to this connector agent, set the Maximum Number of Concurrent Events for the collaboration to a value no lower than the Parallel Process Degree value of the connector agent. If multiple collaborations send requests to this connector agent, set the Parallel Process Degree value equal to the sum of the Maximum Number of Concurrent Events values of the collaborations.

If you are using a connector that has Java Virtual Machine (JVM) heap / stack size parameter values that are other than the defaults shown below, you will need to edit the configuration file for the connector. The JVM values that you set in the connector configuration file will be propagated from the connector agent master to the slave processes. To edit these values, use an XML editor to open the `InterchangeSystem.cfg` file, or the local connector agent configuration file if you are using a local file, and edit the values under the JVM <connector_name> section.

The JVM parameters and their corresponding Java VM options are:

```
MIN_HEAP_SIZE    (-Xms)
MAX_HEAP_SIZE (-Xmx)
MAX_NATIVE_STACK_SIZE(-Xss)
```

By default, the following values are used:

```
MIN_HEAP_SIZE=1m
MAX_HEAP_SIZE=128m
MAX_NATIVE_STACK_SIZE=128k
```

You may need to set different values for your connector. For example, you might set the following values in the JVM section of the configuration file

```
[JVM SAPConnector]
MIN_HEAP_SIZE=256m
MAX_HEAP_SIZE=512m
MAX_NATIVE_STACK_SIZE=1m
```

Distributing connector agents

You can improve the performance of the business integration system by distributing the connector agents. The reasons connector agent distribution improves performance include the following:

- Each InterChange Server and connector agent process runs in its own Java Virtual Machine (JVM). A computer can only have so many JVM instances running on it before performance begins to degrade. The threshold is dependent

upon the computer specifications, but distributing a connector agent decreases the load on the machine hosting InterChange Server by one JVM.

- The performance of an individual connector generally improves the closer it is to the application with which it communicates, with respect to the network topology. Ideally the connector agent should be installed on the same computer that hosts the application server itself, though this might not be possible depending on the environment. If the application host computer and the InterChange Server host computer are on different subnets, however, the connector agent will typically still perform better if installed on another computer in the same subnet as the one in which the application host computer exists.

For information on how to distribute connector agents, see the *Installation Guide for WebSphere Business Integration Adapters*.

Caching static relationships

You can cache static relationships so that the participant data they contain is loaded into memory and any queries for the data are issued against the in-memory data, rather than in the data stored in the database. This can make the relationship lookups much faster than if they had to retrieve the information from the database for each flow.

For more information on configuring static relationships to be cached, see the *Map Development Guide*.

Using database connection pools

It is common to have to retrieve or insert information into a database table during the processing of a flow. Performance is impacted if the interface must establish a connection to the database for each flow, particularly because the log in process must be performed for each connection.

Database connection pools are integration components that establish and maintain a pool of connections to a database resource. The connections are established initially and are available to other components to issue queries against the database resource. When a component is done using a connection, the connection is released and returned to the pool so that another component may use it. This eliminates the need to log in each time a connection is required.

For information on how to create database connection pools, see Chapter 8, "Configuring database connection pools," on page 155. For information on how to use database connection pools in maps, see the *Map Development Guide*. For information on how to use database connection pools in collaboration templates, see the *Collaboration Development Guide*.

Using the memory checker thread

InterChange Server features a memory checker thread that you can use to regulate how events are processed by the system and thereby regulate the consumption of system memory. This can mitigate the risk of system crashes due to lack of memory.

The memory checker thread periodically measures the amount of memory used by InterChange Server and evaluates if it is within an acceptable configured range. If

memory usage is not within an acceptable range then the memory checker thread manages system components to reduce the memory usage.

If the memory usage exceeds a lower threshold specified by the lower threshold then the memory checker thread causes the event listener threads of the connector controllers in the system to sleep for a configurable maximum amount of time. The amount of time varies depending on how much the memory usage exceeds the lower threshold, but does not exceed the amount of time specified by the **Connector pause time at threshold** property. By causing the event listener threads to sleep in this way, the memory checker thread slows down the rate at which events are delivered to InterChange Server and reduces the risk of the upper memory threshold being exceeded.

If the memory usage exceeds the upper threshold specified by the **Memory lower threshold percent** property then the memory checker thread pauses the connectors in the system. When connectors are paused they continue to process business objects that are currently queued, but they do not poll new events. The connectors therefore reduce the number of queued business objects and thereby reduce the amount of memory in use, without using more memory to process new business objects. The connectors remain paused for a minimum amount of time specified by **Connector pause time at threshold** property. When the interval specified by the Connector pause time at threshold property passes, the memory checker thread examines memory usage again. If the memory usage does not exceed the upper threshold anymore then the memory checker thread starts the connectors again. If the memory usage still exceeds the upper threshold, however, the memory checker thread allows the connectors to remain paused.

The memory checker thread sleeps in between its examination and state management operations for the amount of time specified by the Memory checker sleep time property.

To use the memory checker thread, you must follow the instructions in the following sections:

1. "Setting the CW_MEMORY_MAX parameter"
2. "Configuring the memory checker thread properties" on page 300
3. "Further considerations" on page 301

Setting the CW_MEMORY_MAX parameter

You must configure the Java parameter `CW_MEMORY_MAX` in the InterChange Server startup script to set the maximum heap memory that InterChange Server can use. The memory checker thread takes action based on the values of configurable properties that specify percentages of this maximum heap memory amount.

You should set the `CW_MEMORY_MAX` parameter to the same value specified for the `-mx` parameter, which specifies the maximum heap size for InterChange Server. By default the `CW_MEMORY_MAX` parameter is set to same value as the `-mx` parameter, but if you modify the `-mx` parameter then you must modify the `CW_MEMORY_MAX` parameter accordingly.

The following example is from an unmodified `start_server.bat` startup script for InterChange Server on a Windows computer:

```
REM This is the  -mx param value for the Interchange Server's memory heap
set CW_MEM_HEAP=512

REM Start the InterChange Server
```

```
%CWJAVA% -Djava.ext.dirs=%JRE_EXT_DIRS%;"%MQ_LIB%";"%DB2_LIB%"
-Duser.home="%CROSSWORLDS%" -mx%CW_MEM_HEAP%m -DTEAgent=1200
-DCW_MEMORY_MAX=%CW_MEM_HEAP% %ORB_PROPERTY% -classpath %JCLASSES%
ServerWrapper -s%SERVERNAME% %2 %3
```

The CW_MEM_HEAP variable is set to the value 512, and is used to set the -mx parameter, so 512 megabytes of memory are reserved for the Java heap. The CW_MEMORY_MAX parameter is set to the same value by also using the CW_MEM_HEAP variable. Note that the CW_MEMORY_MAX parameter must be proceeded by -D.

Configuring the memory checker thread properties

You must edit the InterChange Server configuration file to set the properties for the memory checker thread. These properties are exposed on the "Misc" tab of the InterChange Server configuration interface. For more information on configuring InterChange Server, see "Configuring miscellaneous properties using System Manager" on page 109. Do the following in the "Server Memory" pane of the "Misc" tab to configure the memory checker thread:

1. Set the **Memory checker sleep time** field to the interval of time for which the memory checker thread should sleep in between each operation. Set the **Memory checker sleep time** field to the value 0 (the default value) to disable the memory checker thread.
2. Set the **Memory upper threshold percent** field to the percentage of the total memory available to InterChange Server (specified by the CW_MEMORY_MAX parameter) which, when exceeded, should cause the memory checker thread to pause the connectors so that the connectors do not poll any new events until the memory usage drops beneath the upper threshold again. The default value is 90, meaning that the upper threshold is 90 percent of the total memory specified by the CW_MEMORY_MAX parameter. For example, if the CW_MEMORY_MAX parameter is set to 512 megabytes, then the upper threshold would be approximately 460 megabytes.
3. Set the **Memory lower threshold percent** field to the percentage of the total memory available to InterChange Server (specified by the CW_MEMORY_MAX parameter) which, when exceeded, should cause the memory checker thread to reduce the speed at which connectors deliver events to InterChange Server. The default value is 80, meaning that the lower threshold is 80 percent of the total memory specified by the CW_MEMORY_MAX parameter. For example, if the CW_MEMORY_MAX parameter is set to 512 megabytes, then the lower threshold would be approximately 410 megabytes.
4. Set the **Connector pause time at threshold** field to an amount of time (in minutes) that both specifies the maximum time for which the rate at which connectors deliver events to InterChange Server is reduced when the lower memory threshold is exceeded, and the minimum of time for which the connectors will remain paused when the upper memory threshold is exceeded. The default value is 5.

Figure 110 on page 301 shows the "Misc" tab and the "Server Memory" properties used to configure the memory checker thread.

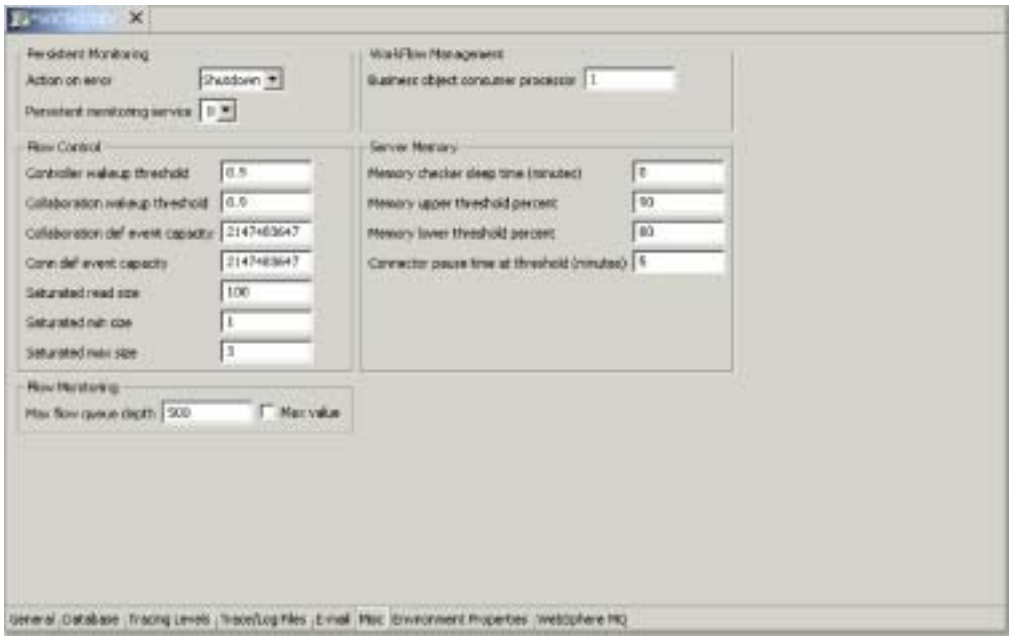

Figure 110. Configuring memory checker thread properties

Further considerations

Consider the following information when using the memory checker thread:

- You should not use the memory checker thread if you are running an InterChange Server instance that processes few transactions. For instance, if you have the `-mx` parameter set to less than 512 megabytes, the Java Virtual Machine garbage collection thread will not be able to free memory effectively.
- Use the following techniques to determine the best configuration for the memory checker thread properties:
  - Initially, set the **Memory lower threshold percent** and **Memory upper threshold percent** properties close to one another, test the behavior of the system, then increase the difference between the values and test the system again. Repeat this process to determine the optimal difference.
  - Set the **Connector pause time at threshold** property between one and five minutes.
  - If InterChange Server must handle a heavy workload, set the Memory checker sleep time property to a small number so that it takes more frequent measurements and regulatory actions.

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800

Burlingame, CA 94010
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not necessarily tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

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

IBM
the IBM logo
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

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

Other company, product or service names may be trademarks or service marks of others. System Manager and other perspectives include software developed by the Eclipse Project (http://www.eclipse.org/)

IBM WebSphere InterChange Server V4.2.2, IBM WebSphere Business Integration Toolset V4.2.2

Printed in USA