

Lesson 3: Top Down Development

Lesson 3 expands on the project begun in Lesson 2, introducing a top down development approach using the Component X Platform. In this lesson you'll create a "market" of buyers and sellers and learn how to use more of the library components that ship with Component X Studio. The completed project stages for this lesson are stored as projects market1 – market4 in the cxTutorials archive, Lesson 3 folder. Lesson sections include:

- 3.1 The Problem Scenario
- 3.2 Defining the Market Roles
- 3.3 Creating and Using an Interface
- 3.4 Adding Implementation to the Seller Role
- 3.5 Adding Pin, Constant and Dialog Components to the Buyer
- 3.6 Testing the Application
- 3.7 Review
- 3.8 Challenge Yourself

One of the problems that has plagued developers is the disconnect between the specification process and the development process. A typical development process requires top down specification of a project down to the lowest levels, and then bottom up development using the specification road map. The Component X Platform facilitates a top down development process that allows development to proceed in conjunction with specification.

3.1 The Problem Scenario

Consider a simple market, which consists of a buyer and a seller who interact with each other. In this market, you assume role of seller, while some external entity is the buyer. You want to create a distributed system that allows you, the seller, to interact with the buyer. You don't know the details of how the buyer will implement business processes, or even how many buyers there might be, each possibly implementing a different business process. At a high level, you'll interface your business logic with the buyer's business logic using some sort of buy/sell protocol. However, at this point you don't even know the exact details of how your own business logic will be implemented. You'll be adding to the project as you learn more about the problem.

Figure 3-1: In the market, the buyer and seller interact with each other.



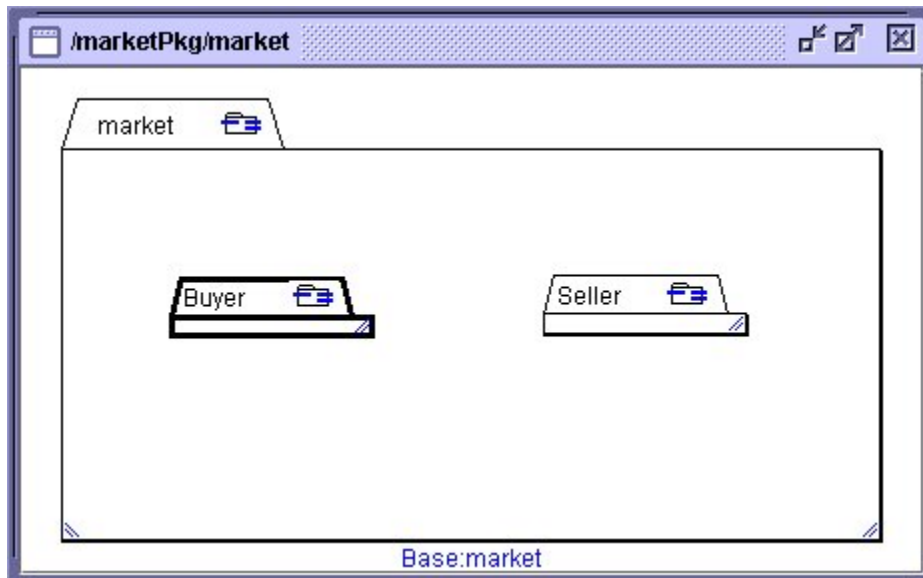
3.2 Defining the Market Roles

The first step is to establish the roles in the market -- the seller (you) and the buyer (the external party) . (The completed project for this section is found in project tutorials:lesson3.market1.)



1. Open a new project and save it as market project in the projects component archive.
 2. Create a new package called marketPkg.
 3. Create a new component called market in the marketPkg package.
 4. Rather than use a pre-built composite component, we'll implement the buyer and seller roles using an empty component and add to it as we discover the required processing.
- From the utility palette, drag an empty component into the market component. Rename it to seller. This is your role.
5. Add a second empty component into the market component, and name it buyer. This is the external entity. The market component should now appear as in Figure 3-2.

Figure 3-2: Create three new components: market and its children, buyer and seller.



3.3 Creating and Using an Interface

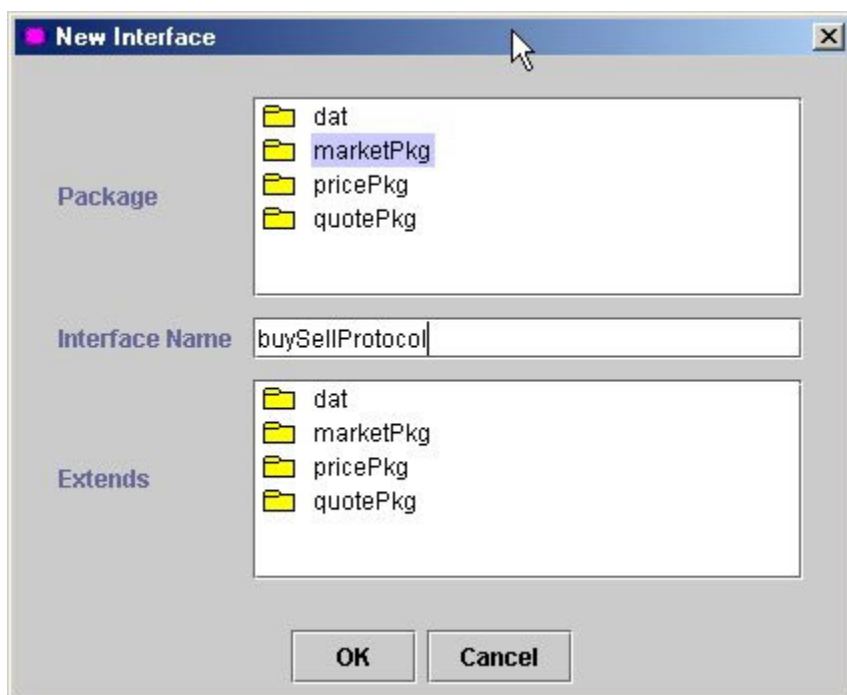
To this point we have created a market component complete with two roles, a buyer and a seller. Now we want to make the buyer and seller interact. In previous lessons input and output ports were used for communication between components. One of the problems with using input and output ports is that the communication is one way. We can create two-way communication using an *interface*, which is a protocol that transmits traffic both into and out of a component.

In researching the next layer of this scenario we discover that buyers request quotes, and sellers supply quotes. But we have already created a set of components and document types to handle quotes in lesson 2. What we would like to do is reuse these components in our market application. (*The completed project for this section is found in project tutorials:lesson3.market2.*)



1. In order to reuse the components that were created in Lesson 2, you could include the project you created in lesson 2 but, instead, include the completed project we have supplied as quotePkg. Use the **Project | Include** menu item, or the include button (🔗), to open the Include dialog, and include project quotePkg in the lesson3 folder of the cxTutorial archive.
2. Now we are ready to create an interface that the buyer and seller will use to communicate. To create a new interface, use the **New | New Interface** menu option, or click the (📄) button.
3. When the New Interface dialog appears, complete it as shown in Figure 3-3 and click OK.

Figure 3-3: Specifying the buySellProtocol interface

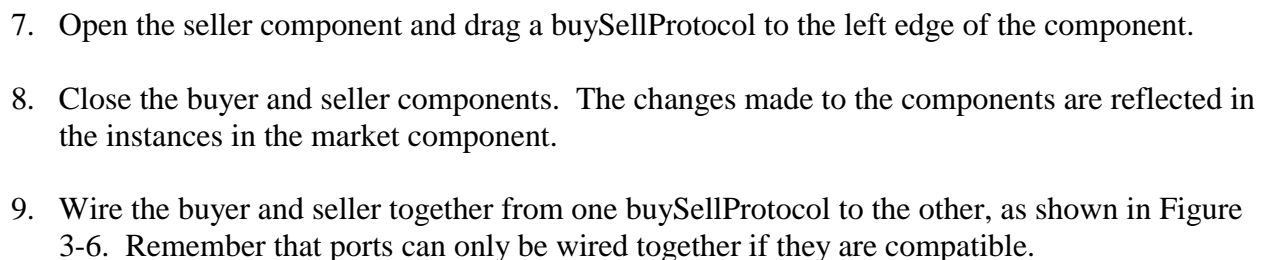


4. In the window for buySellProtocol interface that appears, drop the quoteRequest and quote types into it, as shown in Figure 3-4 (You'll find these types in the quotePkg\quoteTypes nodes of the Component Tree). Drop the quote request on the left margin of the interface and quote on the right margin.

Figure 3-4: Drop quoteRequest and quote document types into the buySellProtocol interface.

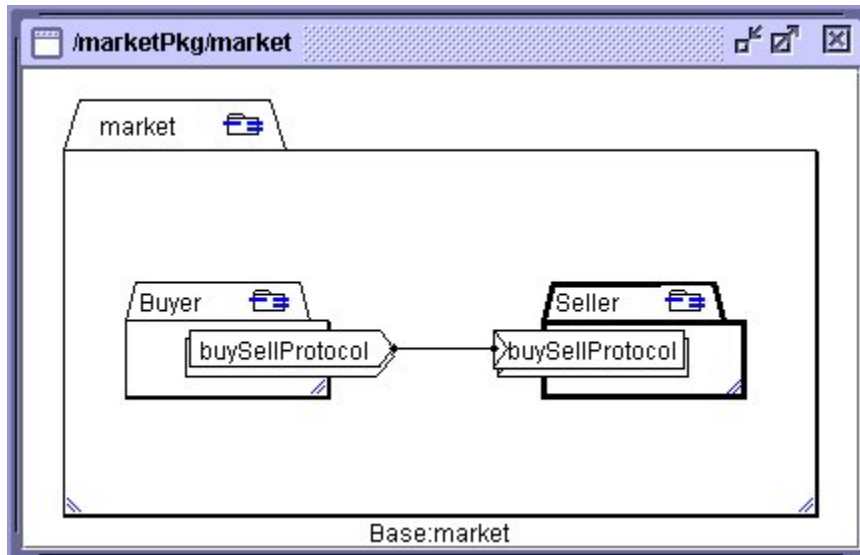


- Figure 3-5: Add the buySellProtocol to the seller component.**



Component X Studio Tutorial Lesson 3 Page 4 © 2001 Data Access Technologies

Figure 3-6: Add the buySellProtocol to the buyer and seller components, and wire them together for two-way communication.



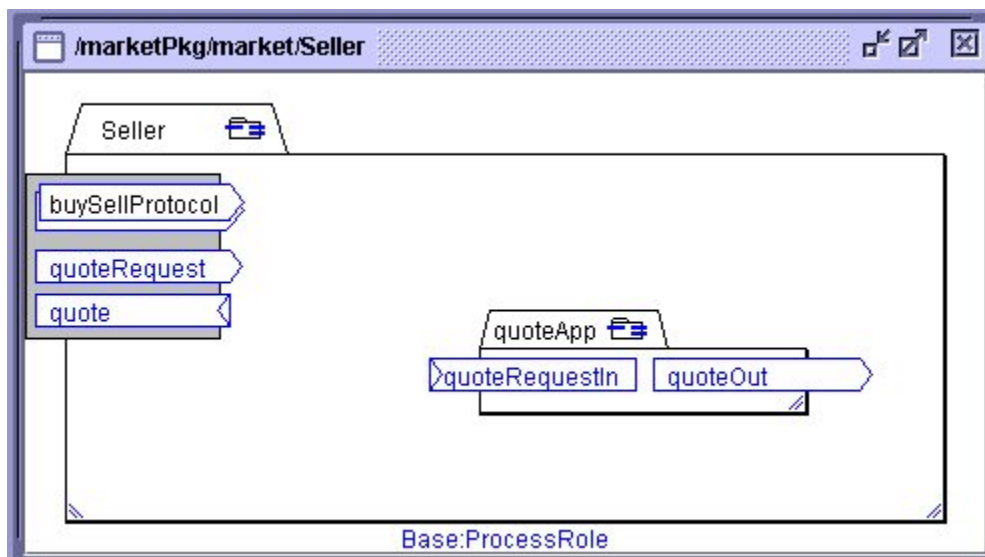
3.4 Adding Implementation to the Seller Role

In the previous sections you created the top-tier components for this project, and created an interface that the components can use for two-way communication.

Now we are ready to add implementation to the seller role. To do this we will make use of the quoteApp component that we created in the previous lesson and is already included in our project in quotePkg. *(The completed project for this section is found in project tutorials:lesson3.market3.)*

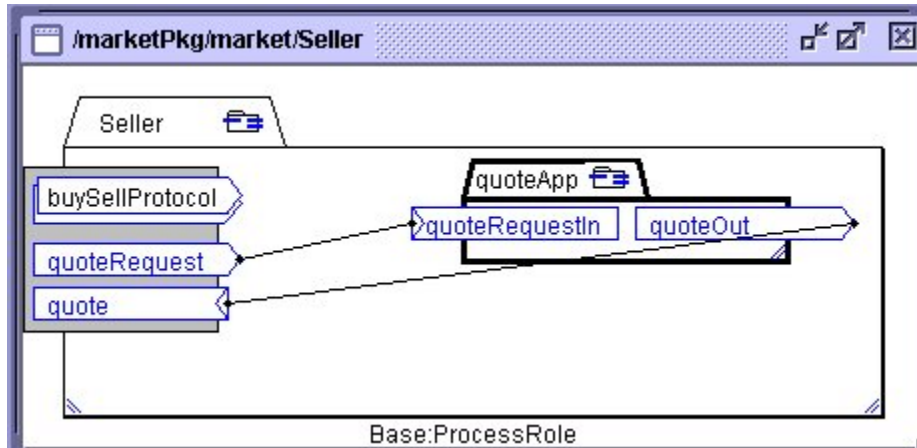
1. Open the seller components (in the market component) and drop into it a quoteApp component, as shown in Figure 3-7.

Figure 3-7: The seller role contains a quoteApp component.



2. Wire the quoteRequest port in the buySellProtocol to the quoteRequestIn port in the quoteApp component, and quoteOut port in the quoteApp component to the quote port in the buySellProtocol, as shown in Figure 3-8.

Figure 3-8: Wire the quoteApp component to the ports of the protocol.



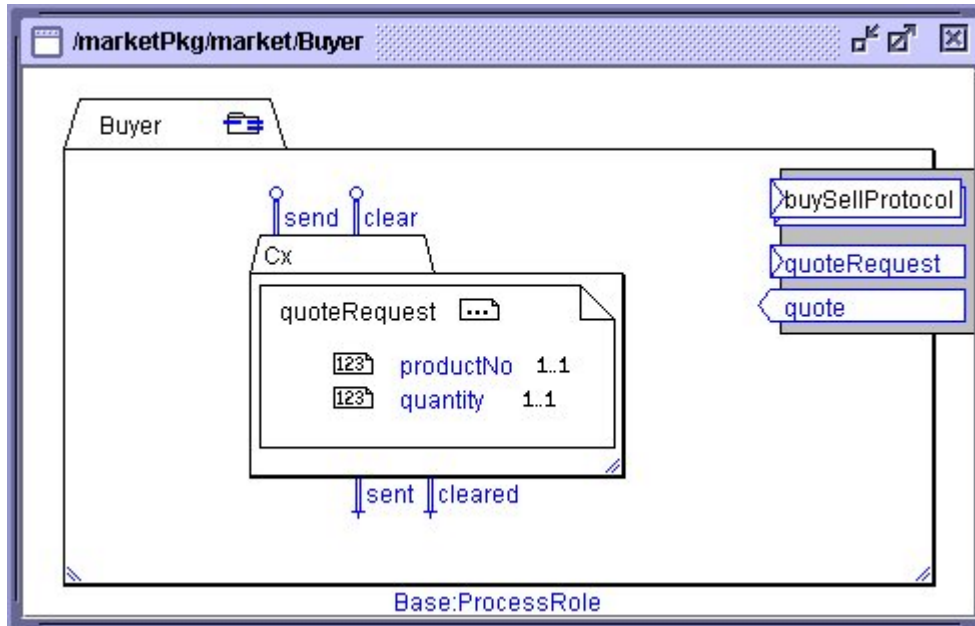
3.5 Adding Pin, Constant and Dialog Components to the Buyer

Before testing the Market application to see if the Buyer and Seller components can exchange the quoteRequest and quote documents, make a few changes to the Buyer component to simplify the testing. In the final implementation, the buyer role might be interfaced to the external entity's business process using adapter for a B2B protocol such as Soap. For now, all we need is to implement some testing logic. *(The completed project for this section is found in project tutorials:lesson3.market4.)*



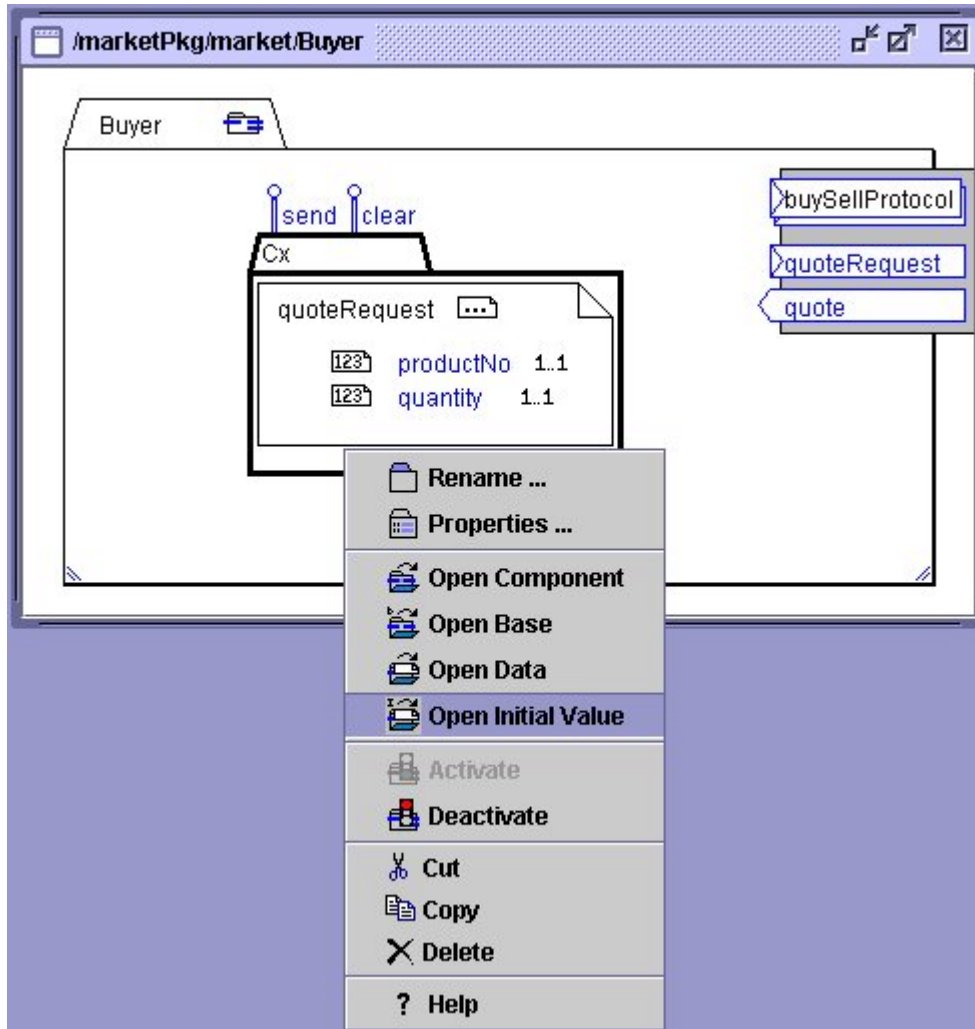
1. Open the buyer component.
2. Drag a quoteRequest component type from the component tree into the buyer component. This creates a variable that can hold a quoteRequest document, as shown in Figure 3-9. A variable can be configured with an initial value using the XML editor.

Figure 3-9: Add a quoteRequest to the buyer component.



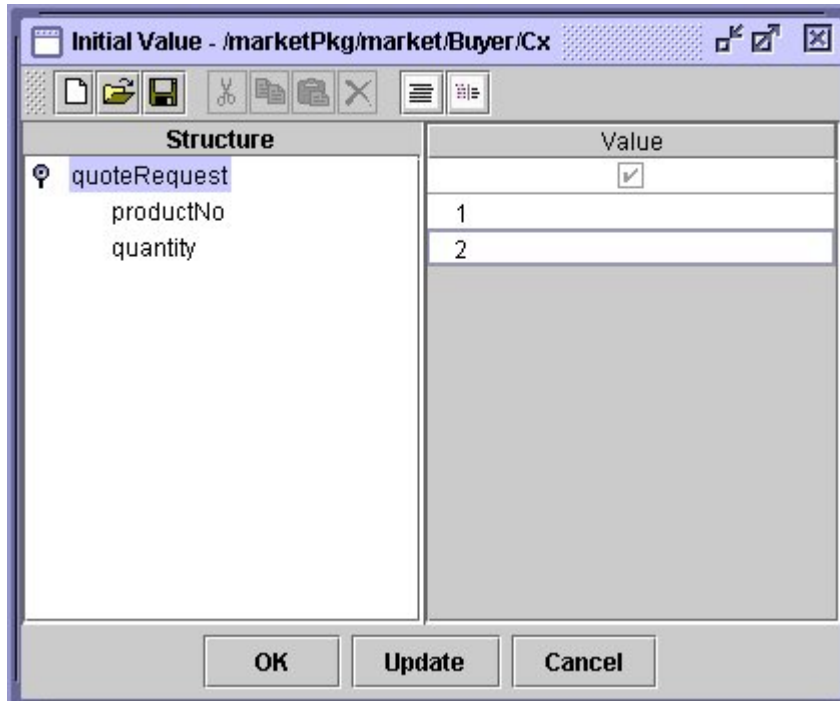
3. Open the XML editor for the initial value of the quoteRequest variable, by selecting **Open Initial Value** menu item from its pop-up menu (A pop-up menu for a component is displayed by clicking the right mouse button over a component), as shown in Figure 3-10.

Figure 3-10: Using a component's pop-up menu to open the XML editor to set the variables initial value.



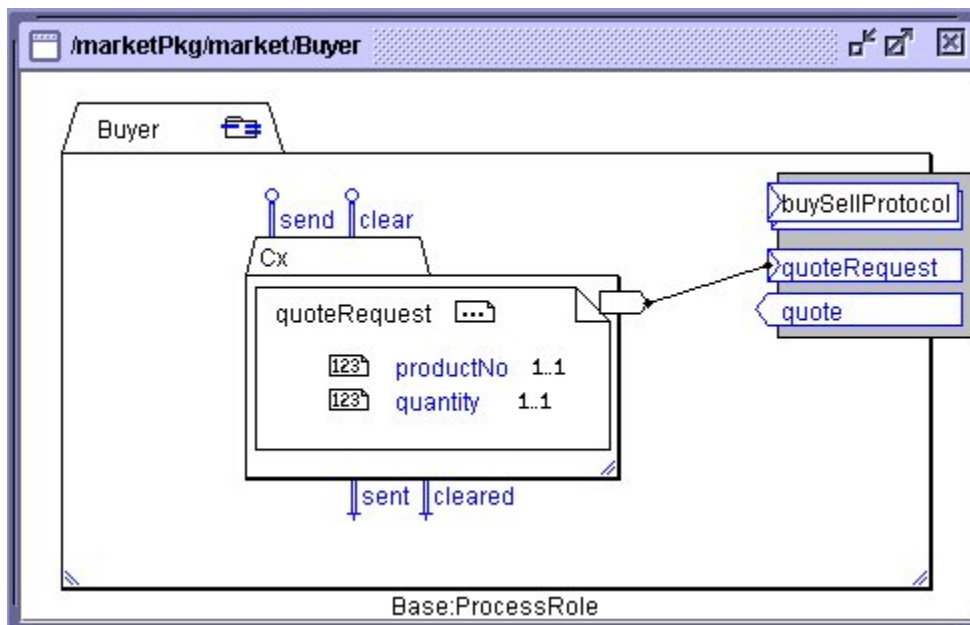
4. Fill in the XML editor shown in Figure 3-11, and click OK.

Figure 3-11: Set the initial value for the quoteRequest variable.



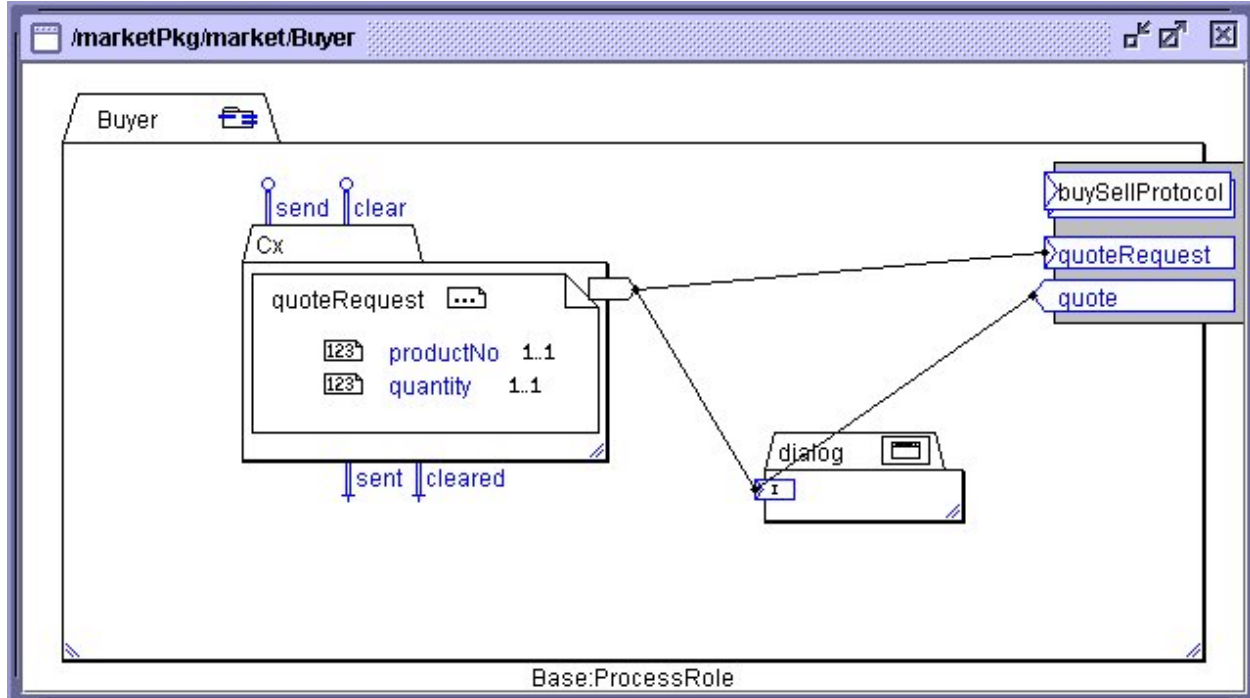
5. Wire an output port of the quote request variable to the quoteRequest port in the buySellProtocol, as shown in Figure 3-12.

Figure 3-12: Wire the constant component to the quoteRequest document in the buySellProtocol interface.



6. Next add a *dialog* component to the buyer component, dragging it from the utility palette. A dialog component is displays a message box dialog when it receives input.
7. Wire the dialog component to the constant component output port, as well as to the quote document in the buySellProtocol.

Figure 3-13: Add a dialog component and wire it to the quoteRequest variable and the quote port in the buySellProtocol.




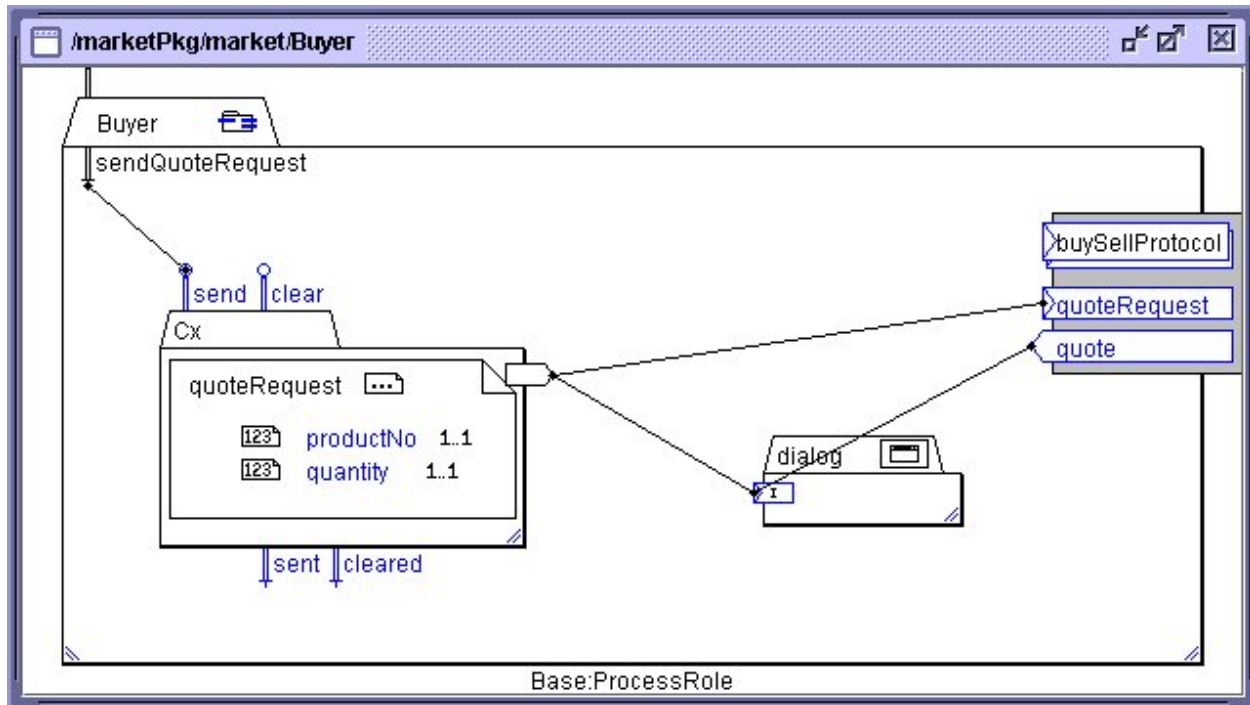
8. One more enhancement will make it easier to test this project. We could signal a send from the constant component by clicking on its send pin. However, to do this requires that we open the buyer component. It would be better if the quoteRequest could be sent directly from the market component. We can do this if we add a send pin to the buyer component, and wire it to the constant component. Click on the Add Control Pin () button, and complete the Add Control Pin dialog as shown. Make sure the Provides choice is selected, so that the pin will be created as an "input" pin. Click OK.

Figure 3-14: On the buyer component, create an "input" control pin called sendQuoteRequest.



9. Now wire the sendQuoteRequest pin to the Send pin on the quoteRequest variable.

Figure 3-15: Wire the sendQuoteRequest pin to the quoteRequest variable, so that the quoteRequest can be sent without opening the buyer component.



10. Now close the buyer component. Open the market component, if it is not already open. Notice the sendQuoteRequest pin on the buyer component.

3.6 Testing the Application

In the previous sections, you created a component, and added two subcomponents to it, and wired them together so that they can “talk” to each other. You also learned that the Component X Studio is an “immediate” development environment, in which your open components are always running. In this section we’ll test the application by sending some documents between them.



1. Use the **Window | Close All** menu item to close all open components.
2. Open the market component.
3. Notice the sendQuoteRequest pin on the buyer component. Open this pin (by double-clicking it) and press the Send button.
4. Dialogs containing the quote and quoteRequest will appear. Examine them, and then click OK to dismiss them. You'll see that the quote is complete with unit price and total price.

Figure 3-16: Testing the market project using the sendQuoteRequest pin.

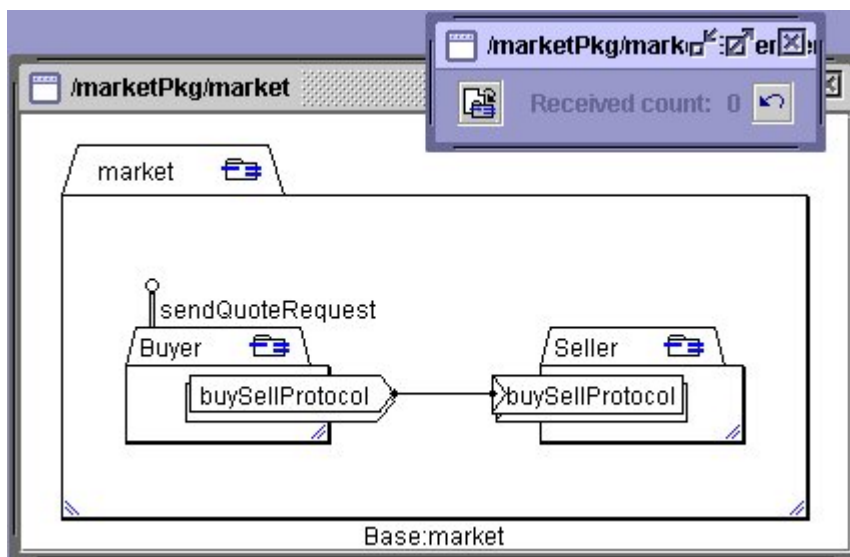
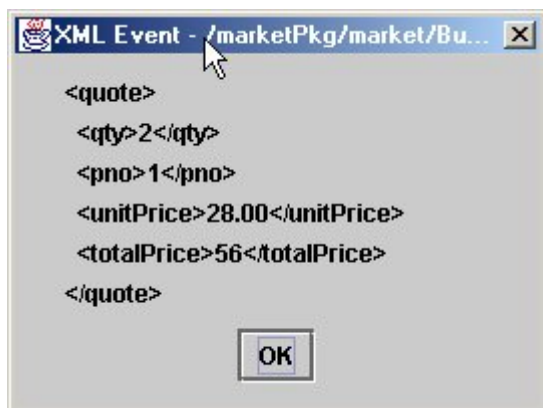


Figure 3-17: The quote result is output to the dialog.



3.7 Review



This lesson demonstrated how Component X Studio can be used as a tool for top down development. It showed how layers of complexity can be added to a project as they are discovered. It also demonstrated how components in an existing Component X project can be reused in a new project.

Templates are a convenient way to create composite components. You used a processRole template to create the buyer and seller roles. Typically you would only implement one side of the business logic, and the other would be implemented by an outside entity.

Interfaces are used to establish two-way communication between components. One or more document types can be added to an interface, so that the components can exchange the documents across a single wire.

Variables hold documents whose initial values can be set using the XML editor. The initial values can be fed to components, and are particularly useful in testing.

Dialog components display the input they receive in a message box, and are also useful for testing.

3.8 Challenge Yourself



Test what you learned in this lesson by adding the following functionality to your project:

1. Make the market component easier still to use by adding a `sendQuoteRequest` control pin to it, and wiring it to the buyer component `sendQuoteRequest` pin. (This enhancement can be found in `market5` of the tutorial component archive.)