



IBM Connect for iSeries

Remote Catalog Implementation Guide

IBM PartnerWorld for Developers
Sept. 20, 2001

License and Disclaimer

This material contains IBM copyrighted sample programming source code ("Sample Code"). IBM grants you a nonexclusive license to compile, link, execute, display, reproduce, distribute and prepare derivative works of this Sample Code. The Sample Code has not been thoroughly tested under all conditions. IBM, therefore, does not guarantee or imply its reliability, serviceability, or function. IBM provides no program services for the Sample Code. All Sample Code contained herein is provided to you "AS IS" without any warranties of any kind. **THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY DISCLAIMED. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSIONS MAY NOT APPLY TO YOU. IN NO EVENT WILL IBM BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES FOR ANY USE OF THE SAMPLE CODE INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOSS OF PROGRAMS OR OTHER DATA ON YOUR INFORMATION HANDLING SYSTEM OR OTHERWISE, EVEN IF WE ARE EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**

COPYRIGHT

(C) Copyright IBM CORP. 2001

All rights reserved.

US Government Users Restricted Rights -Use,

duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Overview

The cXML specification provides for a function known as “Remote Catalog”. Remote catalog is a redirect from the client eProcurement software to a supplier shopping site. The buyer shops for items much like a B2C environment but upon checking out, special messages are sent back to the eProcurement software to allow the buy side to update internal tables.

ISV’s who have catalog management software similar to WebSphere Commerce Suite, may need to integrate their catalog implementation with Connect for iSeries to provide what is known as “RemotePunchOut”. Connect for iSeries release 1.1 provides special hooks to make it easy for 3rd party catalog tool vendors to perform this integration.

The following steps describe an example “Remote Catalog or Remote Punchout” as it could be implemented using Connect for iSeries Version 1.1:

Step 1: The buyer is running eProcurement software and selects a pre approved selling organization for goods and services needed by the buyer at this time.

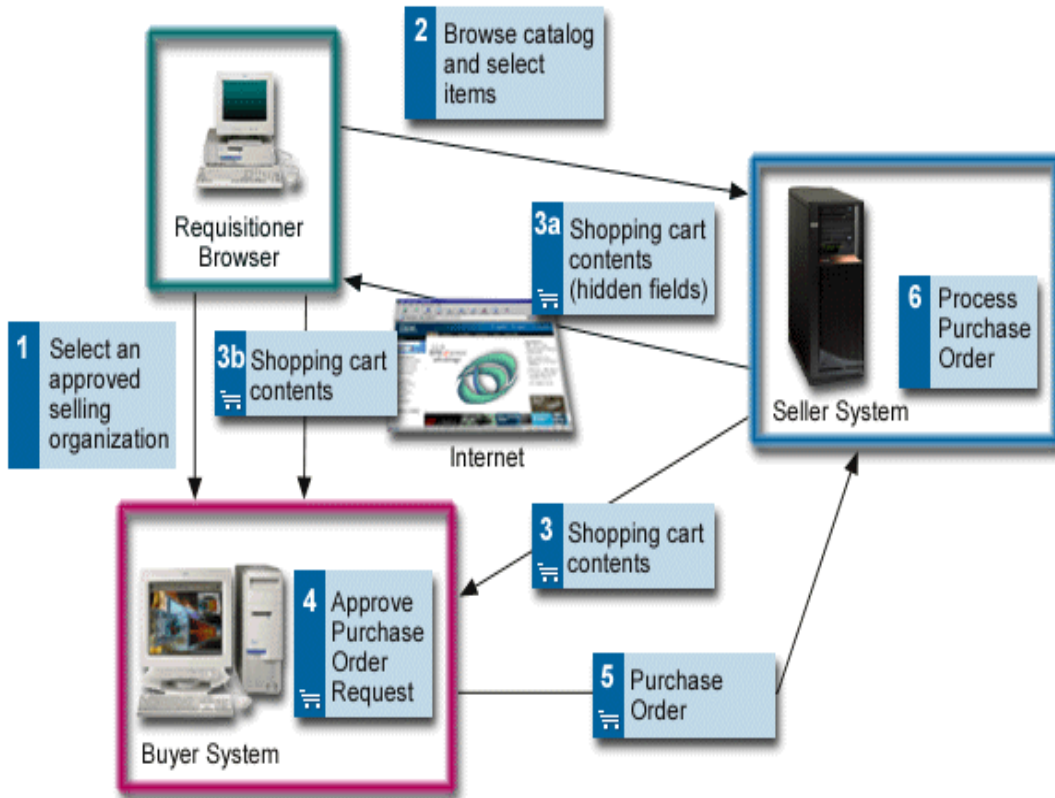
Step 2: A “PunchoutSetup” message is sent to Connect for iSeries. Connect for iSeries authenticates the buyer and returns a URL and other special fields to the buyer.

Step 3: The eProcurement software opens a browser window to the URL provided in the Punchoutsetup response and the buyer is redirected to that URL. The buyer may be automatically logged in with the special fields provided in the PunchoutSetup response message.

Step 3a: The Buyer shops for products and services much like a B2C shopping experience. The last step is buyer approval of the selected items. This is known as generating a “NewQuote” and the shopping experience ends. NewQuote messages are sent to Connect for iSeries so Connect can determine what products were selected by the buyer. These products are sent back to the buyer eProcurement software in the PunchOutOrder Message.

Step 3b The eProcurement updates its internal database of products requested and places them in a submitted state.

Step 4. The submitted items require approval at the buyer site. Upon approval, an OrderRequest message is sent to Connect for iSeries containing the products and services selected during the remote catalog shopping experience. The OrderRequest message is the same type of message sent when the catalog is local to the buyer.



This paper is written only as an example on how one might implement a remote catalog solution. There is really no right or wrong way of implementing this functionality.

The steps involved in the setup and configuration of Connect for iSeries are for RemotePunchOut are:

- Know the database schema for the catalog management software and how it relates to database information native to Connect for iSeries. As an example, assume the catalog management software contains information on shoppers & merchants. Within Connect for iSeries, shoppers are known as buyers and merchants are known as suppliers. It may be necessary to make some association between the catalog tables and Connect for iSeries tables. In addition there may be fields required by the catalog management system that are not native to Connect for iSeries.
- Create custom configuration screens within Connect for iSeries to supply the necessary fields required by connectors or Buyer/Supplier exit routines. These fields can be added to Connect through customization of configuration screens. These fields can then be accessed using API's provided by Connect for iSeries
- Create two required connectors
 - ✓ PunchoutSetupRequest
 - ✓ NewQuote

- Making necessary changes to the catalog management software. This will include becoming “RemotePunchout Aware”. The catalog software will need to generate an html page capable of sending a “NewQuote” message. This is usually the final page the user sees when they are finished shopping. This page will send the necessary data to the NewQuote connector within Connect for iSeries for processing. In addition it may be necessary to customize the shopping experience for each customer. For example, it is a nice feature to automatically login and authenticate the buyer when redirected from Connect for iSeries. This may require special modifications to the catalog management software to provide a means of automatic login.
- Create Buyer/Supplier exit routines. These routines allow the developer to gain control of the Connect for iSeries configuration to provide any function necessary when buyers/suppliers are created, modified, or deleted. This step is not provided in the example.

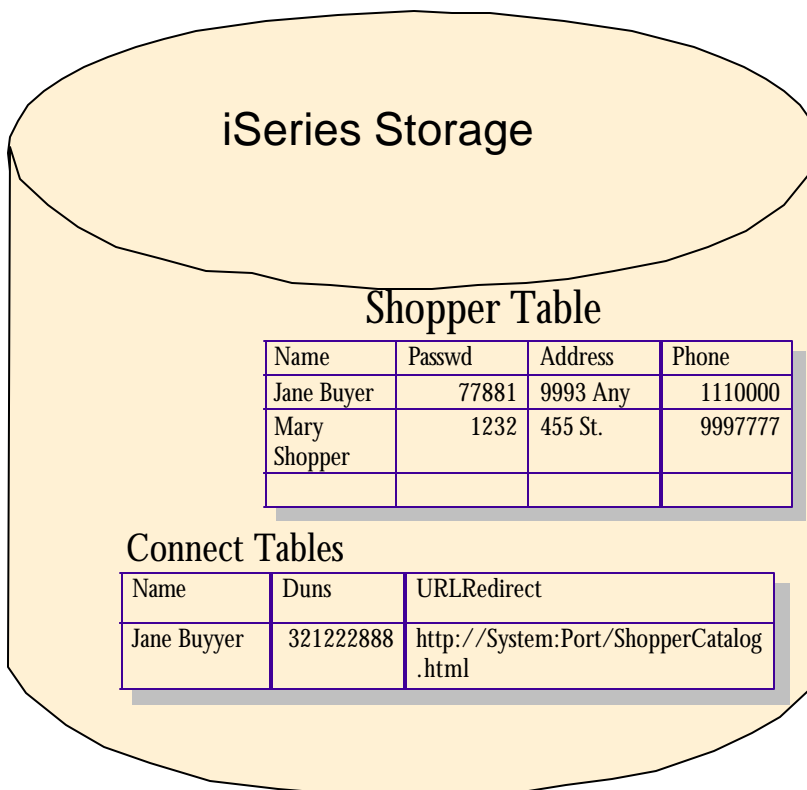
Knowing the Database Schema

Knowledge of the database schema's is critical. Connect for iSeries maintains a set of tables that it uses internally. It is recommended not to access or modify those tables directly. Use the API's provided in `com.ibm.connect.tools.tpa.api` to access table data stored in Connect for iSeries tables. IBM reserves the right to make changes to tables and fields in future releases. Using the API's will guarantee your code will work in future releases of Connect for iSeries

Create Customized Configuration Screens

Connect for iSeries Version 1.1 provides the ability to customize the configuration screens. It is here that one can add extra data needed by the catalog implementation. Connect for iSeries saves this data internally allowing access to this information with a set of API's provided. This customization may be necessary when unique or specific fields are needed to integrate with the catalog system.

As an example Connect for iSeries will need to send back a URL to the PunchOutSetup Request message. A URL field is not a native field provided by Connect for iSeries configuration. The example will create a customized page for the buyer and provide an additional field called URLRedirect. Within the Punchoutsetup.java connector, it will have access to all of the Connect for iSeries buyer configuration information using special API's in com.ibm.connect.tools.tpa.api.



Create two required connectors

Two need to be created. They are the PunchoutSetupRequest and NewQuote connectors. These connectors are Java code that are called when PunchoutSetup (create) and NewQuote messages are received by connect for iSeries.

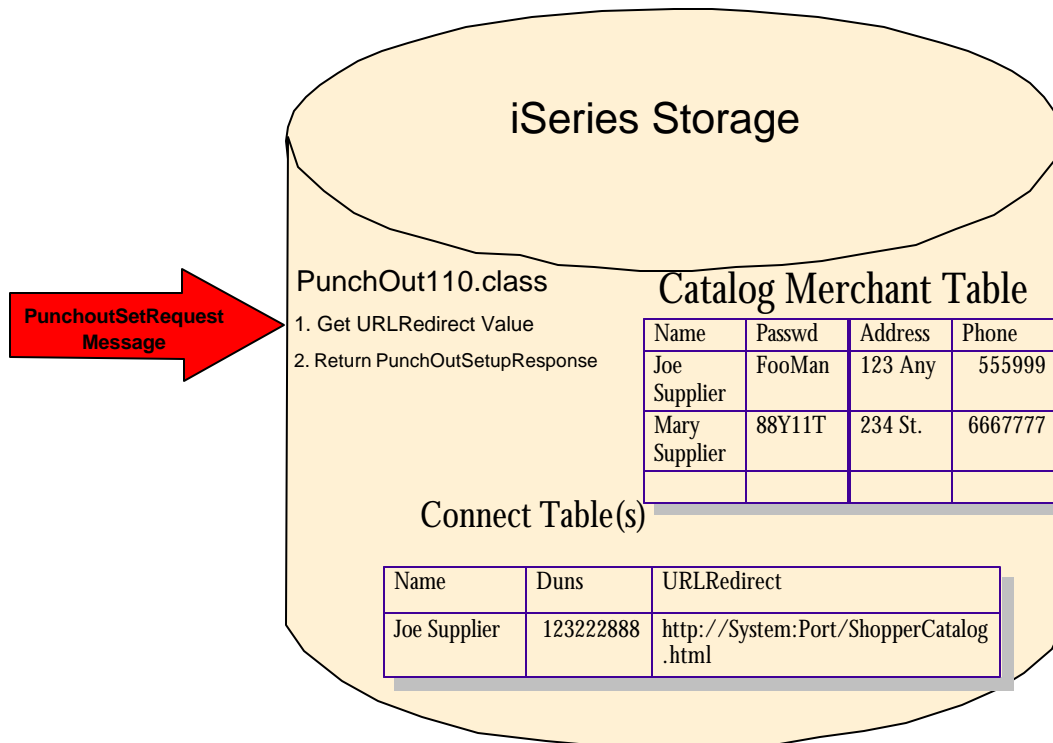
PunchoutSetupRequest (create) responds with URL redirection to the buyer software. This URL will be the main page of the of the catalog management software used for RemotePunchOut.

NewQuote is an IBM defined message that allows the connector to determine what products have been purchased from the catalog software. NewQuote generates a PunchoutOrder message containing products shopped for in the catalog management system.

This paper is not a lesson in developing and writing connectors. Use the examples provided and make modifications to them when implementing custom connectors.

Punchoutsetup Request

For the example, when a PunchoutSetupRequest message is received by Connect for iSeries, the connector “PunchoutConnector.class” will be called. This connector will lookup the URLRedirect field within the Connect for iSeries tables based on the buyer name. It will then format a PunchoutSetupResponse message containing the URL which the procurement software should redirect too.

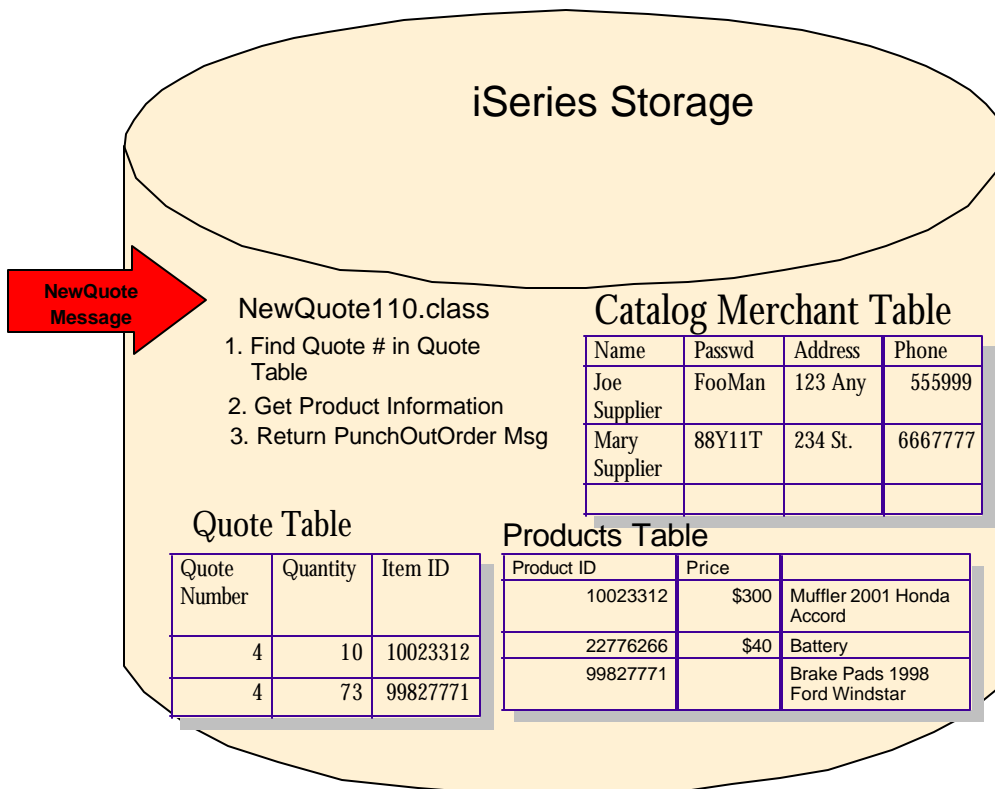


A real world setting might respond back with additional information like a buyer cookie, allowing the buyer to be automatically logged in to the catalog management software. The point to recognize is that from connectors or Buyer/Supplier exit routines, access is provided to the Connect for iSeries tables through a set of API's in com.ibm.connect.tools.tpa.api. Study the API's in this package to learn how to access data stored within Connect for iSeries Tables.

NewQuote

When Connect receives the NewQuote message it passes control to the connector assigned for NewQuote. This connector can then access the quote table filled in by the catalog management software. This allows NewQuote to determine what products were selected from the shopping experience. The NewQuote Connector can then access the associated product information in the products tables, and send back a PunchoutOrder message to the eProcurement software.

As an example, assume the buyer was redirected to a suppliers catalog site. The buyer shopped for items and are ready to accept the order. The last step is the process is a confirmation from the user and the sending of the “NewQuote Message” to Connect for iSeries. In the example provided, the SupplierCatalog.html page provides sending the “NewQuote”. A page similar to SupplierCatalog.html will need to be generated by the catalog management software. In the example, a quote # is sent to Connect for iSeries in the NewQuote message. This allows the NewQuote connector to access a database table containing items in the quote. The NewQuote connector can then create a PunchoutOrder message containing all of the products ordered in the catalog management software.

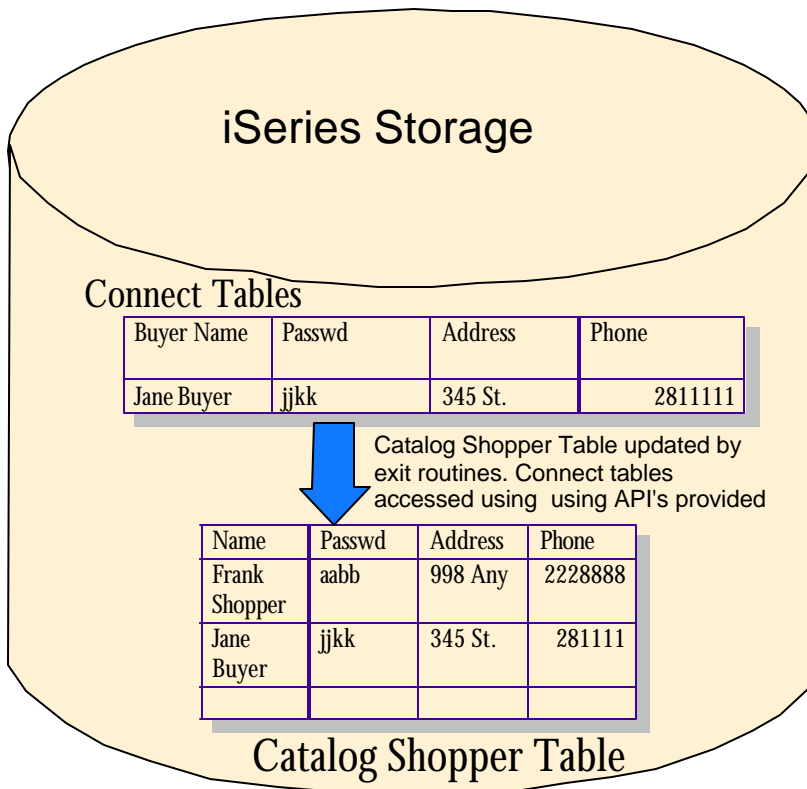


Study the SupplierCatalog.html file. Understand the JavaScript that is called when the “Send Quote” button is pressed.

Buyer/Supplier Exit routines

Connect for iSeries provides the capability for exit routines to be called upon creating, updating, or deleting buyers and suppliers. These routines can be used to “assume control” of Connect for iSeries configuration. This may be necessary so access to tables outside of Connect can be updated.

Assume the catalog management system contains a table containing registered shoppers. By utilizing a buyer exit routine, updates to the catalog management's shopper table can be made when a new buyer is registered within Connect for iSeries. Likewise, if a buyer were deleted from Connect for iSeries, an exit routine could be called to remove the shopper from the catalog management software.



The example provided does not make use of buyer/supplier exit routines. They are noted here so that readers can be aware of this capability when designing a system.

Modifications To Catalog Management Software

The catalog management software will need to know that the buyer is coming in from an eProcurement system and performing a “RemotePunchOut”. Modifications will be required to the catalog system so that when the buyer has finished shopping a special message can be sent back to Connect for iSeries. The message that gets sent is a “NewQuote” message. The NewQuote message is not part of the cXML protocol. NewQuote is a special message developed by the Connect for iSeries team specifically for RemotePunchout. The NewQuote messages calls the associated NewQuote connector. This connector should be written to understand what items were selected by the buyer using the catalog software. The NewQuote connector can then create and send a PunchoutOrder message back to the buyer containing all of the items ordered.

The catalog software may require changes to the DB tables to include DUNS numbers for each item in the catalog. It may be necessary to customize the catalog depending on the supplier currently shopping so that specific items and prices are displayed to the buyer based on purchasing agreements between the 2 companies. Automatic login is another nice functionality for RemotePunchout shoppers. When redirected to the site, providing login information so that the user is authenticated and logged is something that might be added to the system. The changes to the catalog system are implementation dependent. Consider this step carefully during the design phase.

Remote Catalog Example

The example provided is intended to provide the reader with the basic knowledge of how to integrate Connect for iSeries with a catalog management system. This example provides the following:

- A sample eProcurement program to drive the system. It “simulates” what an eProcurement package might do. When it receives the `PunchoutSetupResponse` message it opens a new browser window to the simulated `SupplierCatalog.html`.
- A sample `PunchoutSetupRequest` connector is provided. This connector is called when the `PunchoutSetup` message request is received. This connector responds to the caller with a `PunchoutSetupResponse` containing a `URLRedirect` field. This field is added to the Connect for iSeries table from a custom page in buyer configuration. This URL will be configured so that it can display the `SupplierCatalog.html` page.
- A shopping summary page of a “Simulated” catalog is provided. A page like this would normally be dynamically generated by the catalog management software to send the “NewQuote” message to connect for iSeries. The `SupplierCatalog.html` will send the `NewQuote` message request to Connect for iSeries.
- A sample `NewQuote` connector is provided. The connector is called when the `NewQuote` button is pressed on the `SupplierCatalog.html` page. This `NewQuote` connector will format a `PunchoutOrder` message and send it to a `PostBackURL`.
- A `QuoteTest.class` java servlet is provided that will receive the `PunchoutOrderMessage` and display the order in a browser window. This message is what normally would be received by the buyers eProcurement software so that it may be able to update it’s internal order tables.

When the `PunchoutSetup` response is received by the sample eProcurement program, it will open up a new browser window and redirect to the URL provided to it in the `PunchoutSetupResponse` message. In the example it will point to a `SupplierCatalog.html` URL. `SupplierCatalog.html` is an example of what the catalog implementation would need to generate when the buyer has finished shopping. Study the source to `SupplierCatalog.html`. Notice that the URL that it redirects to is the `storeNewQuote` defined for the `exampleB2B` Instance.

The final step in the process of will not be shown in this example. This step is the `OrderRequest` message received by Connect for iSeries containing the items sent back in the `PunchOutOrder` message. This message will be the same format as if the message originated from a buyer hosting a local catalog implementation.

Within SupplierCatalog.html is JavaScript that is called when the “Send Quote” button is pressed. The script is a redirect to the storeNewQuote URL of the B2B Instance. Notice the information sent to the storeNewQuote URL.

```
<SCRIPT LANGUAGE="JavaScript">
function valid(form){ var input=0; input=document.myform.data.value;
location.href=input+"?SupplierNumber=123222888&SupplierNumberDomain=DUNS&BuyerNumber=321222888&BuyerNumberDomain=DUNS&SupplierCookie=3&QuoteNumber=4&PostBackURL=http://RACKEM/servlet/QuoteTest";
}
</SCRIPT>
```

Required fields:

Supplier/Buyer information is used for gateway authentication and authorization. PostBackURL is used as the location to send back the PunchOutOrder message by Connect for iSeries.

Optional fields:

SupplierCookie is an unused field by the example. It is shown only as an example of how to send additional fields to the NewQuote connector.

Quote Number is used to determine the items purchased in the Catalog software.

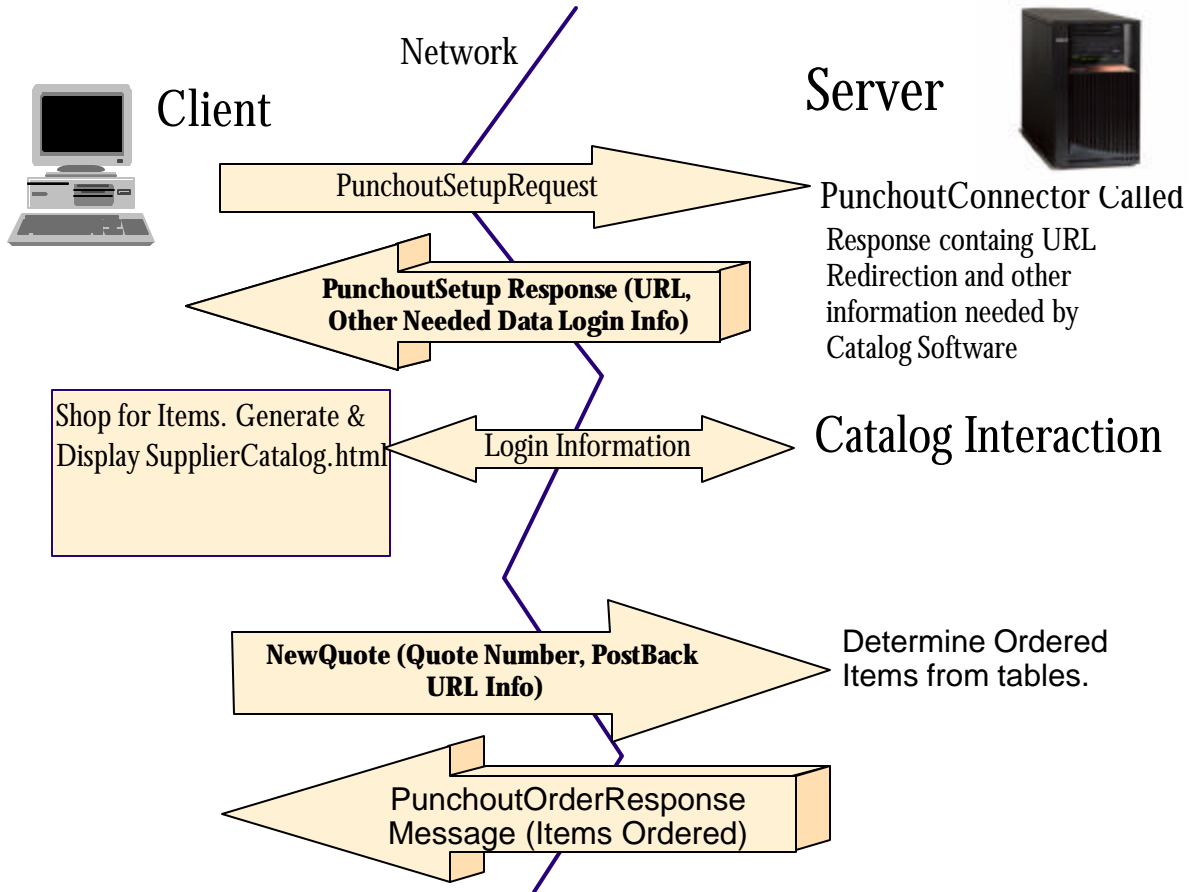
A real implementation may pass the actual shopping data or just a quote that can be looked up in database tables of the catalog to determine what items are ordered. This example passes a Quote Number which is looked up in the Quote Table database by the NewQuoteConnector.

When Connect for iSeries receives the message at the storeNewQuote URL, the NewQuoteConnector class is called. The NewQuoteConnector is responsible for sending back a PunchoutOrder Message containing all of the items that were purchased using the catalog. This example reads the RSAMPQUOTE table looking for quote information. From the quote table, the NewQuoteConnector can determine what products were purchased in the catalog management software. The connector will then send back the PunchoutOrder information to the PostBackUrl.

“Sending Quote...” Will flash in the browser window. What occurs here is that upon load of the page, it redirects the PunchoutOrder message to the PostBackURL specified. In the example it redirects to a Servlet called QuoteTest which displays the PunchoutOrder message in a browser window. The message will be displayed encoded URL base 64. In a real world situation the eProcurement software would be listening on the URL, accept the PunchoutOrder message, and update it’s internal tables as

necessary.

Remote Catalog Sequence of Events



Set Up the Example

The Prerequisites for running an example RemoteCatalog are:

- Connect 1.1 Installed
- Internet Explorer 5.5
- WebSphere Application Server Console
- B2B Instance Created

Internet Explorer 5.5

Make sure Internet Explorer 5.5 or newer is installed on a PC. Verify that the PC and iSeries are configured such that the browser can communicate with the iSeries over TCP/IP.

Connect 1.1

Follow the instructions for installing Connect 1.1 and all of the prerequisite software. It is a good idea to have created an instance and run one of the sample programs prior to working on the Remote Catalog.

WebSphere Application Server Console

Verify that the WebSphere Application Server Console is installed on the PC and can connect to the WAS default admin server running on the iSeries Machine. Start the Default Server instance and run the snoop WAS example to verify that WAS is running and configured correctly.

A B2B Instance Created

The example program will require a connect instance prior to installation. There is no need for Buyer/Supplier registration. Create a B2B Instance using the following fields. It is assumed the reader is familiar with creating B2B Instances.

Field	Value
Run Delivery Gateway on the server	Yes
BtoB instance name:	RemoteTest
MarketPlace Name	RemoteCatalog Example
Supplier URL for marketplace:	.../store
MarketPlace Protocol Supported	CXML 1.1
Supported Requests	ProfileRequest, PunchoutSetupRequest, NewQuote
Create a New HTTP Server	Yes
Port #	3496 (Make sure this is available on the system. Net Option 3, F14 to verify)
Use WebSphere Commerce Suite?	No


IBM Connect For iSeries Administration - PID400A - HENKEL - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print Print with Selection

Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer IBM Standard S

Address http://pid400a:2002/BtoB/Connect



Connect for iSeries

Home **Instances** Suppliers Buyers Catalog Deployment

Marketplace Information

S
B

Enter a name for this new B2B instance and its associated marketplace. Select the marketplace protocol support. To change the list of available marketplace protocols, see the Customize page for instance Protocols tab, you can add your custom protocols to the list of available protocols.

B2B instance name: *

B2B instance description:

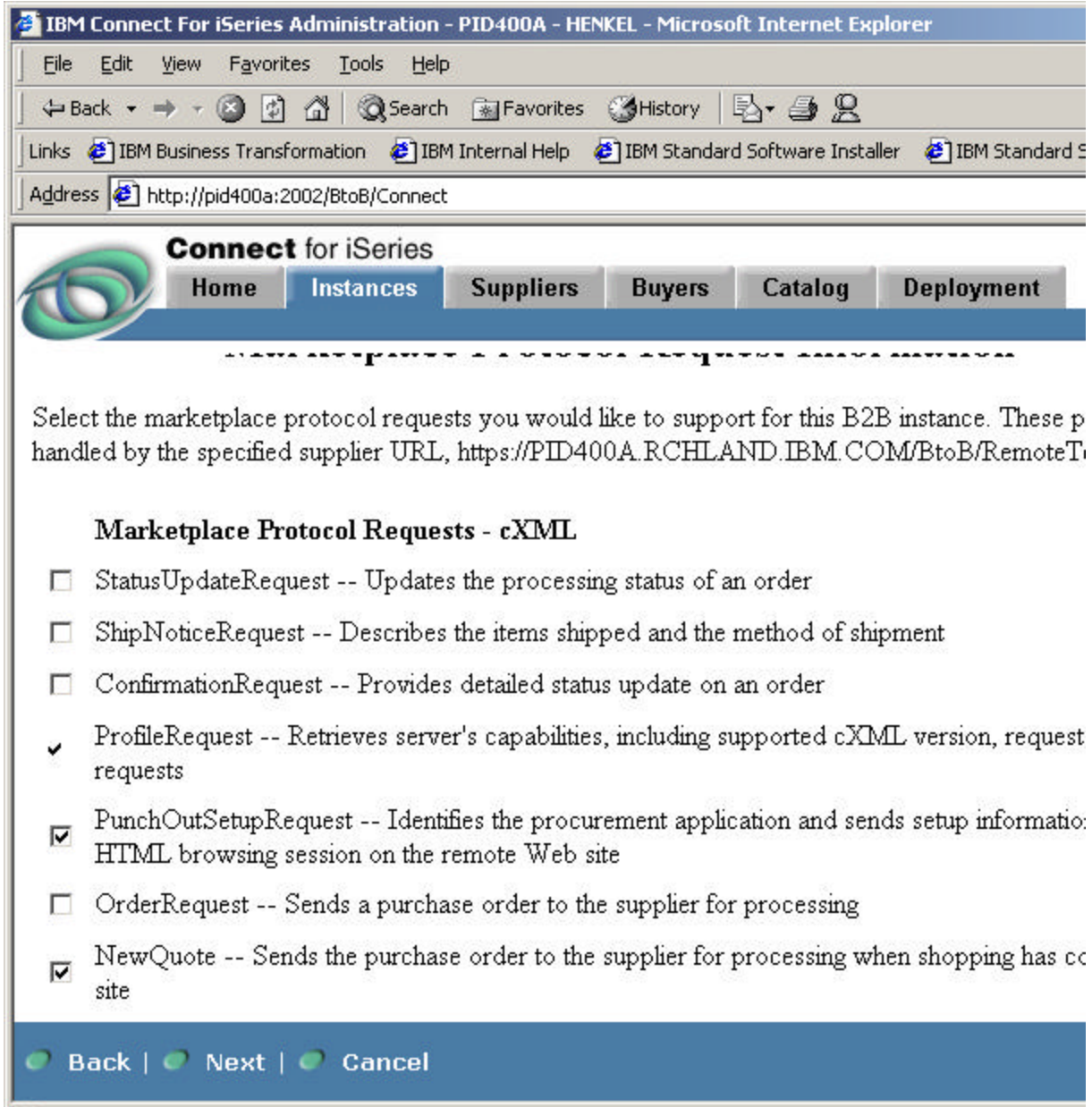
Marketplace name this B2B instance supports: *

Supplier URL for marketplace: *

	Marketplace Protocol	Subtype	Version
Marketplace *	<input checked="" type="checkbox"/> cXML	Ariba	1.1

Back | Next | Cancel

Done



IBM Connect For iSeries Administration - PID400A - HENKEL - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites History

Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer IBM Standard S

Address http://pid400a:2002/BtoB/Connect

Connect for iSeries

Home Instances Suppliers Buyers Catalog Deployment

Select the marketplace protocol requests you would like to support for this B2B instance. These p handled by the specified supplier URL, https://PID400A.RCHLAND.IBM.COM/BtoB/RemoteT

Marketplace Protocol Requests - cXML

- StatusUpdateRequest -- Updates the processing status of an order
- ShipNoticeRequest -- Describes the items shipped and the method of shipment
- ConfirmationRequest -- Provides detailed status update on an order
- ProfileRequest -- Retrieves server's capabilities, including supported cXML version, request requests
- PunchOutSetupRequest -- Identifies the procurement application and sends setup informatio: HTML browsing session on the remote Web site
- OrderRequest -- Sends a purchase order to the supplier for processing
- NewQuote -- Sends the purchase order to the supplier for processing when shopping has cc site

Back | Next | Cancel

Continue through the configuration screens until the instance is created.

Install the Sample

RemoteCatalog Sample is packaged as follows:

- Client.zip Client side Java Program for sending XML based messages (simulates an e-procurement piece of software).
- Connectors.zip Connector classes.
- Deployment.zip PCML, ADC, PFM files for the Connectors.
- Catalog.zip
- Remoteex.savf

Client.zip

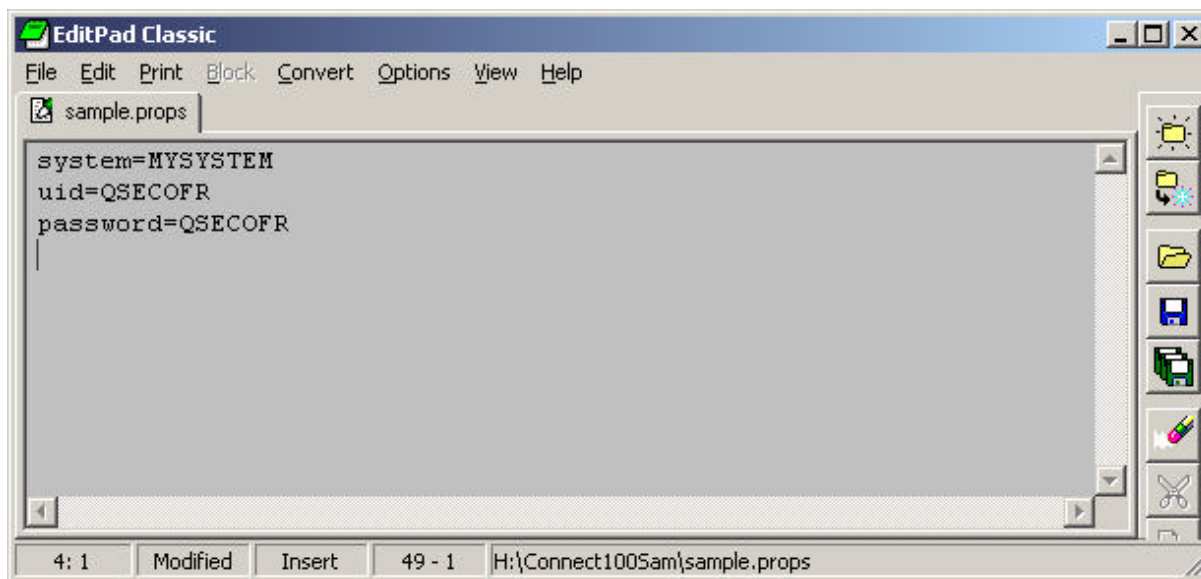
Create a temporary directory on the PC called “eprocure” and unzip the Client.zip file.

Connectors.zip

This file contains the 2 connectors used in this example and a sample.props file. PunchOutConnector.java and NewQuoteConnector.java. Consider using these files as base when starting a real implementation. Create a directory off of the root directory in IFS called /Connect110Sam/RemoteTest and unzip the Connectors.Zip file. **NOTE:** A directory must be created off of the root. There is a hard coded path in the NewQuoteConnector.java looking for a file called sample.props.

Edit the sample.props file and change all 3 fields. System should be set to the system name where the connectors have been installed. The uid should be set to a valid User ID on the system. And password should be a valid password for the User id entered.

This property file is used by the NewQuoteConnector for doing database access.



Deployment.zip

This file contains the ACD/PFM files used for deploying the PunchoutConnector.java and NewQuoteQonnecotor.java connectors. Consider using these files as base when doing starting a real implementation.

Unzip the Deployment.zip file to /QIBM/UserData/Connect110/RemoteTest/Connector. This directory was created automatically when the B2B instance was created.

Catalog.zip

This file contains the SupplierCatalog.html and QuoteTest.java servlet. SupplierCatalog.html is the simulated catalog. QuoteTest servlet is called when the PunchOutOrderRequest message is sent to the eProcurement simulator. This servlet will display the PunchoutOrder request in a browser window either URL or Base64 encoded depending on the system settings. In a real world setting the eProcurement application used by the client will process the PunchoutOrder message. For purposes of this example we will display it in a browser.

From the Catalog.zip file copy SupplierCatalog.html to
\\QIBM\UserData\WebASAdv\default\hosts\default_host\default_app\web

From the Catalog.zip file copy QuoteTest.class to <Mapped Drive Letter>:\QIBM\UserData\WebASAdv\default\hosts\default_host\default_app\servlets

Note: The QuoteTest.java file is provided only as a reference.

REMOTEEEX.savf

This file contains 2 database tables used by the NewQuoteConnector.class.

- RSAMPCAT is a file that simulates items in a catalog.
- RSAMPQUOTE is a table that contains items that were shopped for by a buyer and placed in this table from the catalog management software. Associated with these items is a Quote Number. This Quote Number is used by the NewQuoteConnector to look up the items purchased by the buyer in the catalog management software.

RSAMPCAT should be restored to library QGPL as follows.

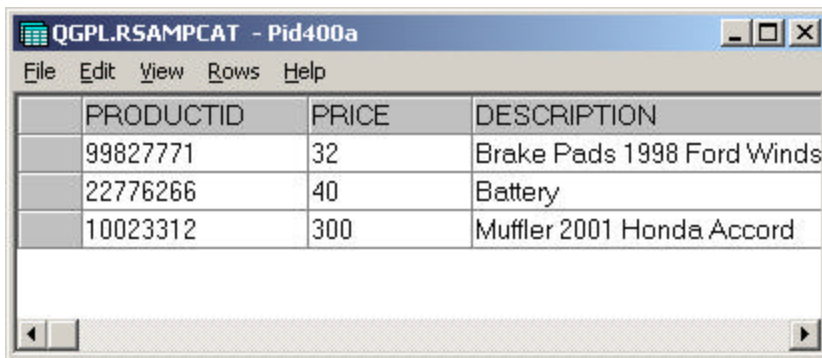
Issue the following command

- CRTSAVF FILE(QGPL/REMOTEEEX)
- Ftp the REMOTEEEX.SAVF file to the QGPL library on the iSeries. Make sure to set the session to Binary mode and then issue the command
 get REMOTEEEX.SAVF (replace)
- RSTOBJ OBJ(*ALL) SAVLIB(QGPL) DEV(*SAVF) SAVF(QGPL/REMOTEEEX)



Note: This assumes an FTP daemon is running on the PC. If an FTP daemon is not running on the PC, FTP from the PC to the iSeries.

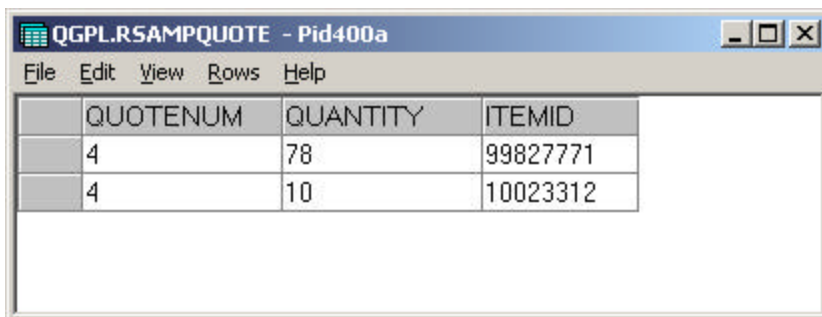
This should restore the tables. Use operations navigator to view the tables. They should contain the following information.



QGPL.RSAMPDAT - Pid400a

File Edit View Rows Help

PRODUCTID	PRICE	DESCRIPTION
99827771	32	Brake Pads 1998 Ford Winds
22776266	40	Battery
10023312	300	Muffler 2001 Honda Accord



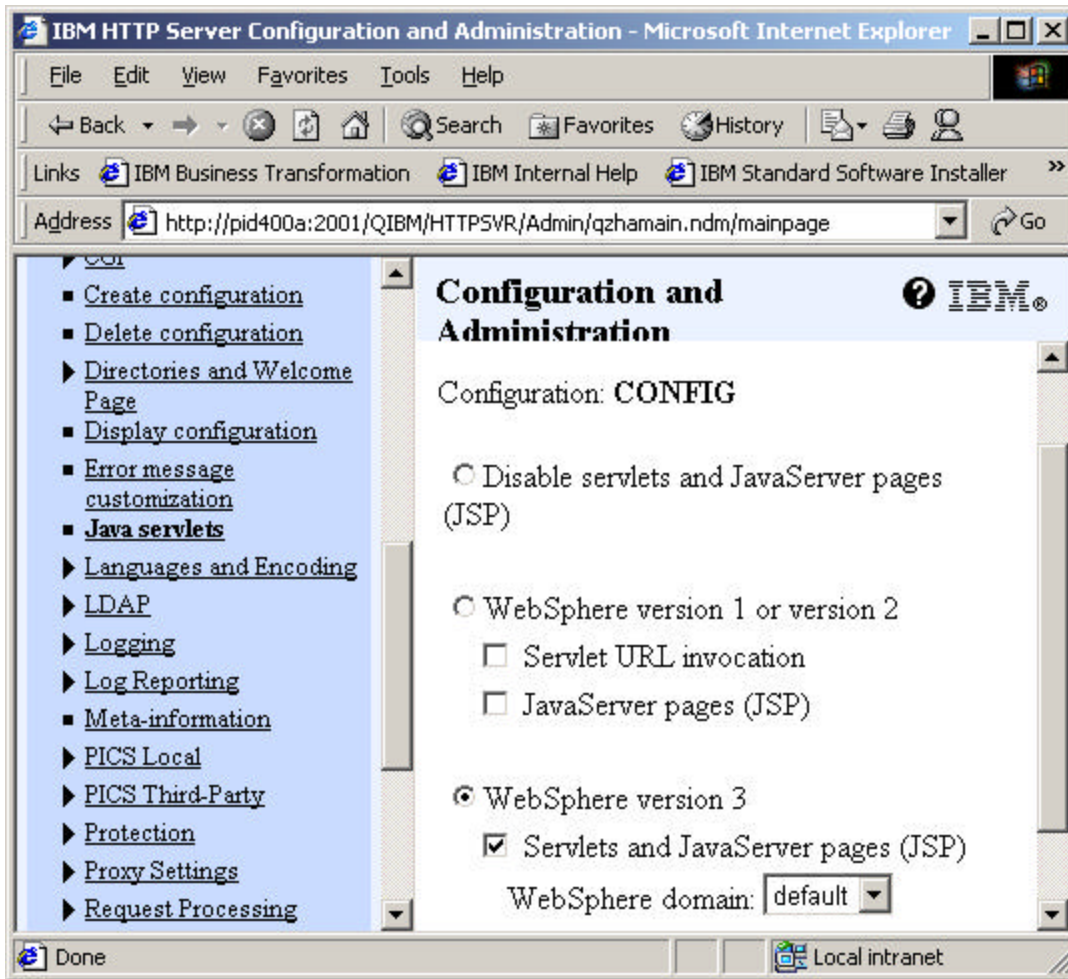
QGPL.RSAMPQUOTE - Pid400a

File Edit View Rows Help

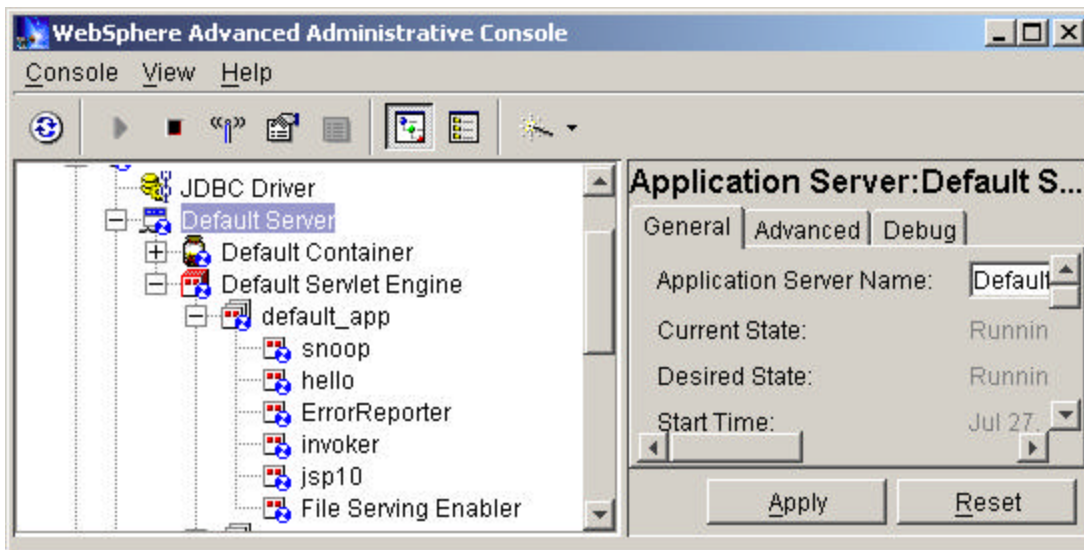
QUOTENUM	QUANTITY	ITEMID
4	78	99827771
4	10	10023312

Configure Separate HTTP/WAS

To properly display the SupplierCatalog.html and QuoteTest servlet information, configure an HTTP and WebSphere default instance to serve up these files. This can be done several ways. For purposes of this example HTTP server port 3497 is used. Make sure the Config instance is forwarding requests on to WebSphere as shown in the following screen shot.



Make sure the WAS default server has “invoker” and “File Serving Enabler” configured under the default_app Web Application.



From the Catalog.zip file copy SupplierCatalog.html to
\\QIBM\UserData\WebASAdv\default\hosts\default_host\default_app\web

From the Catalog.zip file copy QuoteTest.class to <Mapped Drive
Letter>:\QIBM\UserData\WebASAdv\default\hosts\default_host\default_app\servlets

After configuring the HTTP Server and Websphere, serve up the NewQuote.html file from a browser by typing in the path.

http://<SystemName>:<http port>/SupplierCatalog.html

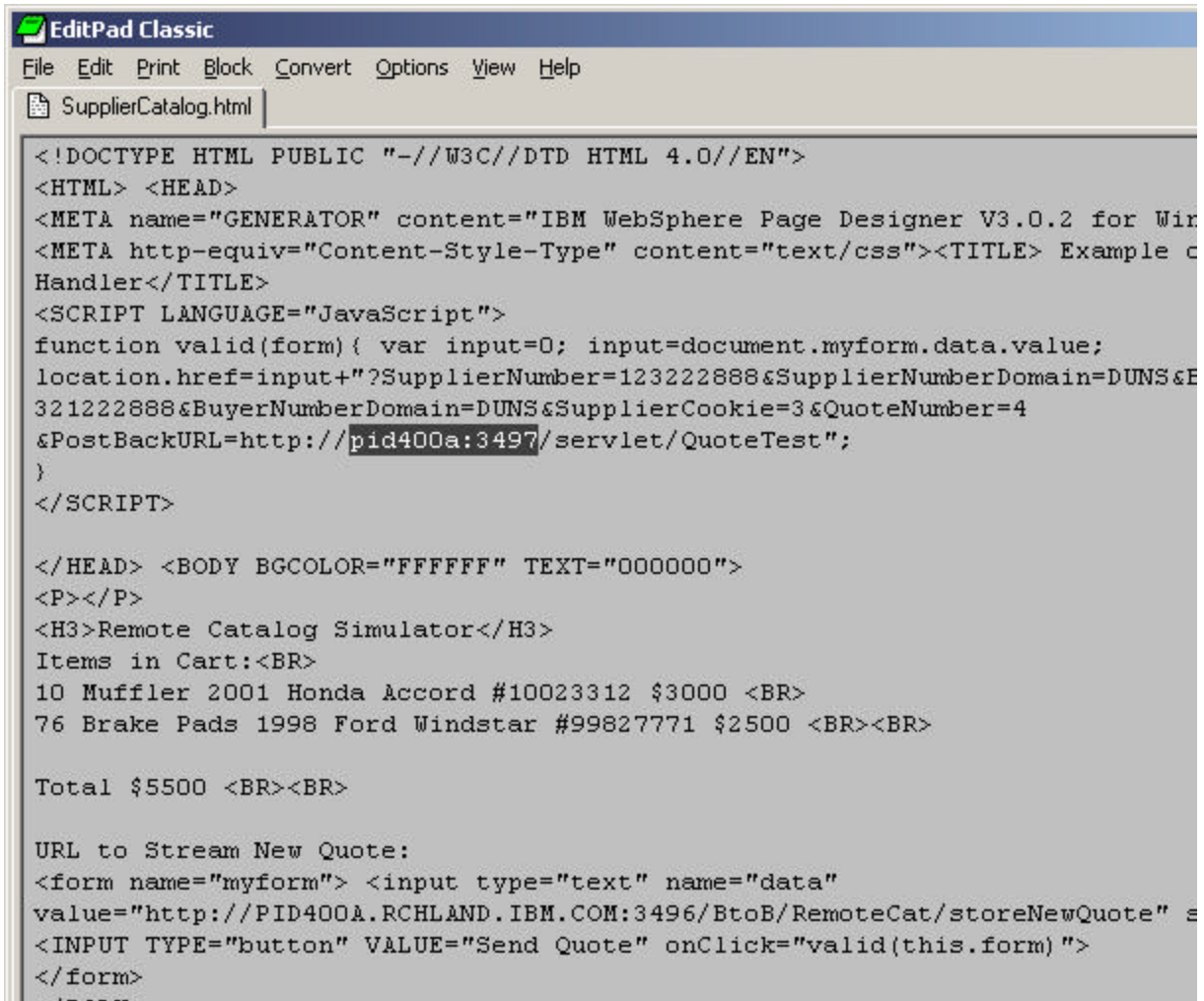
Note: Make sure the File Serving Enabler as been added to the WAS default_app. This allows WAS to serve HTML pages. A pass statement could be added to the HTML configuration as well

If this page cannot be served resolve the problem. The eProcurement simulator supplied with this example will not be able to display if it cannot be displayed from a browser.

Modify SupplierCatalog.html

The Supplier Catalog file will require 2 small modifications. Within the file is a PostbackURL value that is sent to the NewQuote connector. This Postbackurl needs to be change to the system name and HTTP port number where the QuoteTest.class can be served.

The highlighted line below indicates 1 of the changes to make to this file. Change it to the system name and HTTP port number that will serve up this servlet.



```

EditPad Classic
File Edit Print Block Convert Options View Help
SupplierCatalog.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML> <HEAD>
<META name="GENERATOR" content="IBM WebSphere Page Designer V3.0.2 for Win
<META http-equiv="Content-Style-Type" content="text/css"><TITLE> Example c
Handler</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function valid(form){ var input=0; input=document.myform.data.value;
location.href=input+"?SupplierNumber=123222888&SupplierNumberDomain=DUNS&F
321222888&BuyerNumberDomain=DUNS&SupplierCookie=3&QuoteNumber=4
&PostBackURL=http://pid400a:3497/servlet/QuoteTest";
}
</SCRIPT>

</HEAD> <BODY BGCOLOR="FFFFFF" TEXT="000000">
<P></P>
<H3>Remote Catalog Simulator</H3>
Items in Cart:<BR>
10 Muffler 2001 Honda Accord #10023312 $3000 <BR>
76 Brake Pads 1998 Ford Windstar #99827771 $2500 <BR><BR>

Total $5500 <BR><BR>

URL to Stream New Quote:
<form name="myform"> <input type="text" name="data"
value="http://PID400A.RCHLAND.IBM.COM:3496/BtoB/RemoteCat/storeNewQuote" s
<INPUT TYPE="button" VALUE="Send Quote" onClick="valid(this.form)">
</form>
  
```

When the SupplierCatalog.html page is displayed, a “Send Quote” button will be next to a text field containing a URL. This will be the URL where the NewQuote message is sent. The default in the example is “http://PID400A.RCHLAND.IBM.COM:3496/BtoB/RemoteCat/storeNewQuote”. This can be seen toward the bottom of the image above. bottom of the screen. Change the PID400A name to the name of the system where Connect for iSeries is installed and configured.

Configure the Sample

Create a Supplier

Create a Supplier and associate it to the marketplace using the information below. When the SupplierCatalog.html sends the NewQuote it will contain some of the buyer/supplier information presented here. If you configure it incorrectly authentication errors will occur within Connect for iSeries.

Supplier Login Information	
Supplier	Joe Supplier
DUNS	12322888
First Name	Joe
Last Name	Supplier
Supplier ID	123222888
Supplier ID Domain	DUNS
Log-on ID	123222888
Domain	DUNS
Password/Shared Secret	secret

Select the defaults for all of the other supplier information.

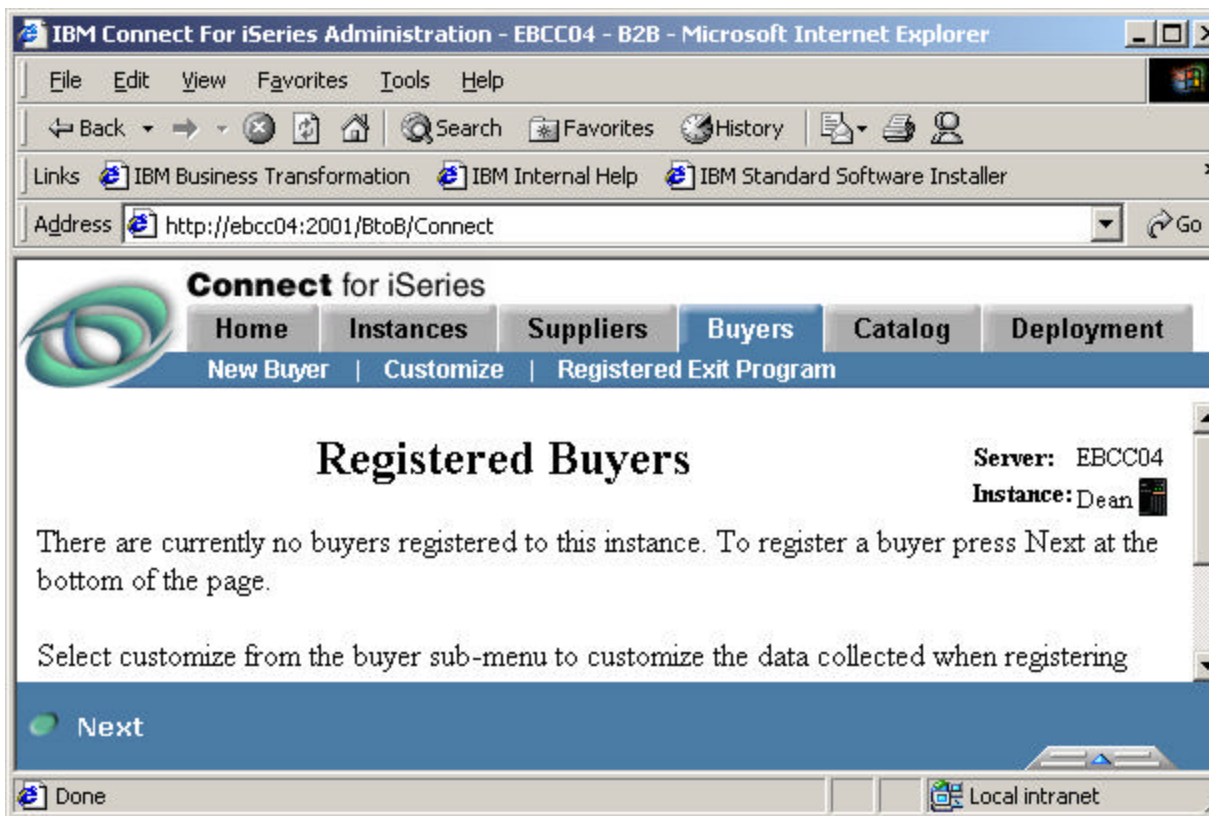
Configure Buyer Information

Buyer configuration for this example is more complicated than supplier configuration because this example uses the Custom Page feature available in Connect for iSeries 1.1

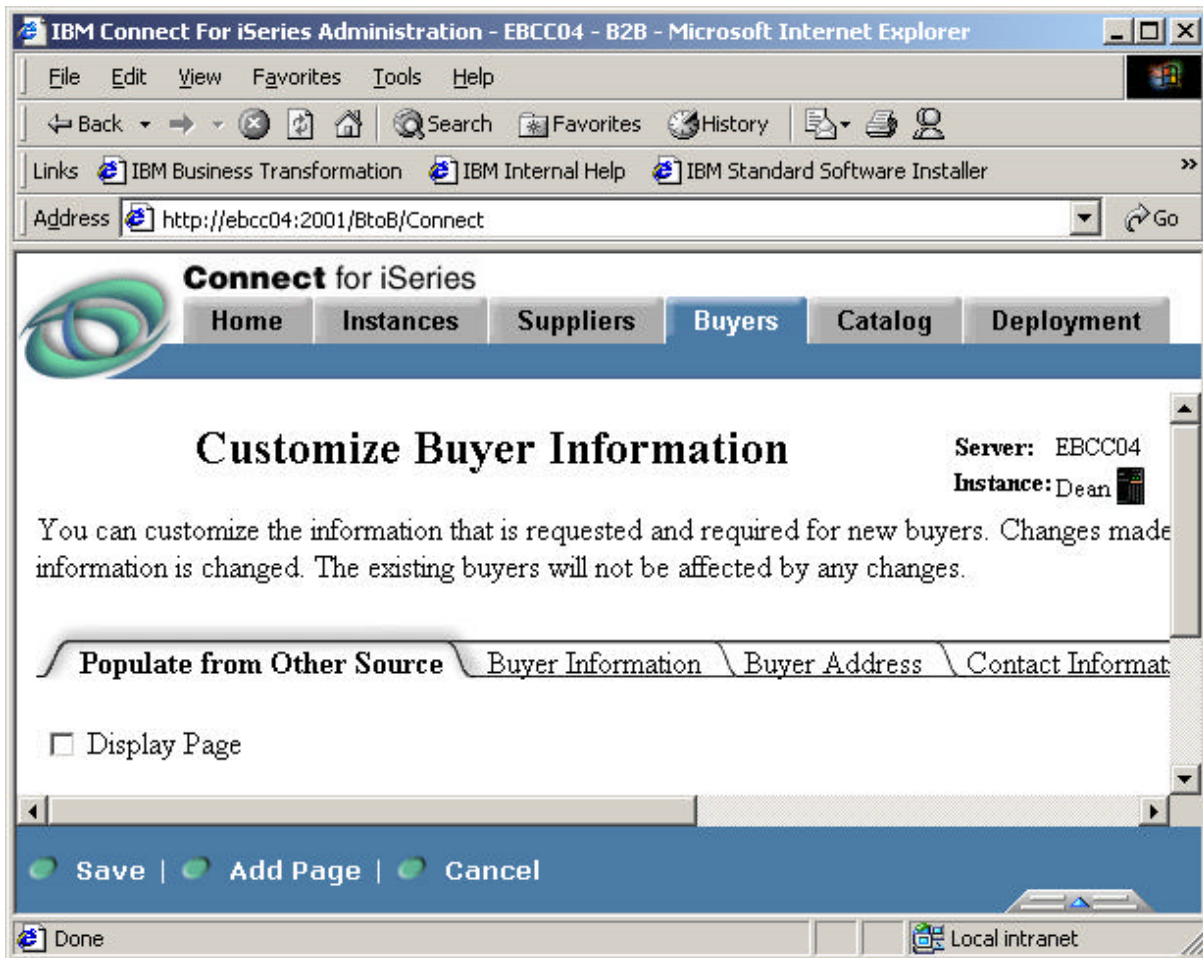
This feature exists as well within Supplier Configuration and you may need to utilize this type of functionality in a real implementation. For the purposes of this example we only use these feature in buyer creation.

Creating a Custom Page

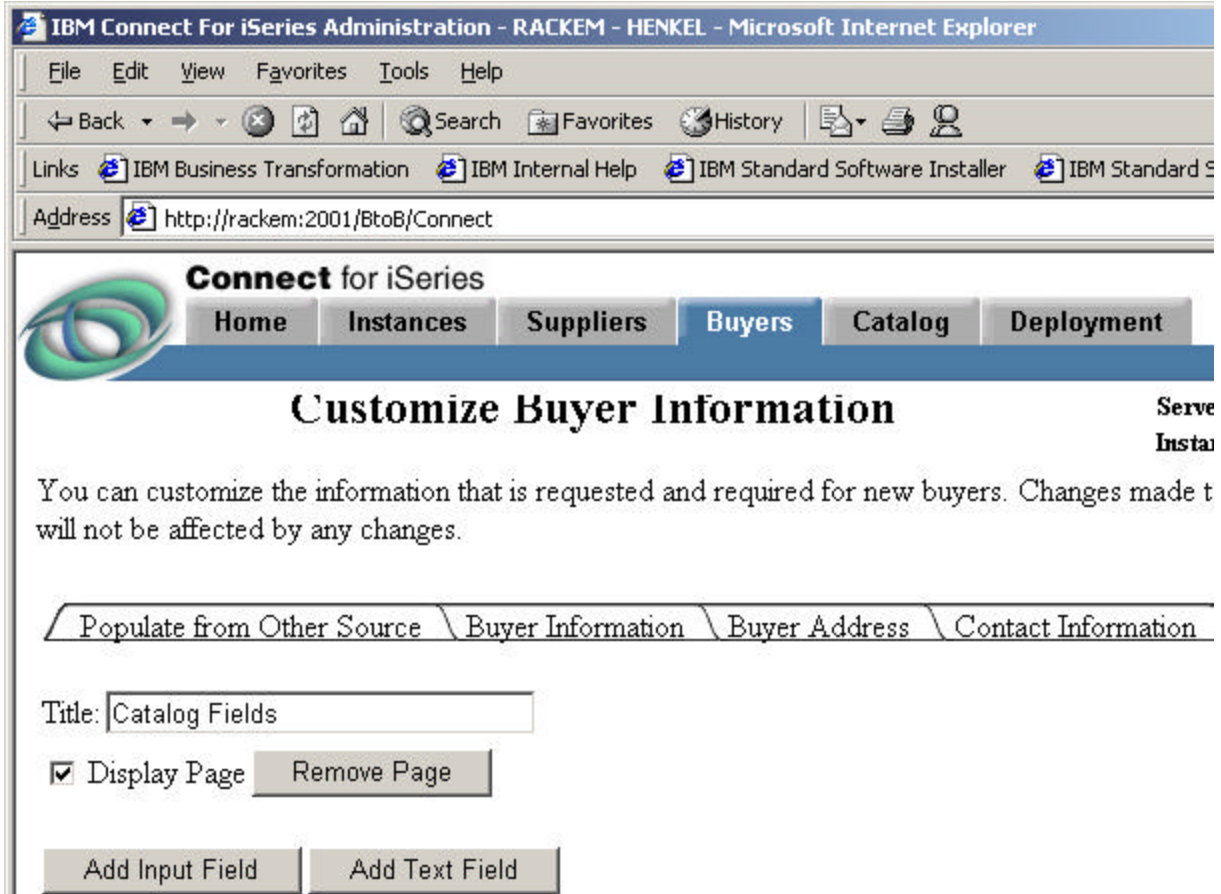
A custom page is how additional data is provided to Connect for iSeries. Our example creates a field that containing a URL. This data can then be accessed by the PunchOutConnector and returned in a PunchoutSetupResponse message. This field indicates where the e-procurement software should redirect. Go the buyers tab.



Click the Customize Link located near the top of the page.



Click Add Page



IBM Connect For iSeries Administration - RACKEM - HENKEL - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print

Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer IBM Standard S

Address http://rackem:2001/BtoB/Connect

Connect for iSeries

Home Instances Suppliers **Buyers** Catalog Deployment

Customize Buyer Information

Serve
Insta

You can customize the information that is requested and required for new buyers. Changes made t will not be affected by any changes.

Populate from Other Source \ Buyer Information \ Buyer Address \ Contact Information

Title:

Display Page

Change the Title of the screen to “Catalog Fields”
Click Add Input Field.

IBM Connect For iSeries Administration - EBCC04 - B2B - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print

Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer

Address http://ebcc04:2001/BtoB/Connect

Connect for iSeries

Home Instances Suppliers Buyers Catalog Deployment

Define Input Field Server: EBCC04 Instance: Dean

Descriptive label: * URL Redirect Field

Variable name: * URLRedirect

Type: * Any Characters

Length:

Default value:

Help URL:

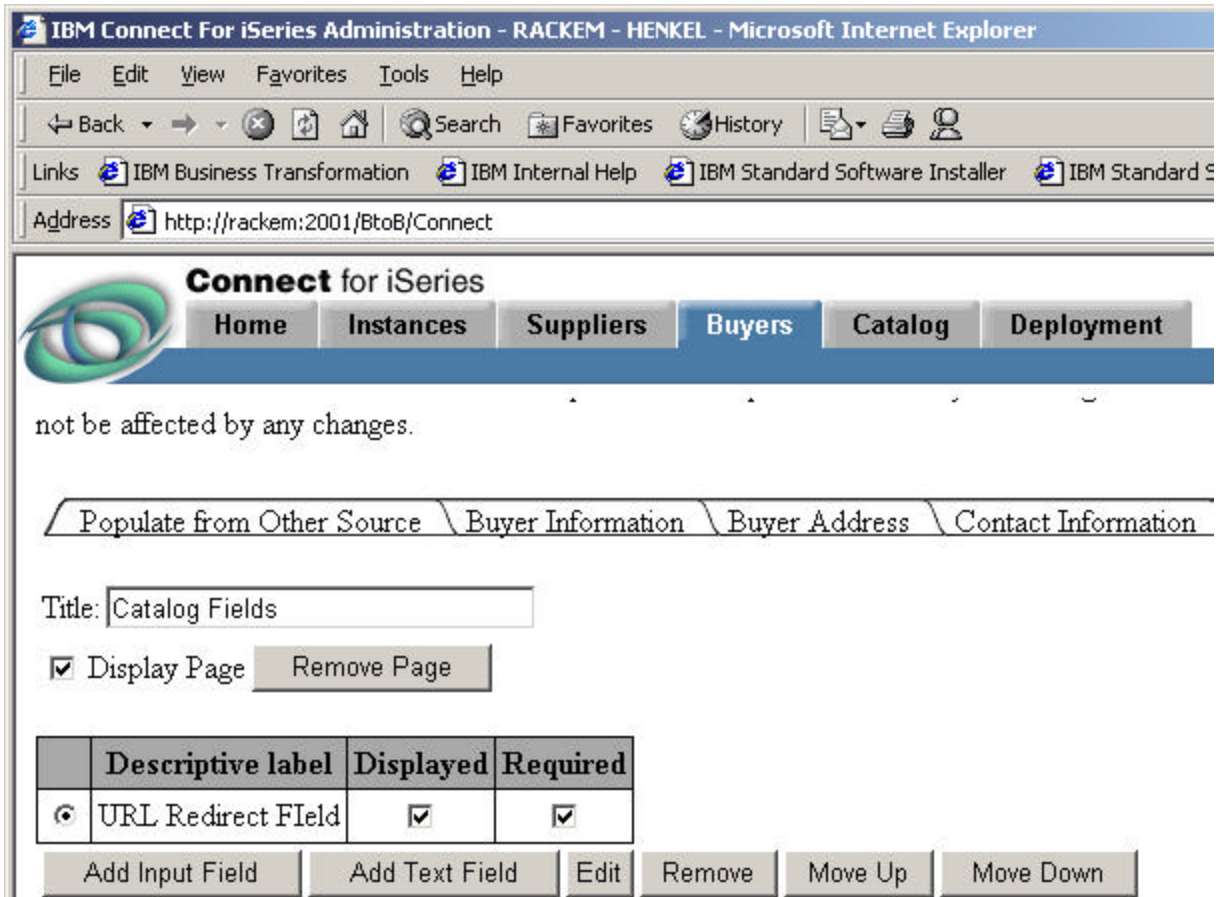
Help URL Text:

OK | Cancel

Local intranet

Enter "URL Redirect Field" in the Descriptive Label required field.
Enter "URLRedirect" in the Variable Name field.

Click Ok



IBM Connect For iSeries Administration - RACKEM - HENKEL - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print Print Preview User

Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer IBM Standard S

Address http://rackem:2001/BtoB/Connect

Connect for iSeries

Home Instances Suppliers **Buyers** Catalog Deployment

not be affected by any changes.

Populate from Other Source \ Buyer Information \ Buyer Address \ Contact Information

Title:

Display Page

	Descriptive label	Displayed	Required
<input checked="" type="radio"/>	URL Redirect Field	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Click the Required Check Box

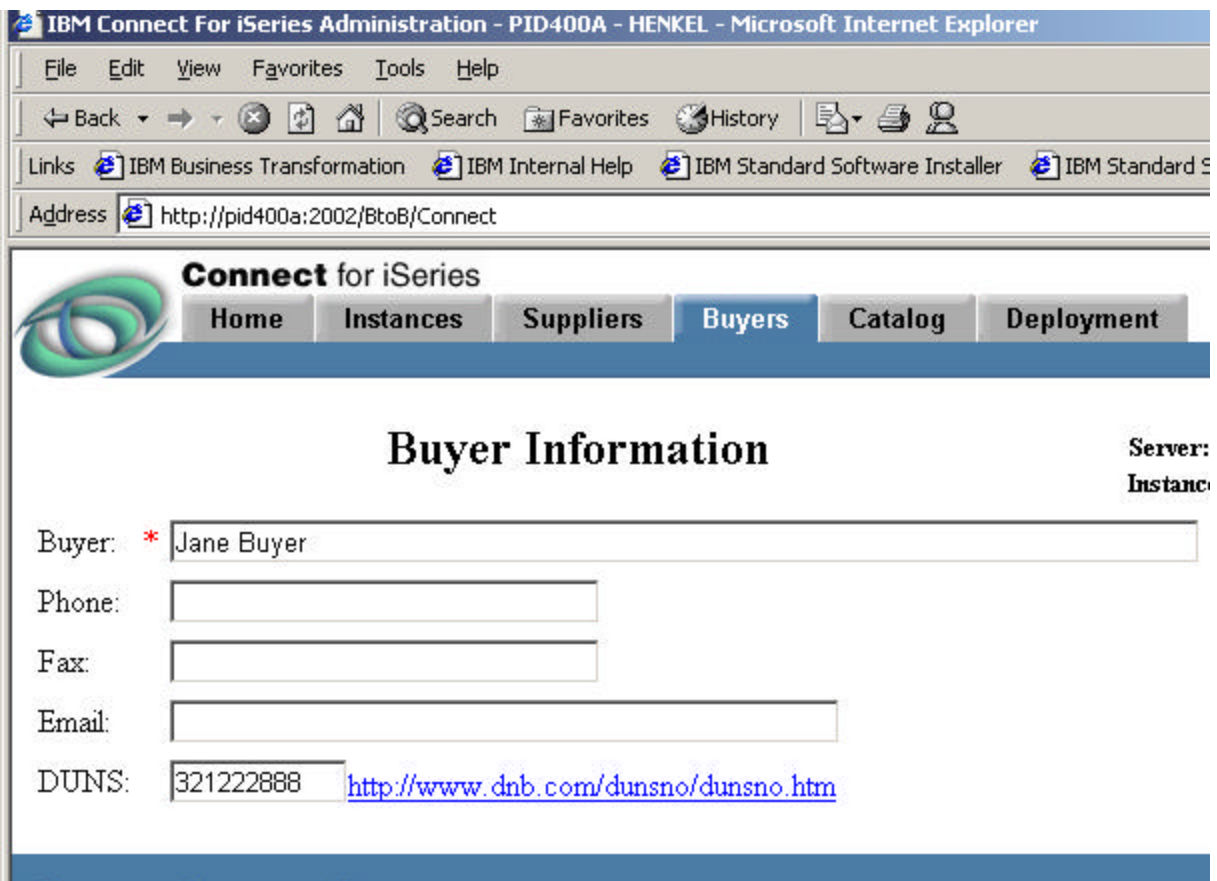
Click Save **NOTE:** Click SAVE or the changes will be lost!

A screen has now been added to the Buyer Configuration process. It will require a URLRedirect variable to have a value. The last page will say something about not having any registered buyers.

Click Next

The next step in the process would be to Create a buyer and associate it to a marketplace. Create a buyer using the following buyer information.

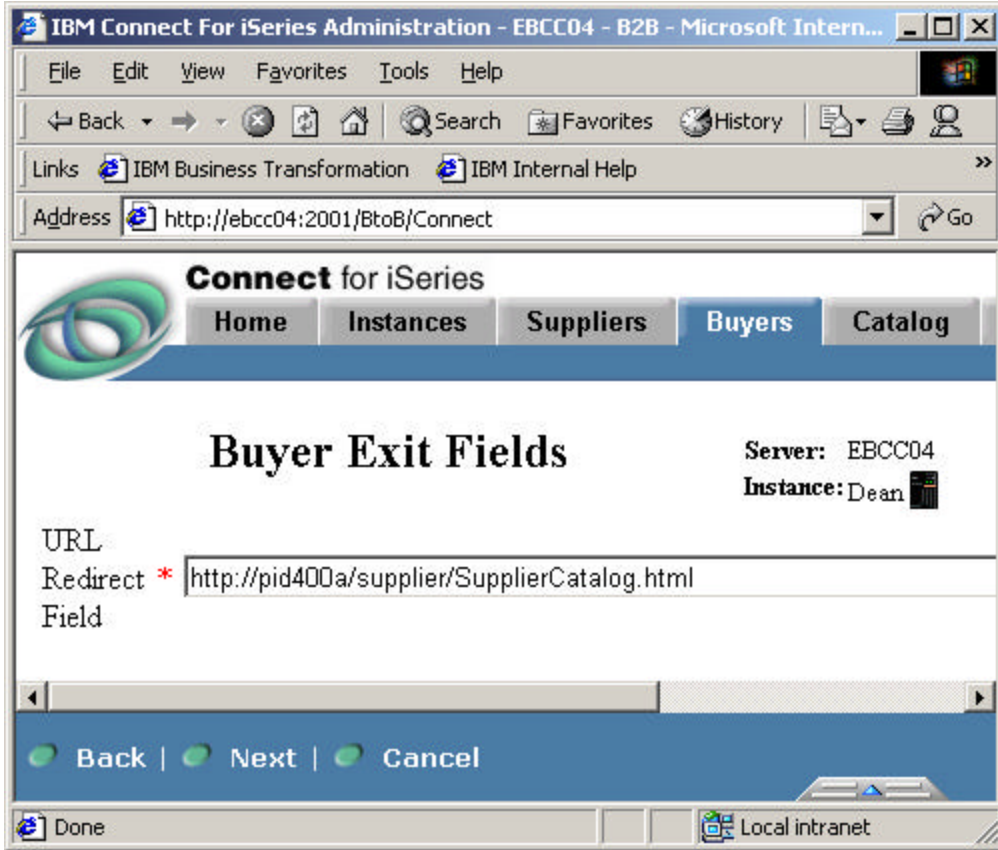
Buyer Login Information	
Buyer	Jane Buyer
DUNS	321222888
Buyer First	Jane
Buyer Last	Buyer
Buyer ID	321222888
Buyer ID Domain	DUNS



The screenshot shows a web browser window titled "IBM Connect For iSeries Administration - PID400A - HENKEL - Microsoft Internet Explorer". The address bar shows "http://pid400a:2002/BtoB/Connect". The page has a navigation menu with "Home", "Instances", "Suppliers", "Buyers", "Catalog", and "Deployment". The main content area is titled "Buyer Information" and includes a "Server:" and "Instance:" label. The form fields are as follows:

- Buyer: * Jane Buyer
- Phone:
- Fax:
- Email:
- DUNS: 321222888 <http://www.dnb.com/dunsno/dunsno.htm>

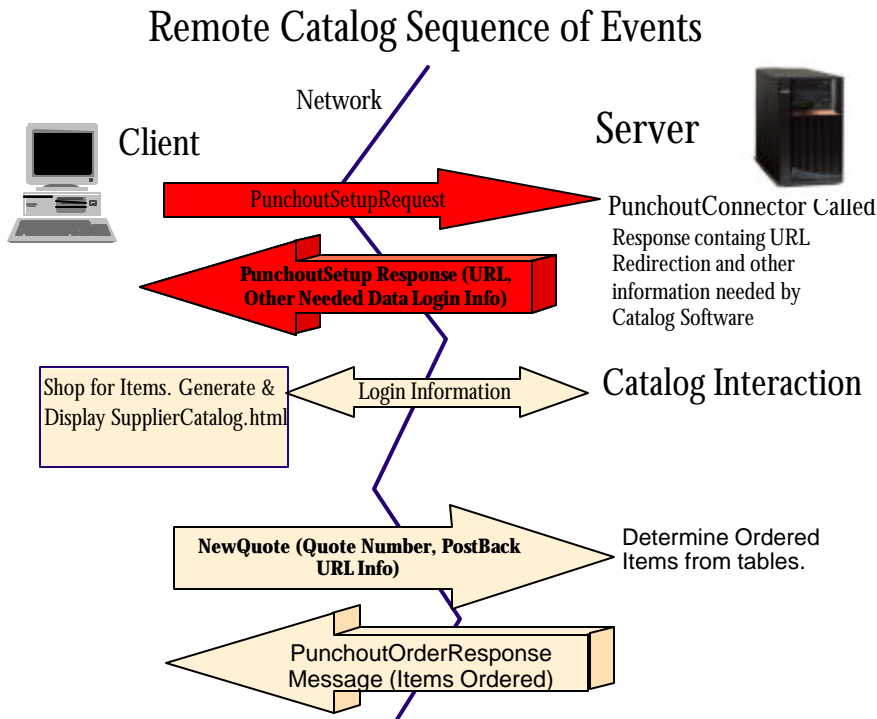
Move through buyer registration selecting the defaults by clicking Next. The last screen is the Customized screen created earlier. The URL Redirect Field, is the URL of the catalog management system. This custom field will be saved internally by Connect for iSeries and accessed by connectors using API's provided.



Enter the url of the configured HTTP/WAS instance used to display the SupplierCatalog.html file here.

PunchOutConnector

The PunchOutConnector example connector is very basic. The source takes as input a DUNS number, and uses the Connect API's to lookup the URL that should be sent back in the PunchoutSetup Message



The diagram above colored in **RED** graphically depict the work done by the PunchOutConnector. A real implementation may find it necessary to use input parameters that would allow access to shopper information in the database of the catalog management system. In addition it may be necessary to send back login information that can be passed as input data to the catalog URL. This example retrieves a URLRedirect field from the Connect for iSeries database. This field was created and saved when the Custom Page was created earlier in the example during buyer creation. To retrieve this information from the Connect for iSeries database, use the API's provided in com.ibm.connect.tools.tpa.api SupplierBuyerAPI's. Below is a code snippet from the PunchOutConnector that retrieves the URL data.

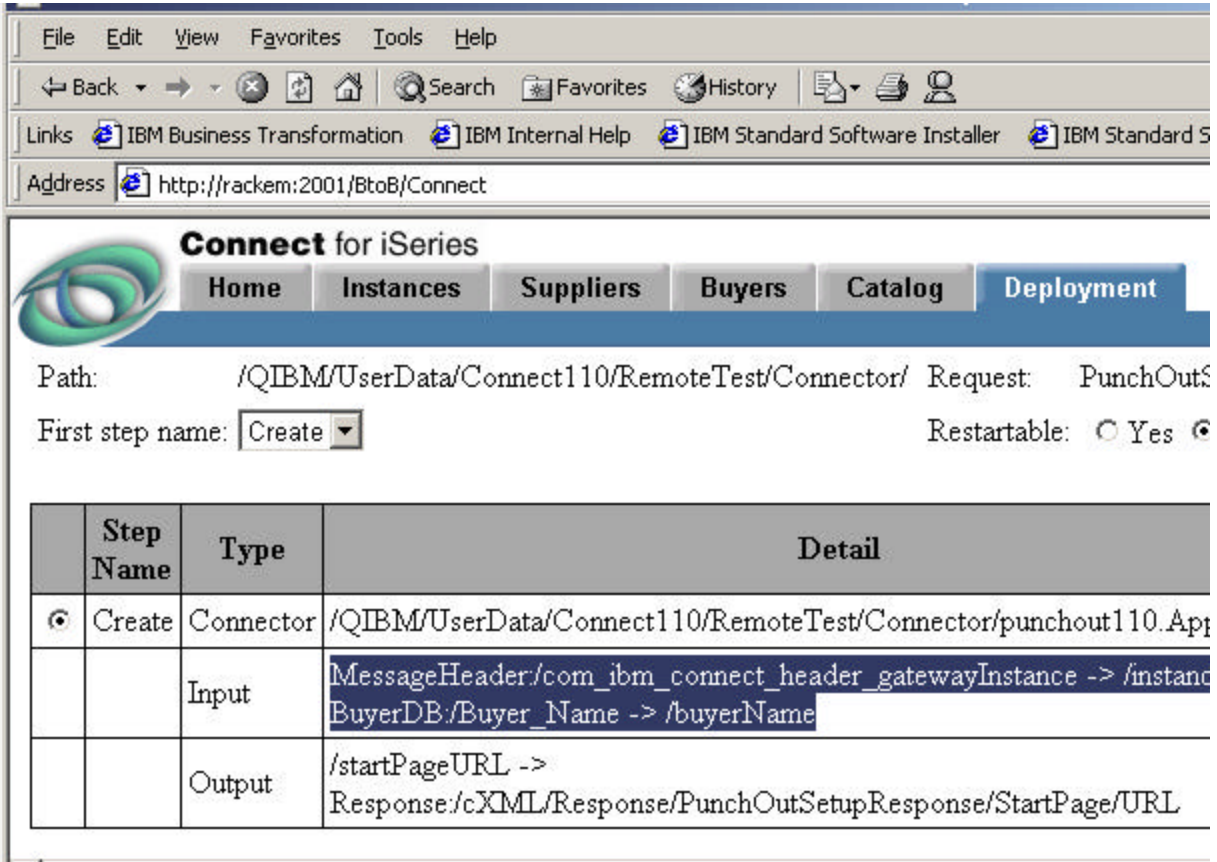
```

Field buyerNameFld = parms.getInputField("buyerName");
parms.bindInfieldToCursor(buyerNameFld);
String[] buyerNameVal = parms.getFieldAsString(buyerNameFld);
printIt("buyerName is", buyerNameVal[0]);
  
```

```

SupplierBuyerAPIs sbapi = new SupplierBuyerAPIs(instanceNameVal[0]);
String url=sbapi.getBuyerProperty(buyerNameVal[0],"URLRedirect");
  
```

Notice how the buyerNameVal is retrieved. The buyerName field is mapped to the BuyerDB:/Buyer Name as an input field to the PunchOutConnector. The following screen shot shows this mapping. Study the mappings for both the PunchOutConnector and NewQuote connector using the Connect for iSeries Deployment tools.



Step Name	Type	Detail
Create	Connector	/QIBM/UserData/Connect110/RemoteTest/Connector/punchout110.App
	Input	MessageHeader/com_ibm_connect_header_gatewayInstance -> /instanc BuyerDB:/Buyer_Name -> /buyerName
	Output	/startPageURL -> Response:/cXML/Response/PunchOutSetupResponse/StartPage/URL

Once the buyer name is obtained, may of the APIs can be called to access information about the buyer. In this example the URLRedirect field is retrieved.

```
SupplierBuyerAPIs sbapi = new SupplierBuyerAPIs(instanceNameVal[0]);
String url=sbapi.getBuyerProperty(buyerNameVal[0],"URLRedirect");
```

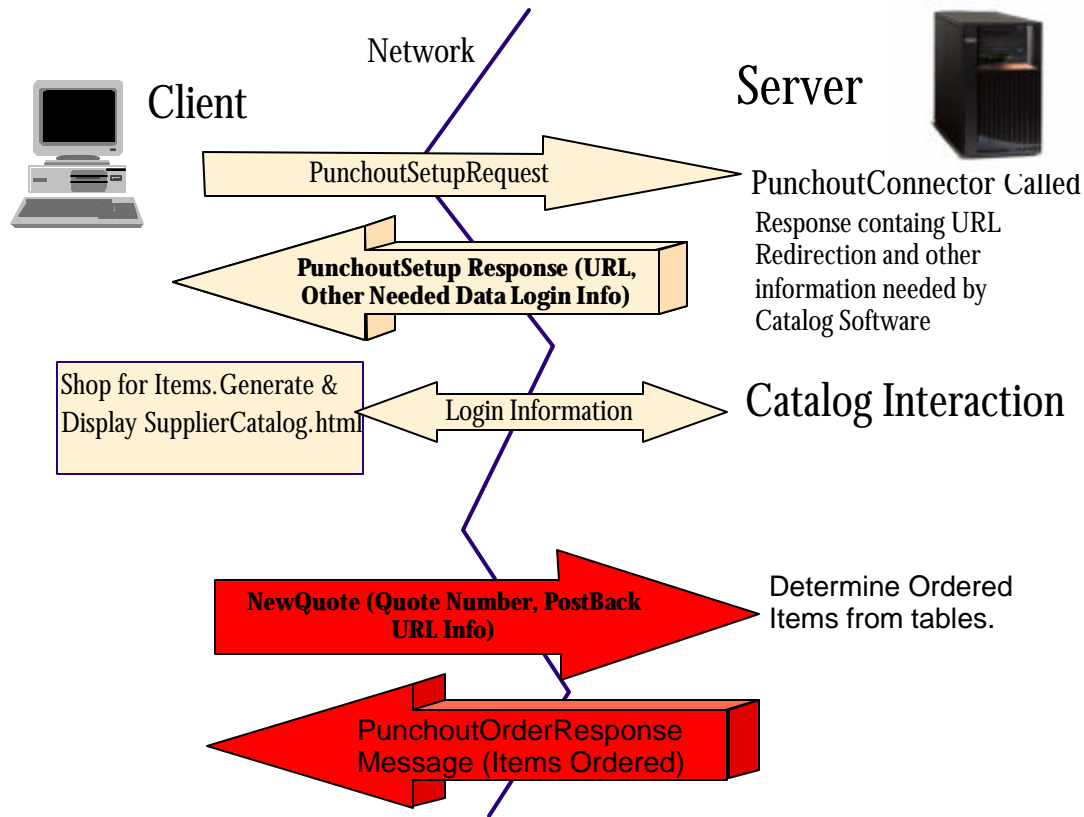
The following code demonstrates that sending back a url field in the PunchOutOrder message consists of binding the field to the cursor and setting the field. Additional field can be sent back in the PunchOutOrder message in a similar manner.

```
// Return the URL in the PunchoutSetup Response message.
Field urlFld = parms.getOutputField("startPageURL");
parms.bindOutfieldToCursor(urlFld);
parms.setFieldFromString(urlFld,url);
```


NewQuote Connector

It is within this NewQuote connector that information is sent regarding what items have been purchased in the catalog management system. This may be a simple ID which can be used to look up all items in the shopping cart, or it may be the actual shopping cart data itself. The example will access a table containing quote information. The quote number is passed in as input to the connector for lookup in the table. From this table the connector can then access the catalog information to determine the description and price of the items requested. The response will be a PunchoutOrder response Message.

Remote Catalog Sequence of Events



The diagram above colored in **RED** graphically depicts the work done by the NewQuoteConnector.

The example first needs to determine the quote number. This field is mapped as an extrinsic name value pair. Extrinsic fields are fields not native to the cXML protocol but added by Connect for iSeries. Once retrieved, the NewQuoteConnector looks up the items in a quote table to determine what products were selected from the catalog management software. Once the Quote is known, all of the items within the catalog tables can be accessed and mapped to the output fields of the NewQuoteConnector. In addition, the quoteTotal is calculated based on the price of each item and is sent back in a mapped output field.

It is the response from the NewQuoteConnector that causes a PunchoutOrder Message to be redirected back to the browser. Connect for iSeries will handle sending out the PunchoutOrder message based on the mapped output parameters for the NewQuoteConnector. Connect for iSeries will send PunchoutOrder message back to the PostBackURL which was input to the NewQuoteConnector.

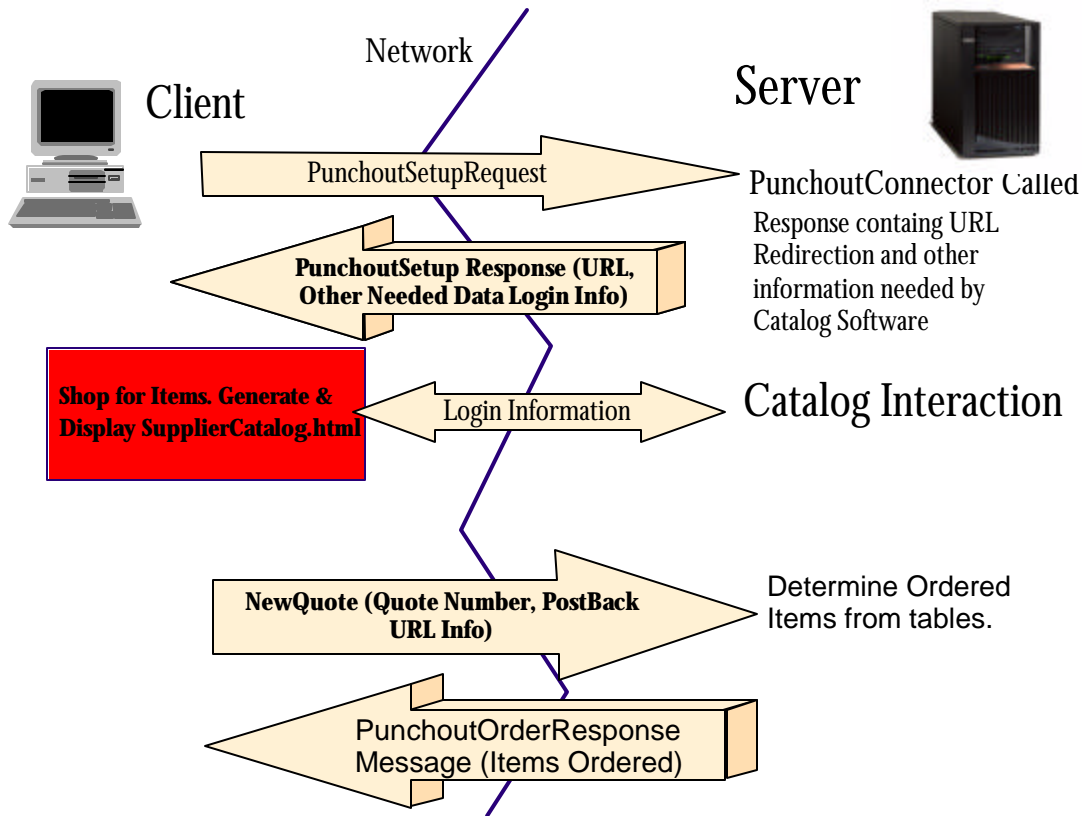
In this example we send back a total price, and a repeating array of Items that contain ID, Price, Description, and quantity. Study the code of NewQuoteConnector.java to see how the fields are sent back.

At some point it would be necessary to remove the items from the RSAMPQUOTE table for cleanup. This could be implemented in a variety of ways. For the purpose of the example, cleanup of the quotable was left out.

SupplierCatalog.html

The SupplierCatalog.html file is provided to simulate the Catalog Management software. It really only simulates what might be the last step in the process, the confirmation that the user wishes to reserve these items for possible purchase.

Remote Catalog Sequence of Events



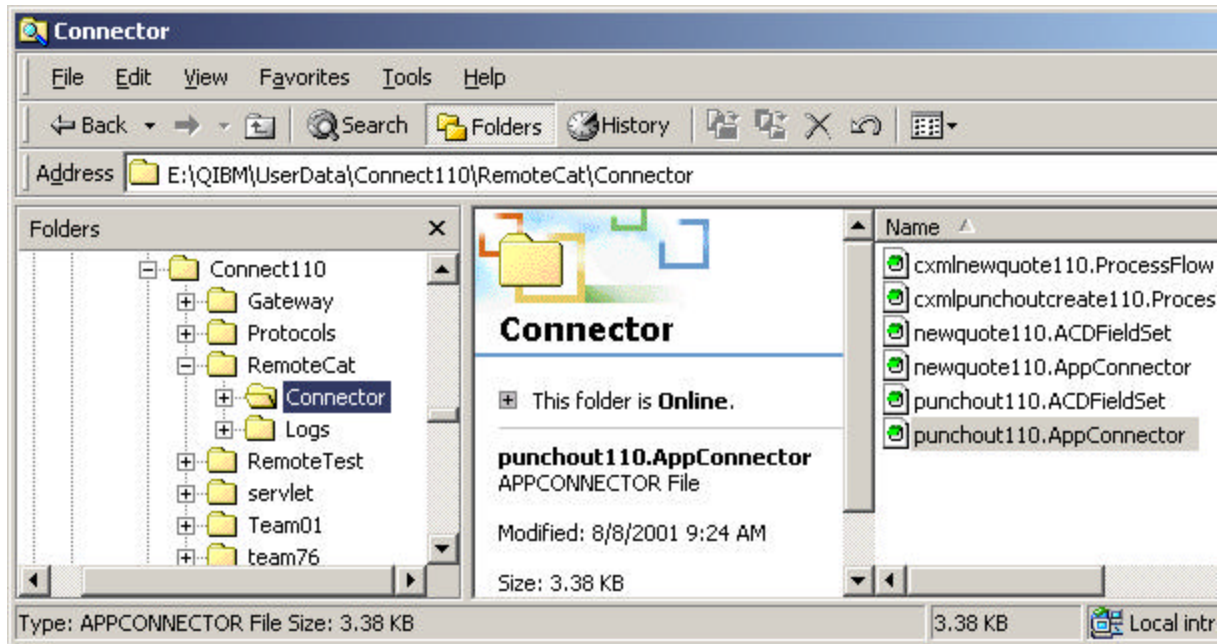
The diagram above colored in **RED** graphically depicts the work **SIMULATED** by the NewQuoteConnector. The SupplierCatalog.html page would be something like the buyer would see just prior to exiting out of the catalog management software. Within the SupplierCatalog.html page is a JavaScript function. This function is called when the SendQuote button is pressed. Study the JavaScript within the SupplierCatalog.html page. These fields can be mapped as extrinsic name value pairs to the NewQuoteConnector. The quote number field is used for looking up the quote and associated items in the RSAMQUOTE table. The PostBackURL parameter is used by the Gateway manager within Connect for iSeries as the location to send the Punchout Order Messages.

```
function valid(form){ var input=0; input=document.myform.data.value;
Location.href=input+"?SupplierNumber=123222888&SupplierNumberDomain=DUNS&BuyerNumber=321222888&BuyerNumberDomain=DUNS&SupplierCookie=3&Quote Number=4&PostBackURL=http://<SystemName>:<HTTP Port>/servlet/QuoteDone";
```

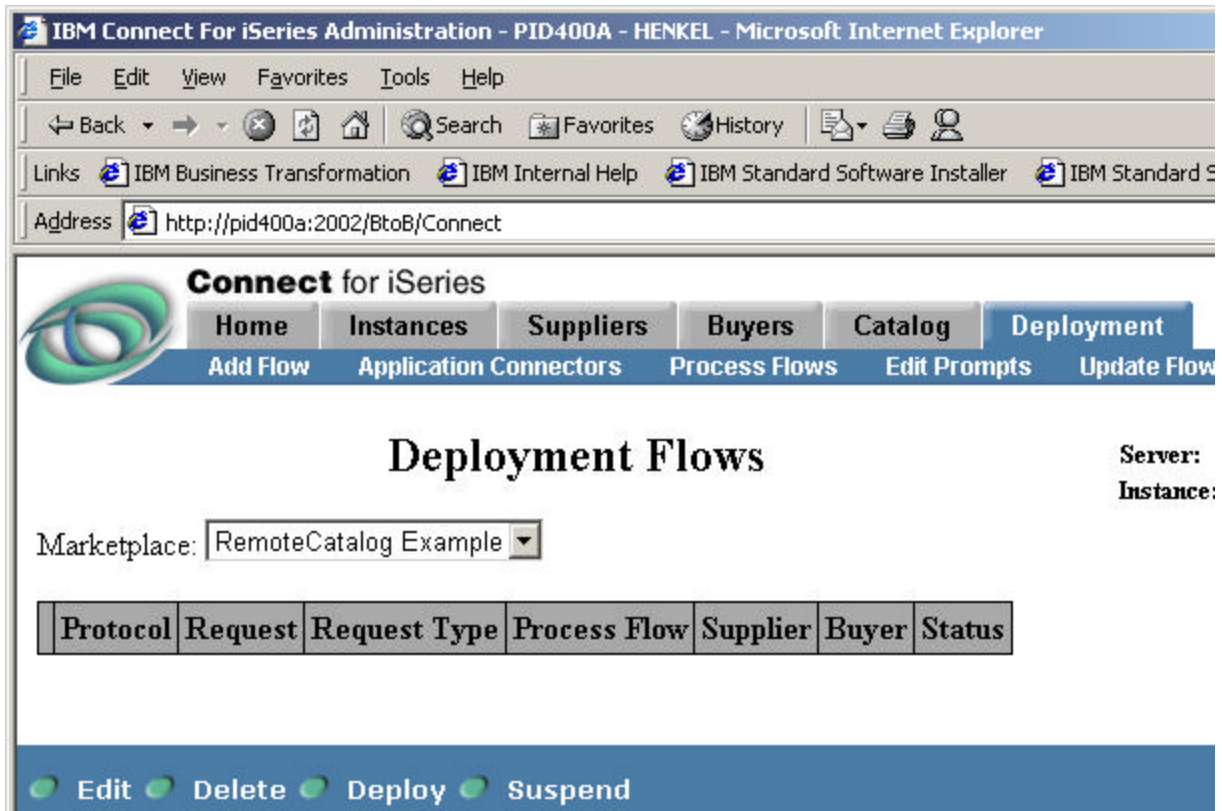

Deployment

Unzip the Deployment.zip file to /QIBM/UserData/Connect110/<InstanceName>/Connector.

The next steps will generate Meta data files used by the Flow Manager.



Click Deployment



Connect for iSeries

Home Instances Suppliers Buyers Catalog **Deployment**

Add Flow Application Connectors Process Flows Edit Prompts Update Flow

Deployment Flows

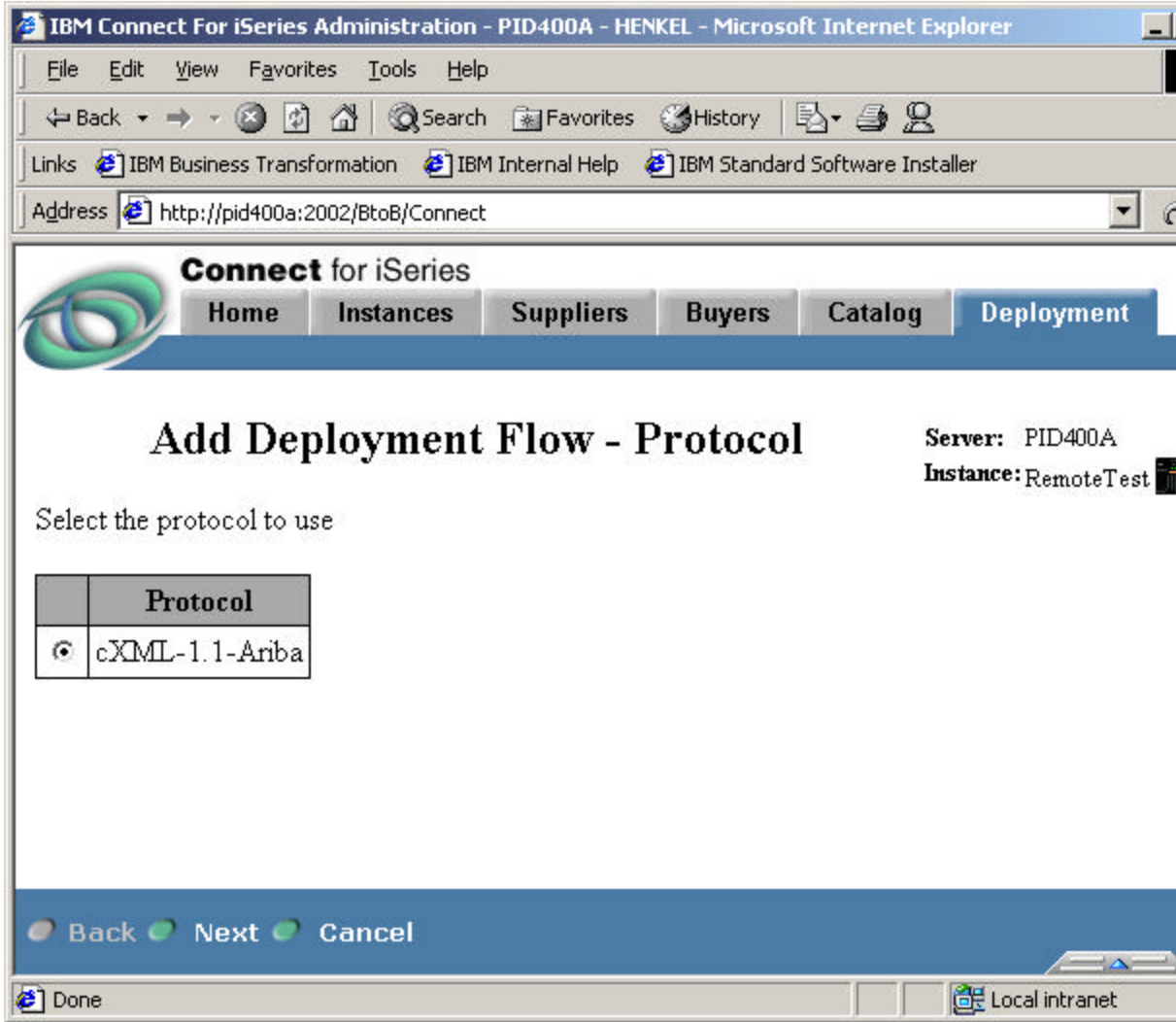
Server:
Instance:

Marketplace: RemoteCatalog Example

Protocol	Request	Request Type	Process Flow	Supplier	Buyer	Status
----------	---------	--------------	--------------	----------	-------	--------

Edit Delete Deploy Suspend

Click Add Flow



IBM Connect For iSeries Administration - PID400A - HENKEL - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print Copy Paste

Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer

Address http://pid400a:2002/BtoB/Connect

Connect for iSeries

Home Instances Suppliers Buyers Catalog **Deployment**

Add Deployment Flow - Protocol

Server: PID400A
Instance: RemoteTest

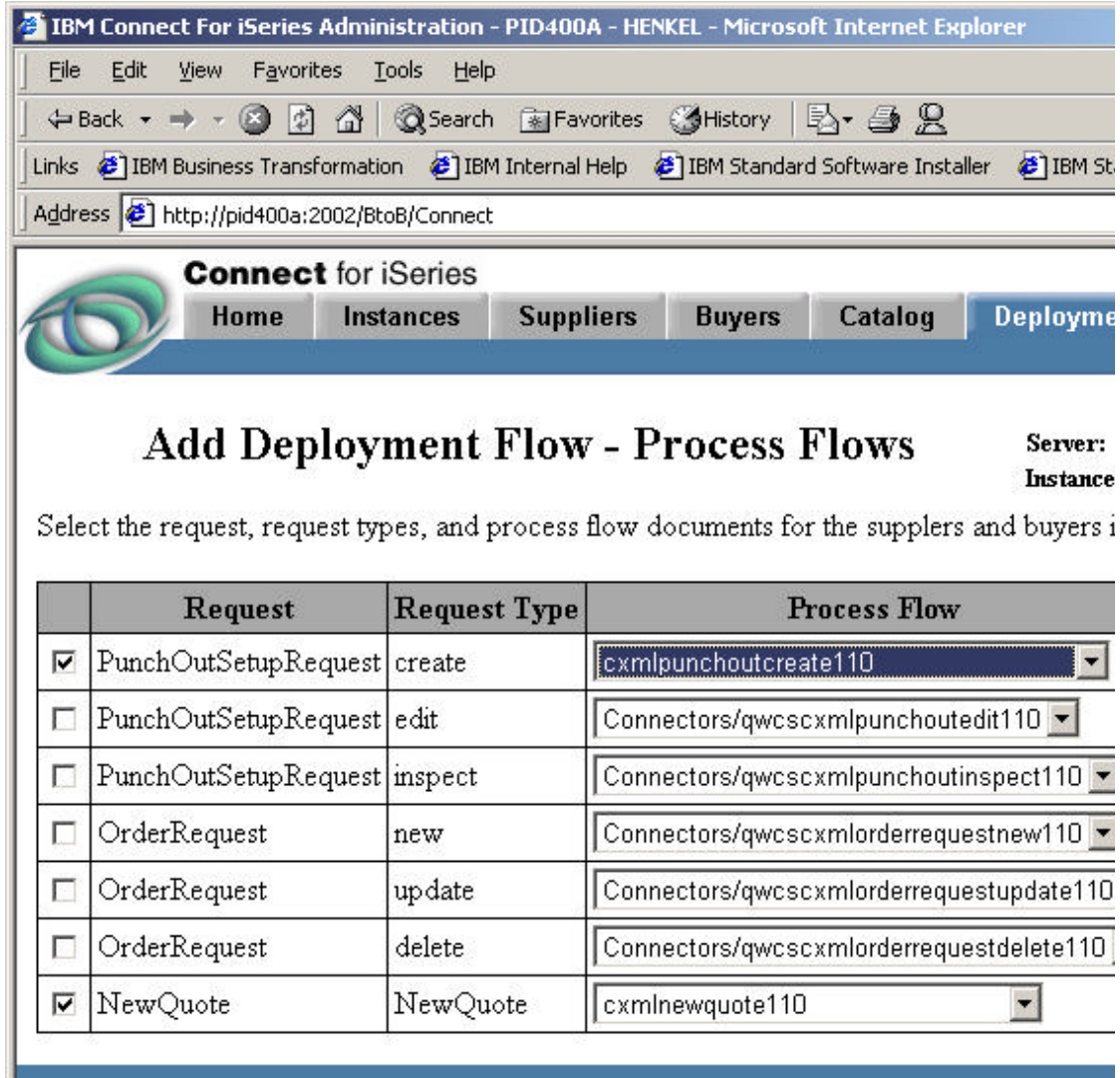
Select the protocol to use

Protocol
<input checked="" type="radio"/> cXML-1.1-Ariba

Back Next Cancel

Done Local intranet

Click Next



Add Deployment Flow - Process Flows Server:
Instance

Select the request, request types, and process flow documents for the suppliers and buyers i

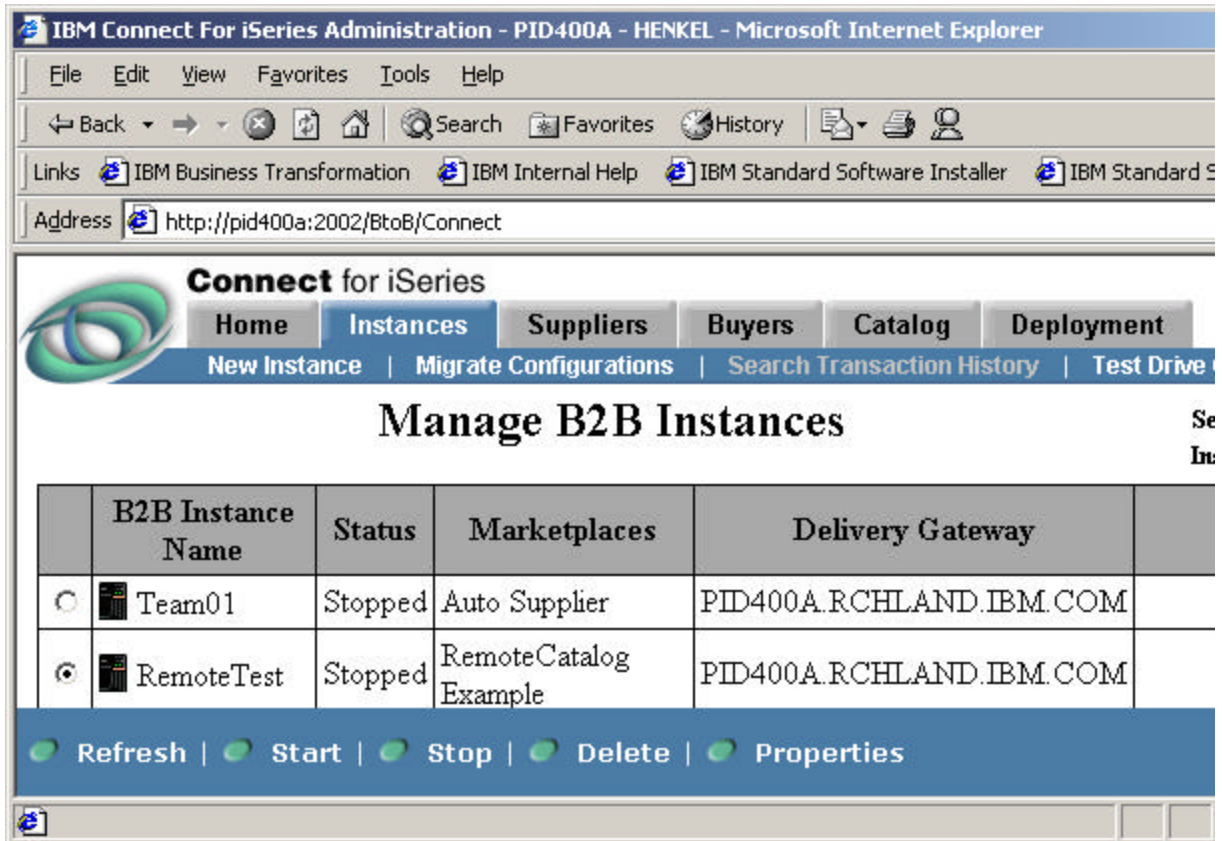
	Request	Request Type	Process Flow
<input checked="" type="checkbox"/>	PunchOutSetupRequest	create	cxmlpunchoutcreate110
<input type="checkbox"/>	PunchOutSetupRequest	edit	Connectors/qwscscxmlpunchoutedit110
<input type="checkbox"/>	PunchOutSetupRequest	inspect	Connectors/qwscscxmlpunchoutinspect110
<input type="checkbox"/>	OrderRequest	new	Connectors/qwscscxmlorderrequestnew110
<input type="checkbox"/>	OrderRequest	update	Connectors/qwscscxmlorderrequestupdate110
<input type="checkbox"/>	OrderRequest	delete	Connectors/qwscscxmlorderrequestdelete110
<input checked="" type="checkbox"/>	NewQuote	NewQuote	cxmlnewquote110

Check the PunchOutSetup Request and NewQuote check boxes. From the drop down combo boxes select cxmlpunchoutcreate110 for PunchoutSetupRequest and cxmlnewquote110 for NewQuote.

These are the process flow files that tell connect to call the PunchOutConnector.class and NewQuoteConnector.class files. Finish the deployment using the defaults. This will create a RuntimeMeta.xml and MetaMeta.xml file in the IFS directory
 /qibm/userdata/connect110/RemoteTest/connector.

Configure Path To Connectors & Turn off validation

It is required to provide the FlowManager with the classpath to the connector files so they can be loaded and their entry points called when a PunchoutSetup and NewQuote Message is received. Select the B2B instance.



Connect for iSeries

Home | **Instances** | Suppliers | Buyers | Catalog | Deployment

New Instance | Migrate Configurations | Search Transaction History | Test Drive

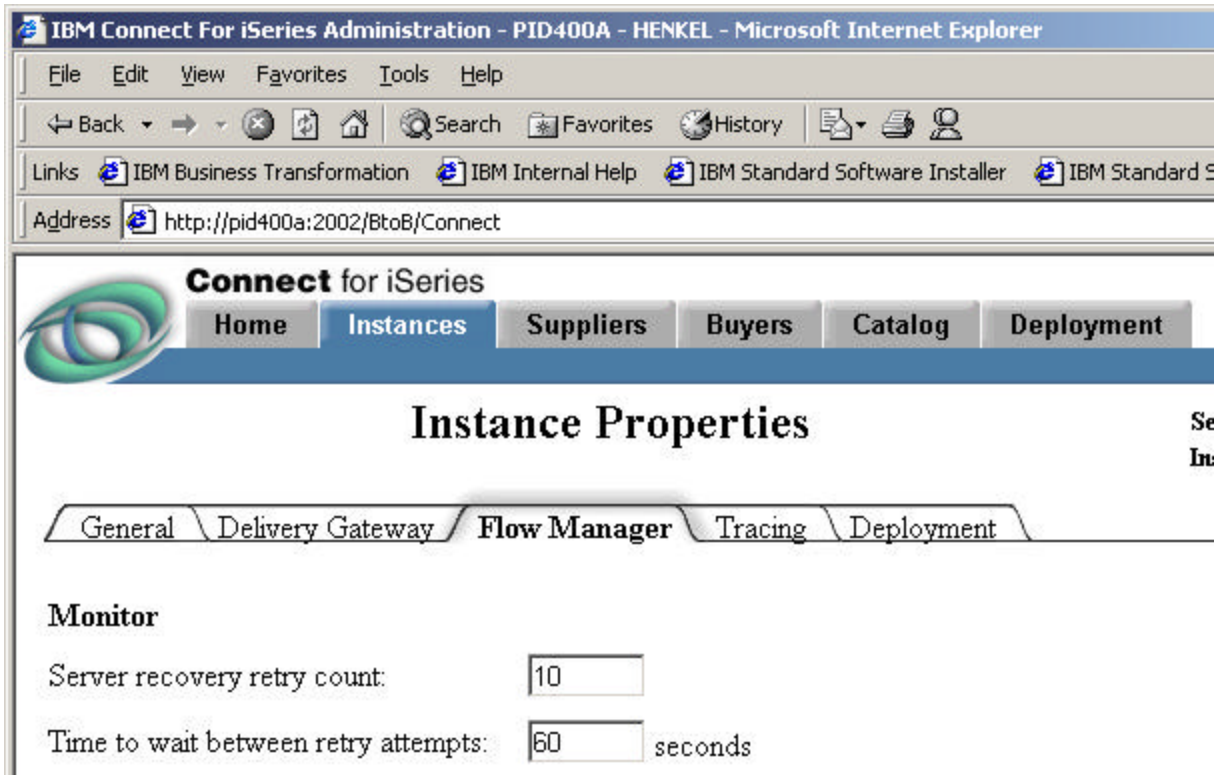
Manage B2B Instances

	B2B Instance Name	Status	Marketplaces	Delivery Gateway	
<input type="radio"/>	Team01	Stopped	Auto Supplier	PID400A.RCHLAND.IBM.COM	
<input checked="" type="radio"/>	RemoteTest	Stopped	RemoteCatalog Example	PID400A.RCHLAND.IBM.COM	

Refresh | Start | Stop | Delete | Properties

Click Properties

Click the Flow Manager Tab



IBM Connect For iSeries Administration - PID400A - HENKEL - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print

Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer IBM Standard S

Address http://pid400a:2002/BtoB/Connect

Connect for iSeries

Home Instances Suppliers Buyers Catalog Deployment

Instance Properties

General Delivery Gateway **Flow Manager** Tracing Deployment

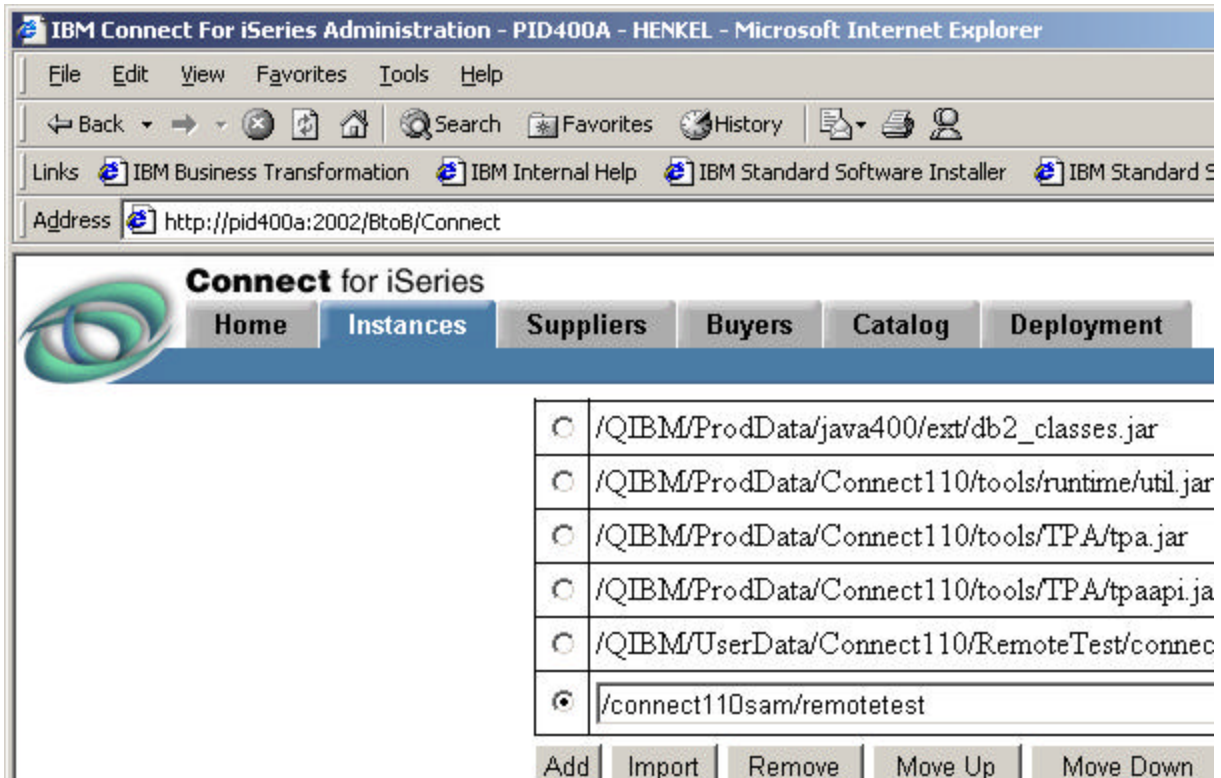
Monitor

Server recovery retry count:

Time to wait between retry attempts: seconds

Scroll down to the bottom of the screen

Click Add

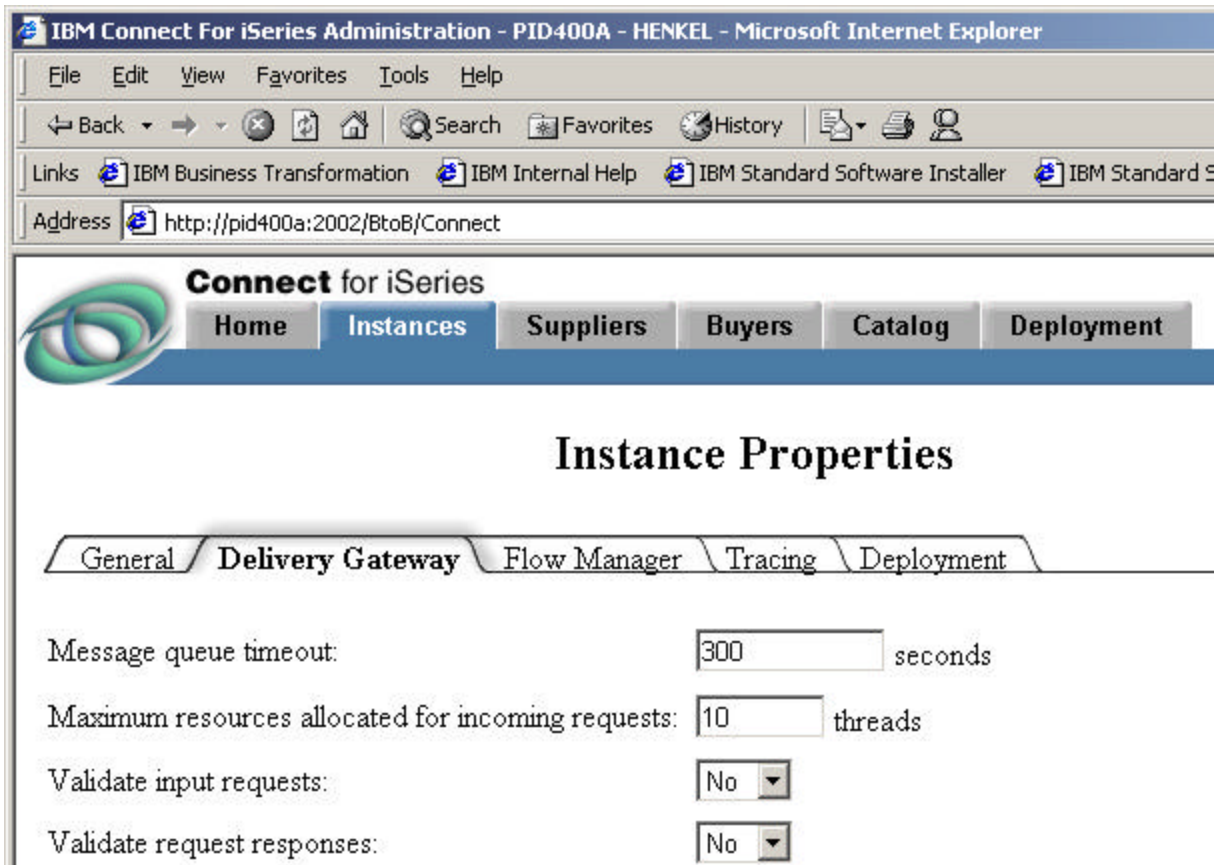


Type the path where Connectors.zip was unzipped. For this example it is “/connect110sam/remotetest”.

Click the Delivery Gateway Tab.

The example provided here sends back Price, Description, Quantity, and ID in a repeating array of Item Structures. This response is well formed xml but not valid because the example does not send back a Currency and Description Language. Because of this turn off validate request response..

The exercise of making the PunchOutOrder response valid is left to the reader.



The screenshot shows a Microsoft Internet Explorer browser window titled "IBM Connect For iSeries Administration - PID400A - HENKEL". The address bar shows "http://pid400a:2002/BtoB/Connect". The page content includes a navigation menu with "Home", "Instances", "Suppliers", "Buyers", "Catalog", and "Deployment". The main heading is "Instance Properties". Below this, there are tabs for "General", "Delivery Gateway", "Flow Manager", "Tracing", and "Deployment". The "Delivery Gateway" tab is active. The configuration settings are as follows:

Message queue timeout:	<input type="text" value="300"/>	seconds
Maximum resources allocated for incoming requests:	<input type="text" value="10"/>	threads
Validate input requests:	<input type="text" value="No"/>	
Validate request responses:	<input type="text" value="No"/>	

Make sure Validate input request is set to No.

Make sure Validate request responses is set to No.

Click OK

Note: Click OK the changes will not be saved!

Start the B2B Instance

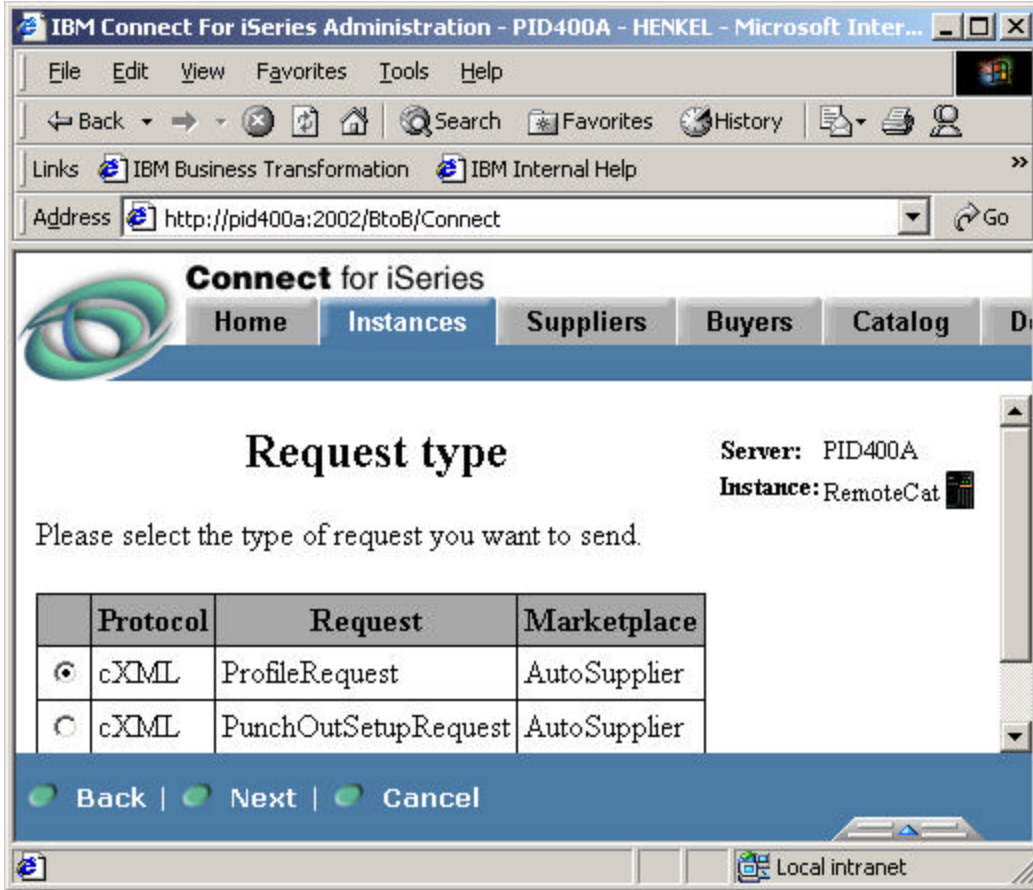
Start the B2B instance. Everything should be configured and ready to accept PunchoutOrder and NewQuote requests. Before we attempt to send Punchout requests verify a couple of things.

- Using the Test Drive Connect utility from the browser, stream a ProfileRequest message to the B2B instance.
- Using the Test Drive Connect utility from the browser stream a PunchoutSetup request message to the B2B instance
- Verify that SupplierCatalog.html can be served up from a browser

Stream a ProfileRequest Message

Connect for iSeries 1.1 has a very nice feature built into the tool called Test Drive Connect. To verify that everything is installed, configured, and started, stream a ProfileRequest message to the B2B instance. To stream profile request to the B2B instance click Test Drive Connect from the B2B Instances page.

Move through the wizard and select ProfileRequest



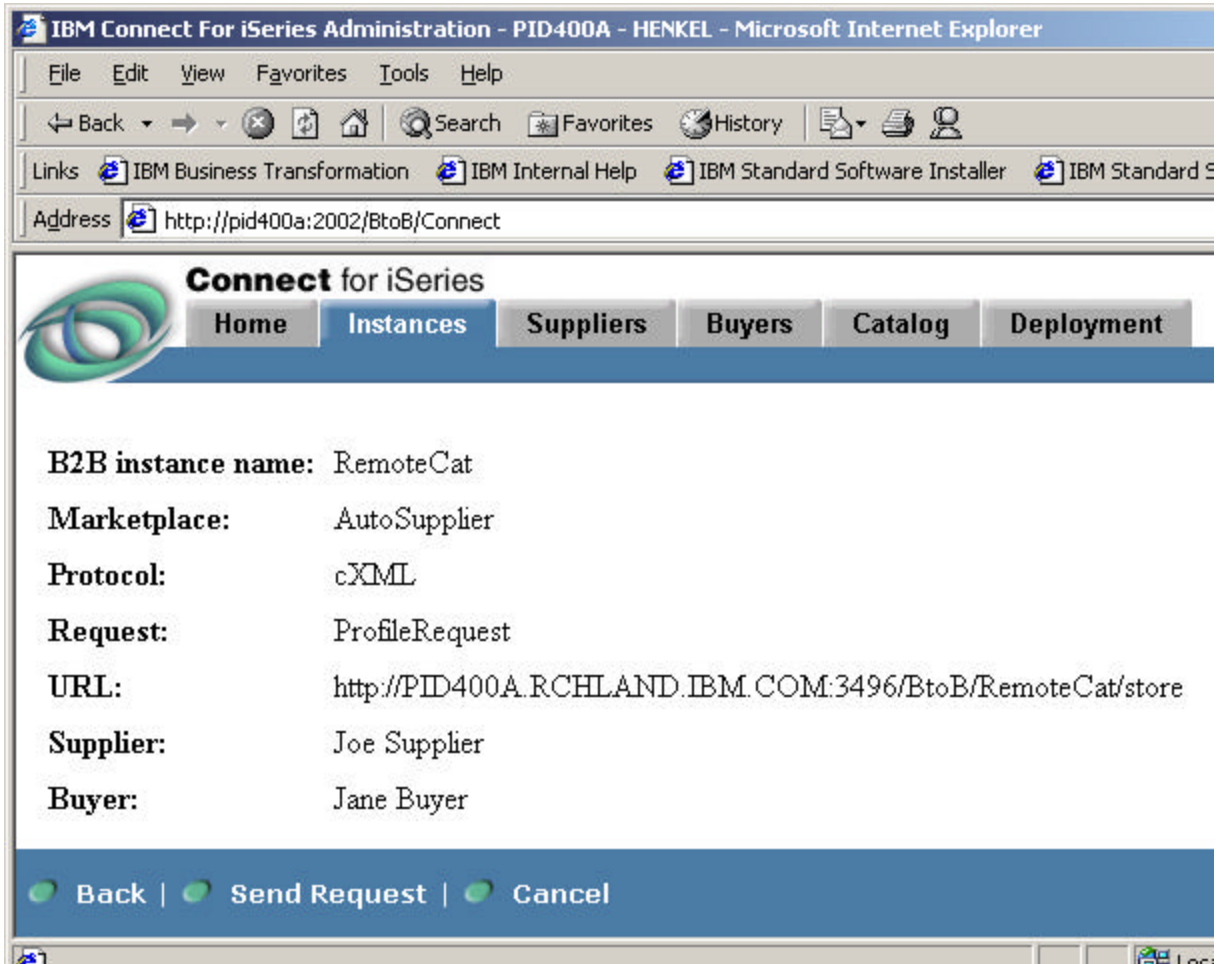
Continue moving through the screens by pressing next.

When selecting a supplier, type in the Password/Shared secret then press next. Following this example the password is “secret”. This is case sensitive.



Continue moving through the wizard accepting the defaults.

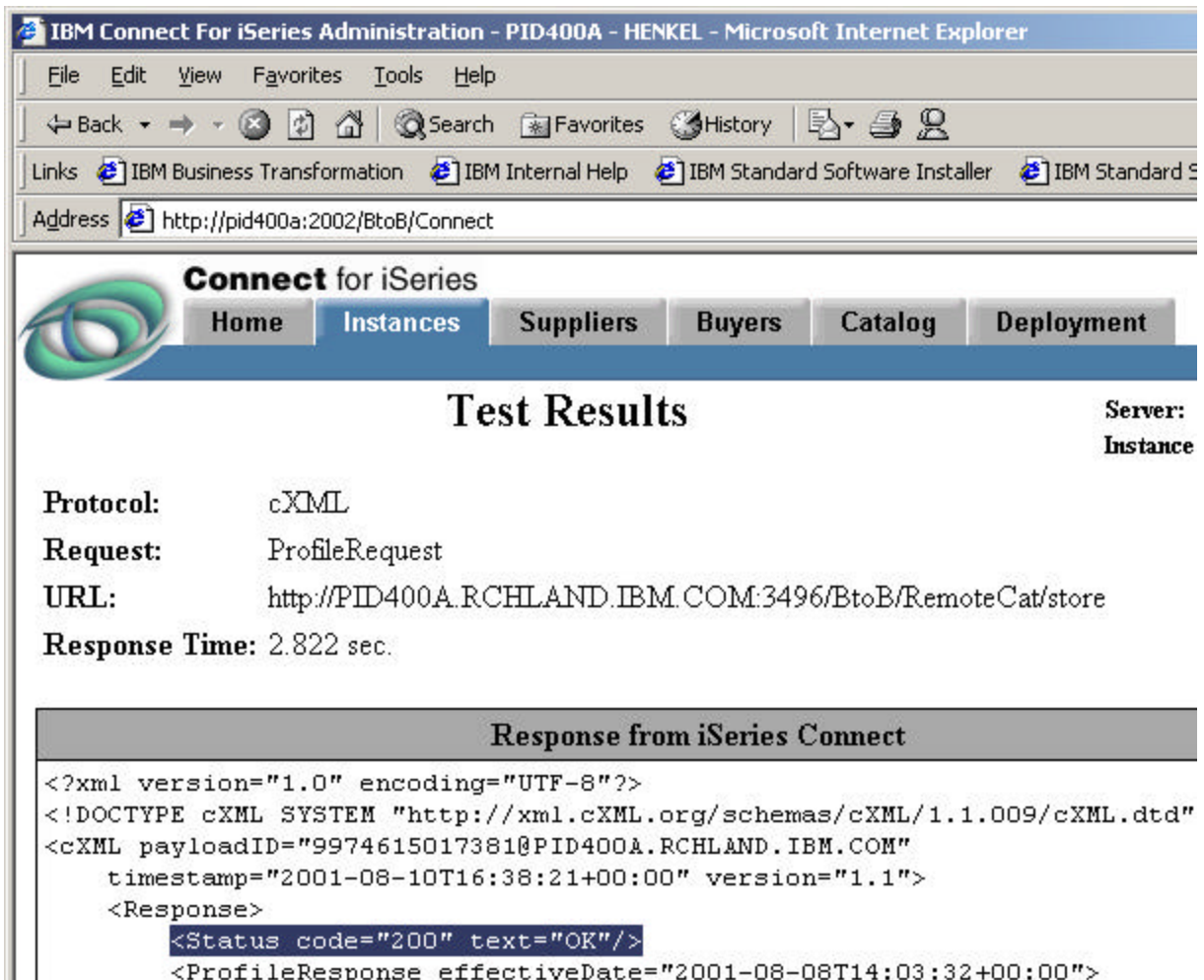
Eventually a screen similar to the following will be displayed



Press the Send Request.

This will send a profile request message to the instance.

If everything worked successfully the a screen similar to the following will be displayed.



Test Results Server:
Instance

Protocol: cXML
Request: ProfileRequest
URL: http://PID400A.RCHLAND.IBM.COM:3496/BtoB/RemoteCat/store
Response Time: 2.822 sec.

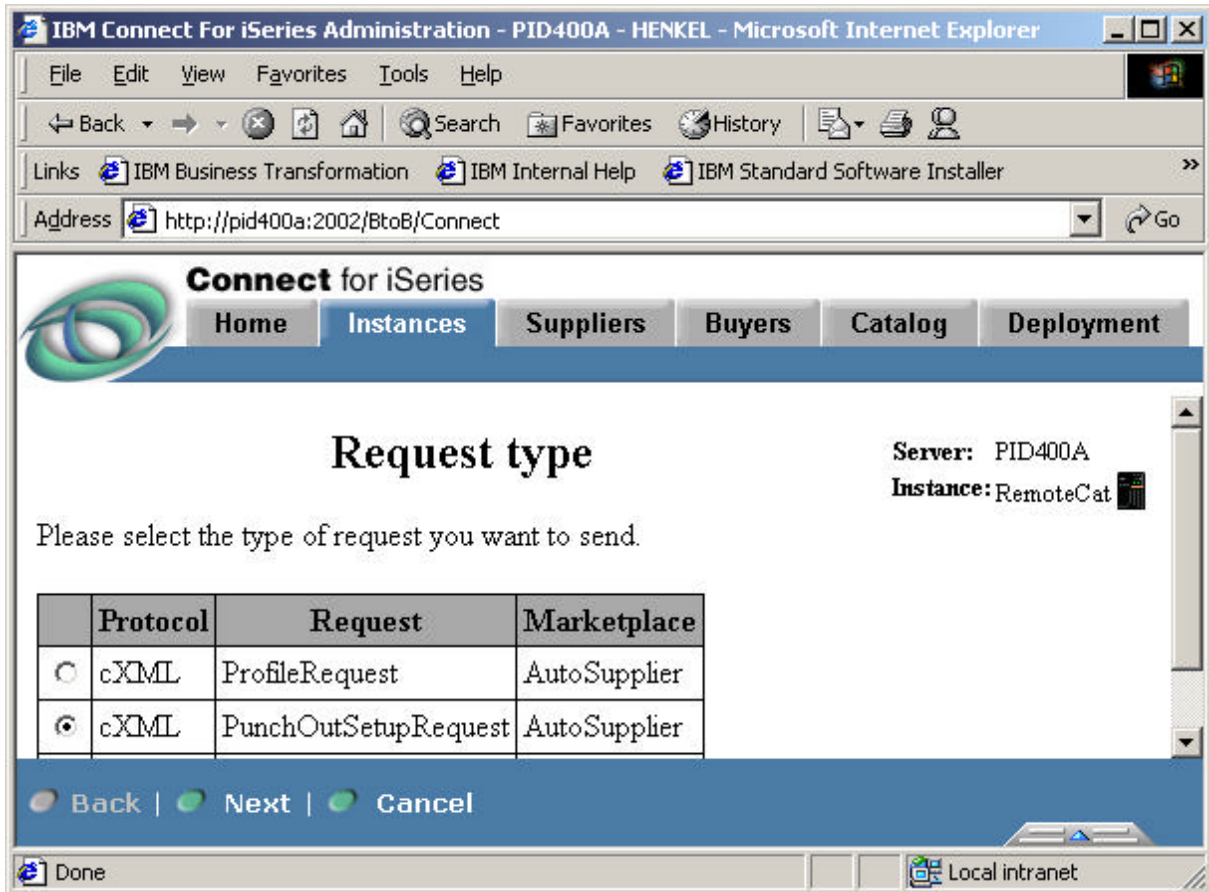
Response from iSeries Connect

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.1.009/cXML.dtd"
<cXML payloadID="9974615017381@PID400A.RCHLAND.IBM.COM"
  timestamp="2001-08-10T16:38:21+00:00" version="1.1">
  <Response>
    <Status code="200" text="OK"/>
    <ProfileResponse effectiveDate="2001-08-08T14:03:32+00:00">
```

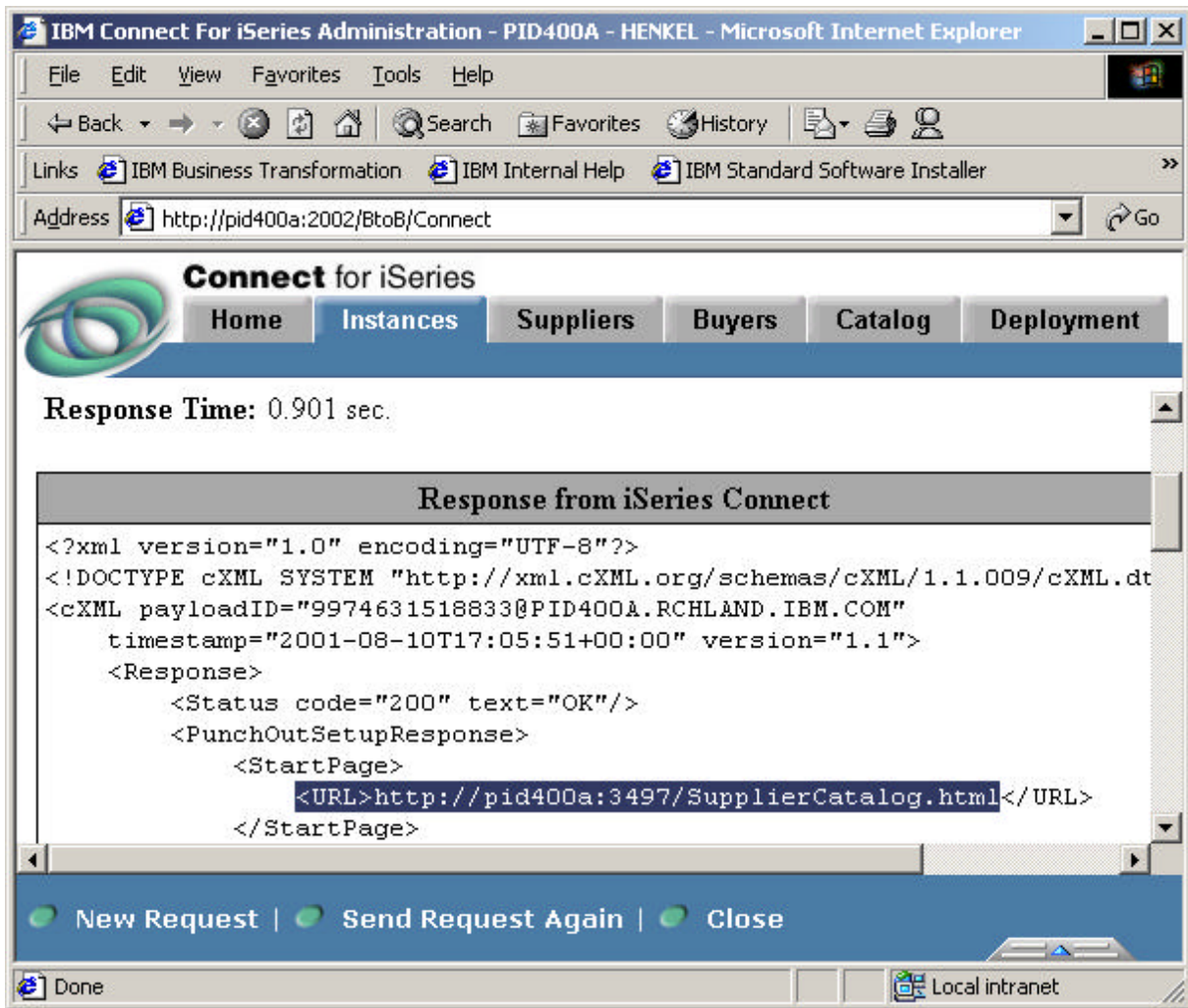
Notice the highlight line 200, OK. If a 200 message was not received, something is wrong. It may be something is not configured correctly, not started, or the shared secret was entered incorrectly. Determine the problem and resolve before moving on.

Stream PunchoutSetup Request Message

Once the ProfileRequest message is working, send a PunchoutSetup Request message to the instance using the Test Drive Connect tool. The steps here are very similar to sending the ProfileRequest except to select PunchoutSetupRequest.



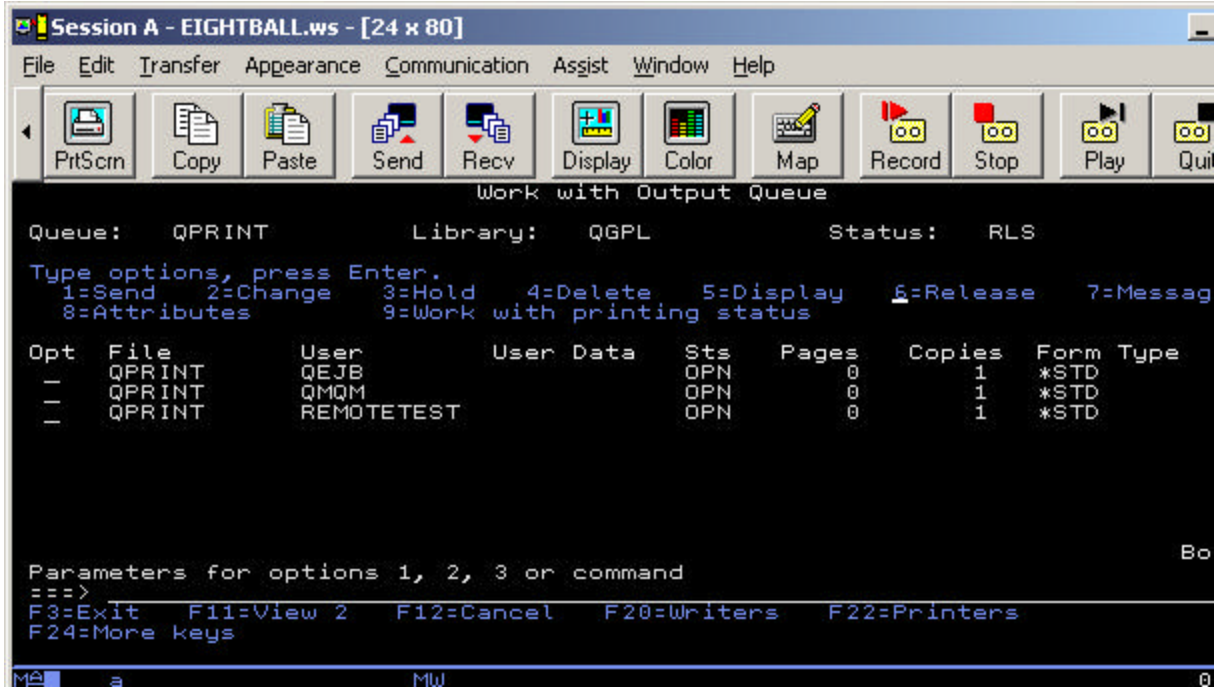
Again move through the wizard selecting the defaults and entering the correct Supplier Password then press the Send Request on the last screen.



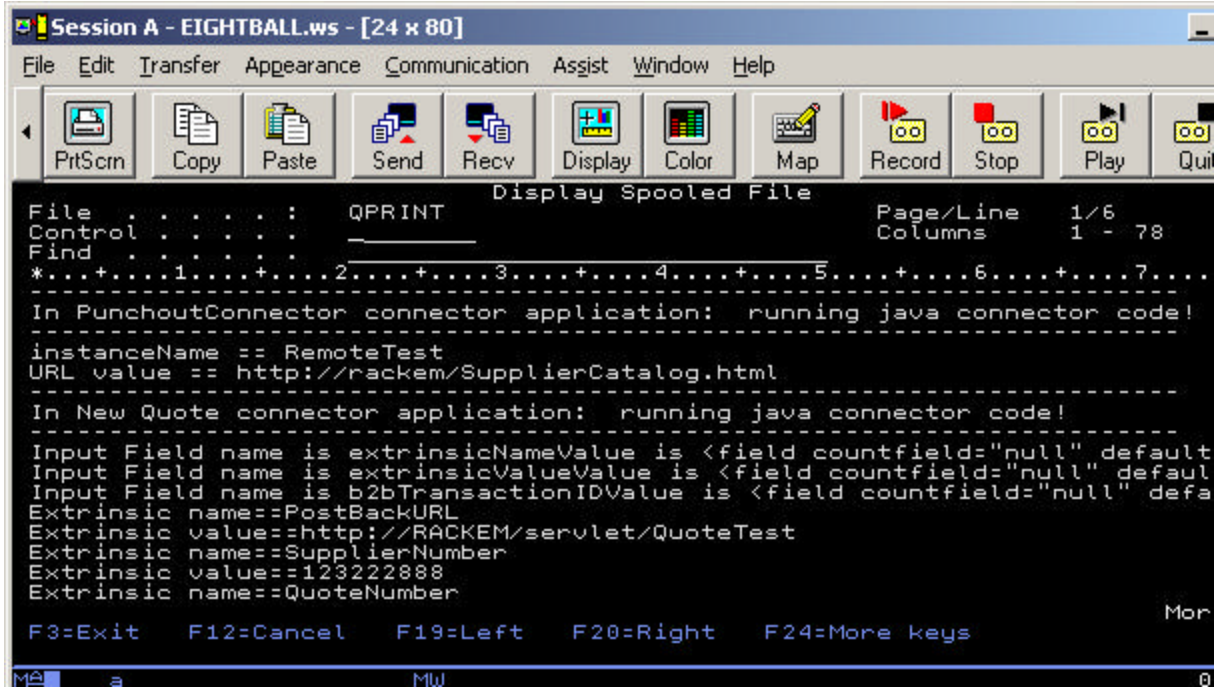
If successful, a screen similar to the one above will be displayed. Notice the highlighted URL. This is the URL stored internally by Connect for iSeries when the custom page was added to the buyer configuration.

What just occurred is the PunchoutSetup Request message was received by Connect and the PunchoutConnector.class connector was called. The PunchOutConnector class accesses the Connect for iSeries tables looking for the URL Redirect field for buyer Jane Buyer. These tables are accessed using API's provided by Connect for iSeries. The PunchOutConnector formats a response containing the URL data.

Any System.out.println() statements in the PunchOutConnector class will be redirected an output queue in QPRINT with the user id matching the name of the B2B Instance. Stop the B2B Instance to view this file.

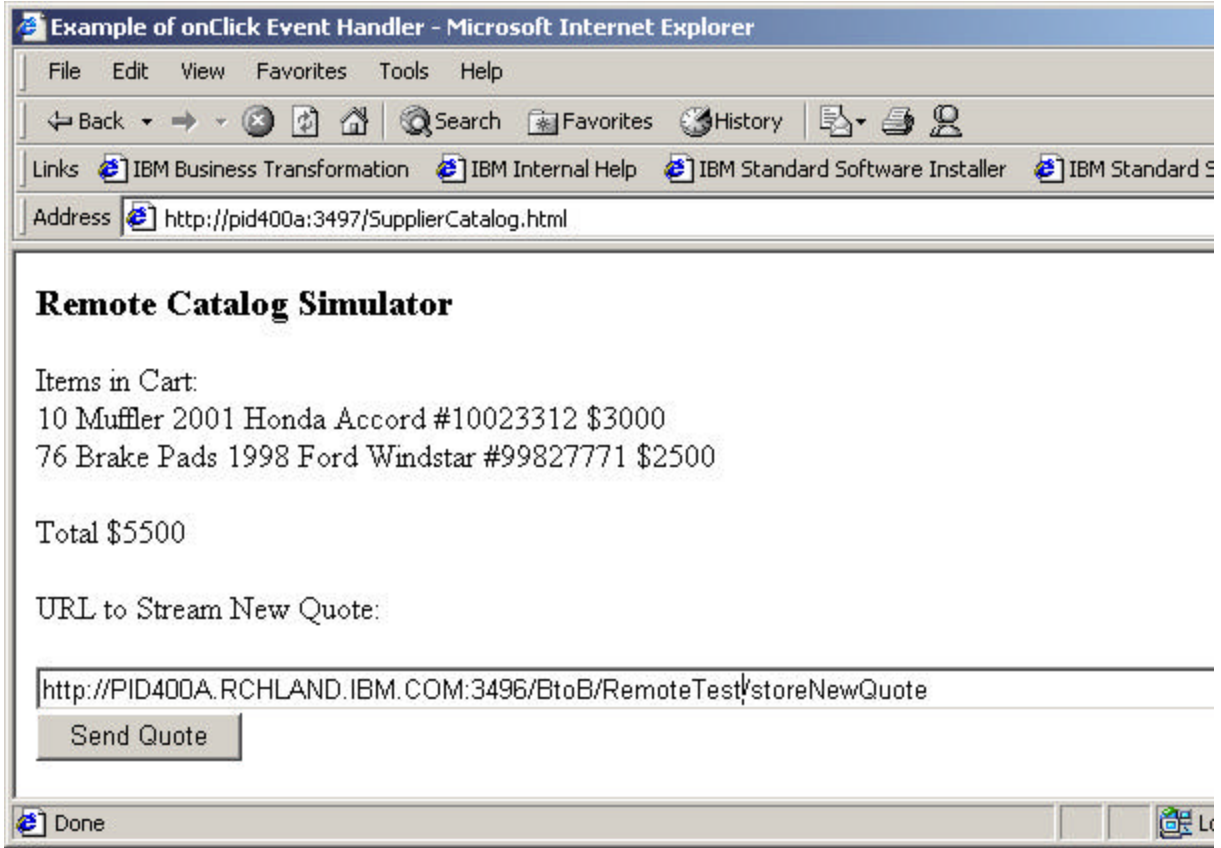


Viewing the Queue should reveal outputs from the PunchoutConnector.java and NewQuoteConnector.java file.



Display the SupplierCatalog.html file in a browser

As we indicated earlier, the simulated eProcurement software provided will open up a new browser window and display the SupplierCatalog.html file. It is important that a properly configured HTTP and WAS instance can serve up this file, as well as the QuoteTest servlet. Bring up a browser window and type the URL where the SupplierCatalog.html has been installed and configured. If everything worked the following will be displayed:



If the page cannot be served, determine why and fix the problem.

Run the Simulation

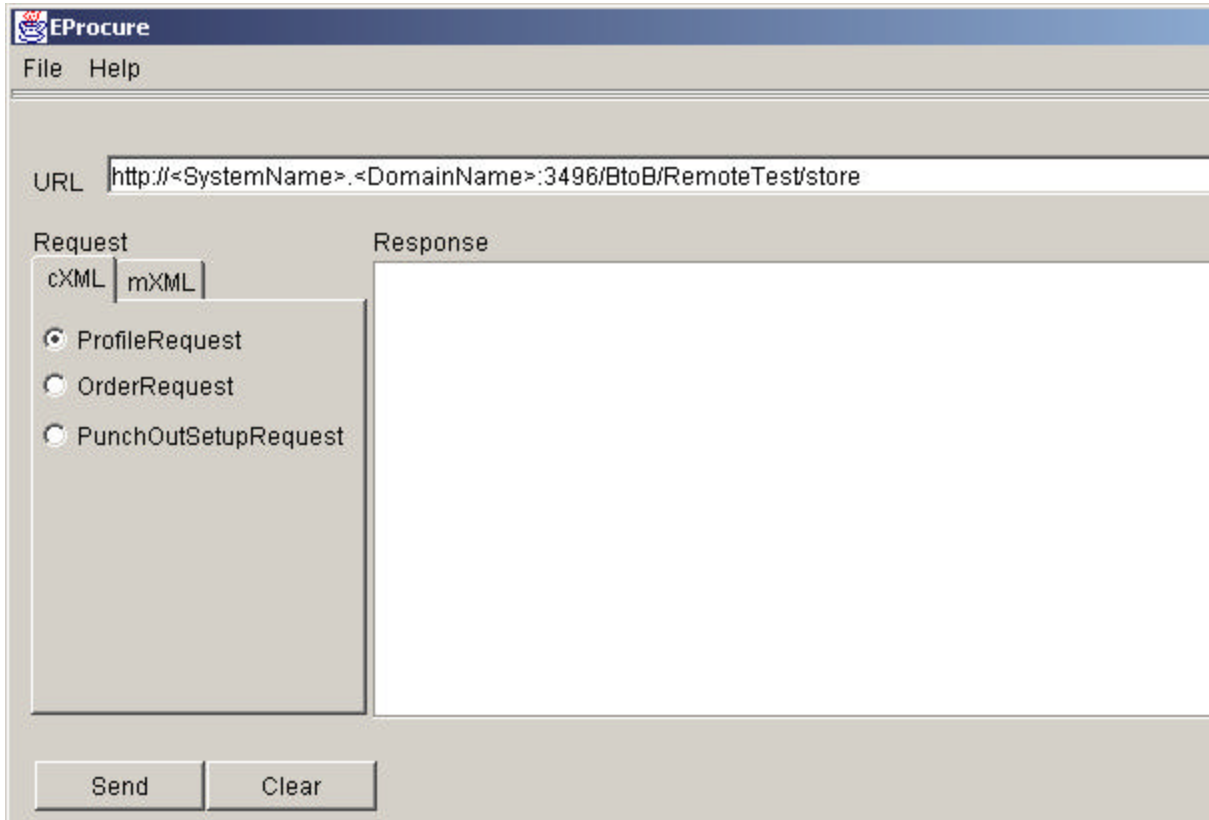
The final step in this example is to run the complete setup using eProcurement simulator. This demonstrates the whole process from buyer sending a PunchoutSetup Request, the eProcurement software redirecting to the Supplier Catalog, the NewQuote sent to Connect, and the PunchoutOrderMessage being sent in our case to a browser. The simulator is shipped with a properties file sample.props. It defaults the browser to C:\\ProgramFiles\\Internet Explorer\\IEXPLORE. If IE is stored in a different location modify this file to point where the browser is installed.

Client.zip file contains the following files:

Eprocure.bat	- Runs the sample program
Eprocure.jar	- Simulator
OrderRequest.xml	- OrderRequest sent to Connect
ProfileRequest.xml	- ProfileRequest message sent to Connect.
PunchoutSetup.xml	- PunchoutSetup Request message sent to Connect.
Sample.props	- Properties file used by Simulator
Xerces.jar	- Contains XML Parser Code
Xml4j.jar	- Contains XML Parser Code.

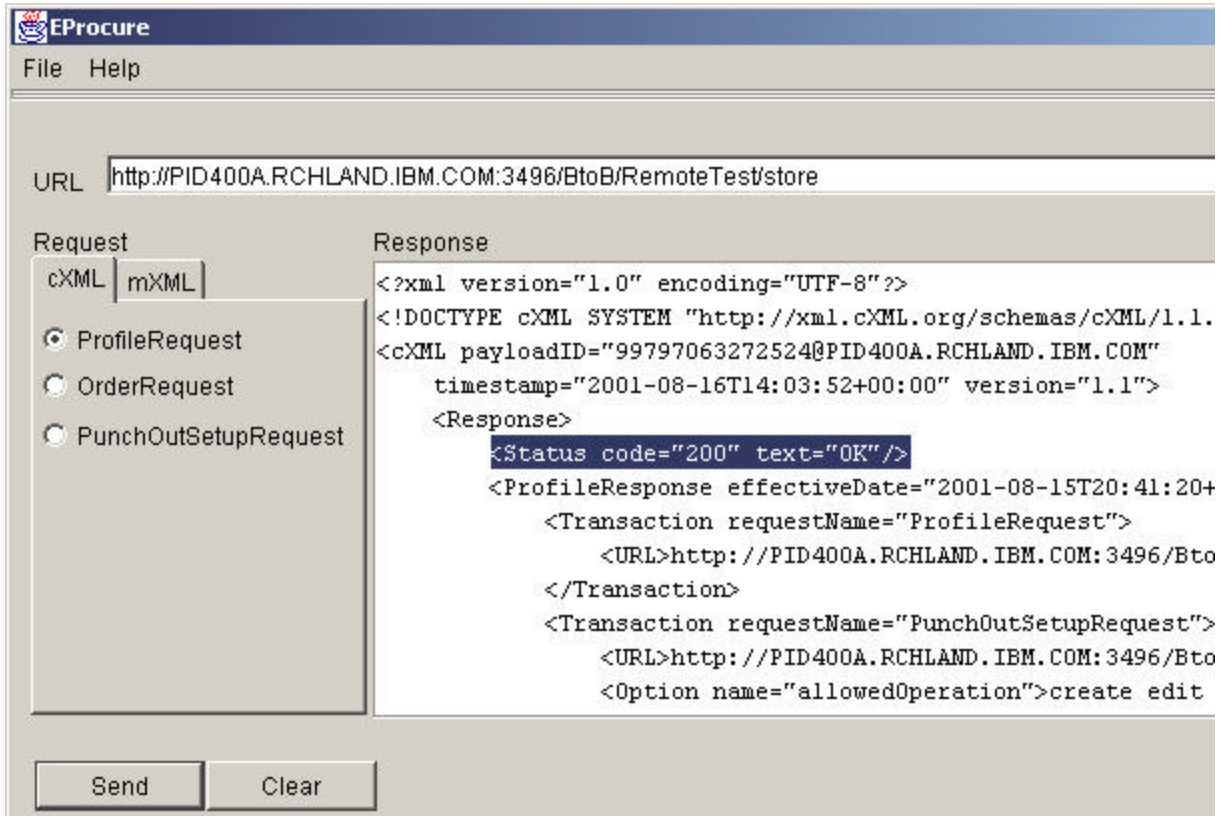
It is assumed at least JDK 1.8 is installed on the PC, and JAVA_HOME is set correctly.

Bring up a DOS window and change directories to the eprocure directory. Type eprocure at the prompt.



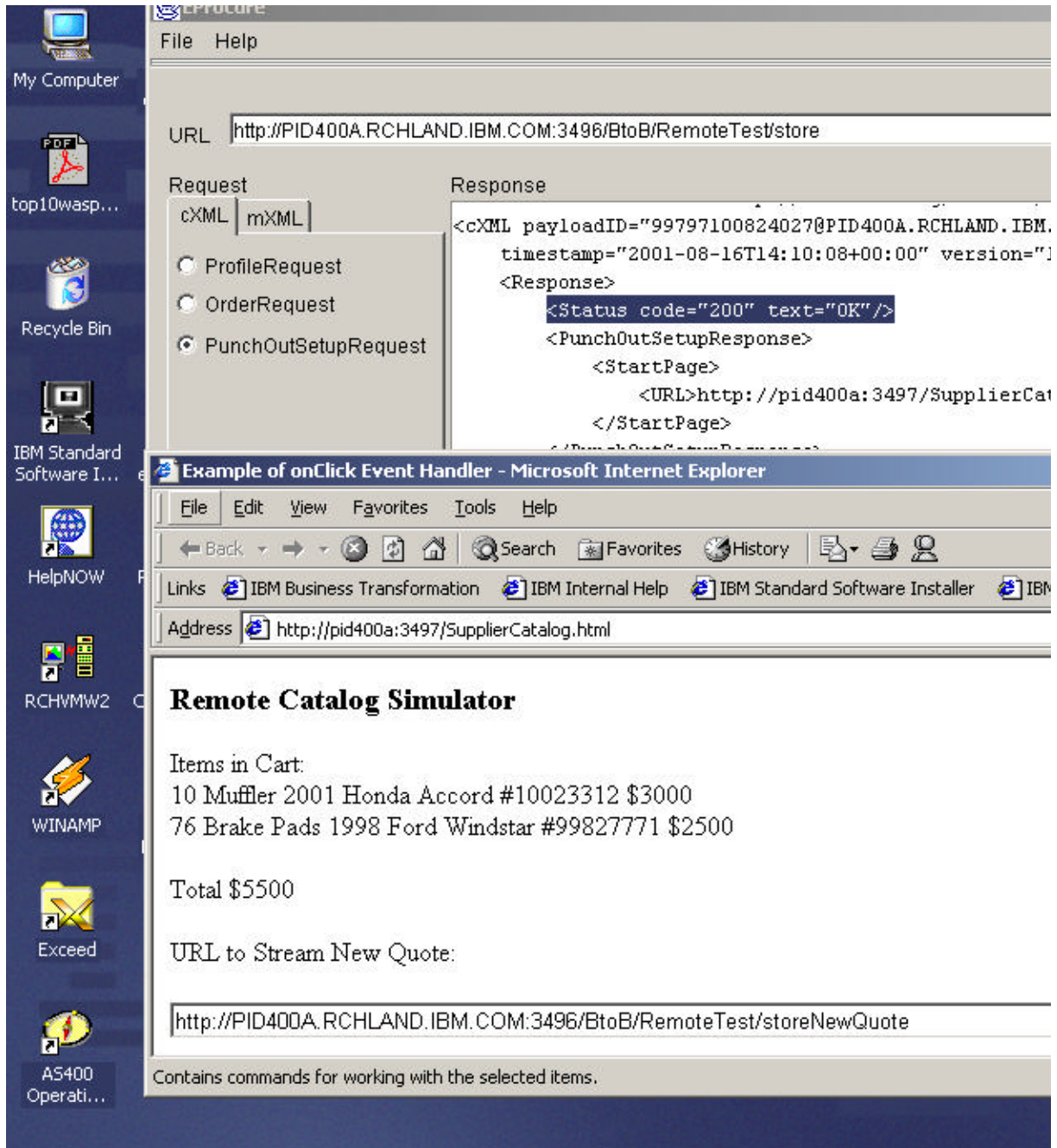
This will display the following window. Change the <SystemName>.<DomainName>:3496 to match the system name, domain name, B2B port number, and URL to match the configured B2B Instance.

To make sure things are working click the ProfileRequest radio button. Then Click the Send button. The ProfileResponse message should displayed in the response window.



If successful click the PunchOutSetupRequest radio button, then click the Send Button.

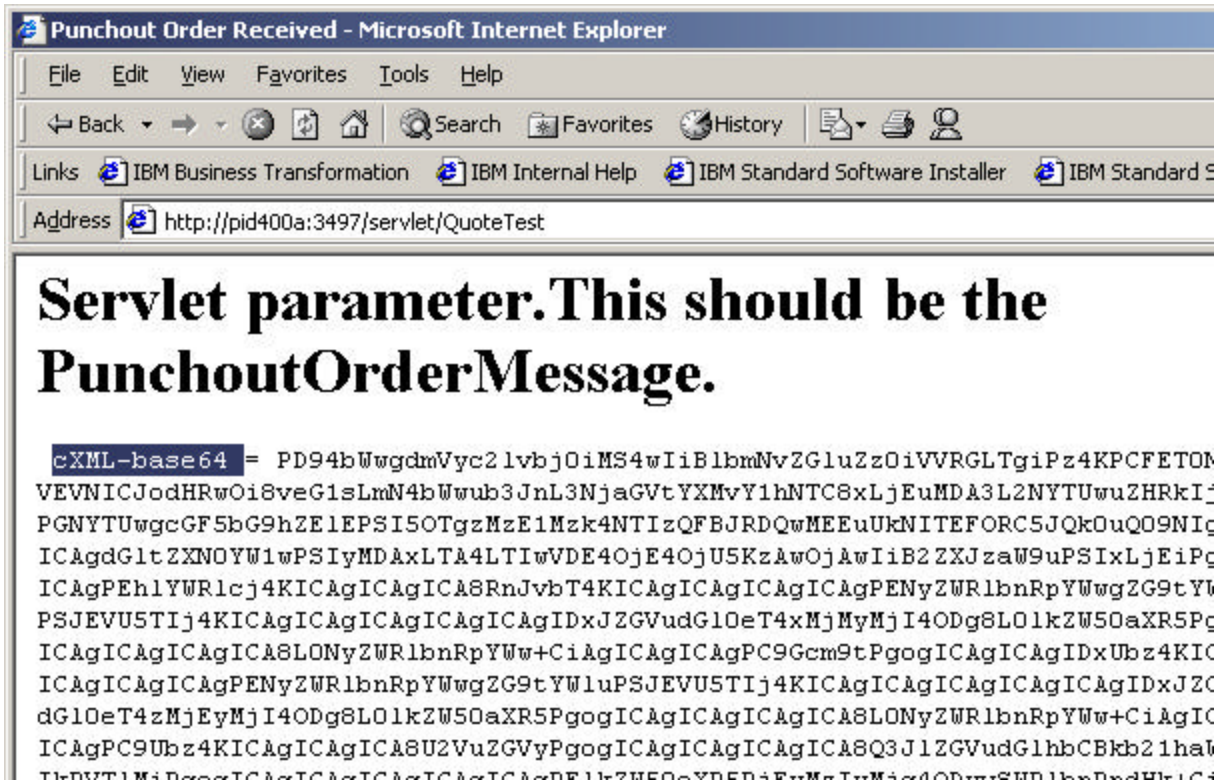
The PunchOutSetupResponse message will be displayed in the response window and a new browser window should be displayed.



The new browser window displayed simulates a purchase of items and should be the final screen displayed after the shopping experience. Pressing the SendQuote button will stream a NewQuote to the URL in the text box. Change this URL to match the instances storeNewQuote URL.

Press the SendQuote button to stream a NewQuote message to Connect for iSeries.

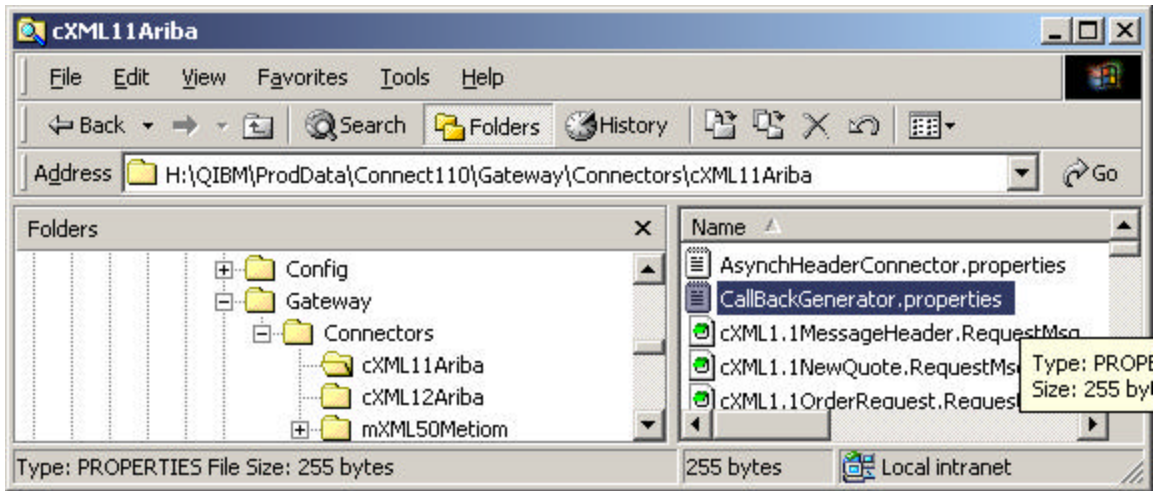
This will call the NewQuote Connector and stream a PunchoutOrder Message to the url specified as the PostBackURL. In our example it is the QuoteTest Servlet. Scroll down about one half of the page and there will be a message similar to the following..



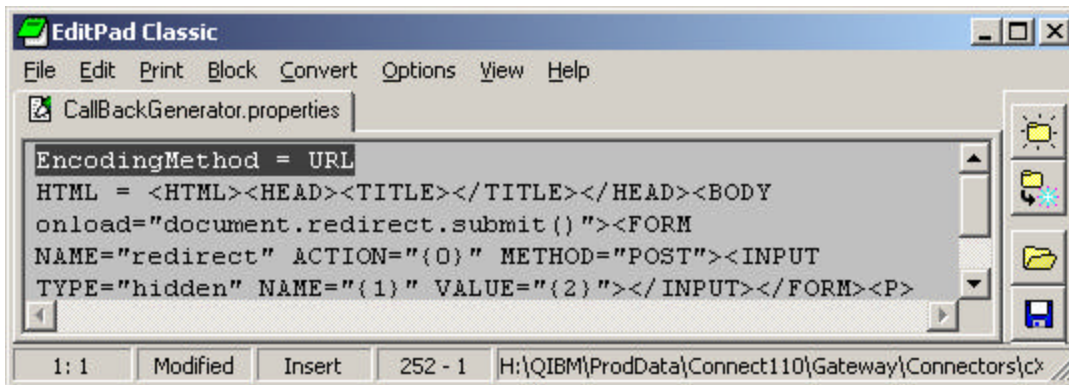
This is the payload that the eProcurement software would read to update the buy side tables. Notice that it is encoded in cXML-base64. To see this displayed in a browser window human readable, a simple modification will be required.

In the CallbackGenerator.properties file Connect defaults to cXML-base64 for both cXML 1.1 and cXML 1.2. This can be changed by editing the CallbackGenerator.properties file for cXML 1.1 or CA12_CallbackGenerator.properties and change the encoding method from Base64 to URL.

These files are located in \QIBM\ProdData\Connect110\Gateway\connectors\cXML11Ariba if the instance is configured for cXML 1.1

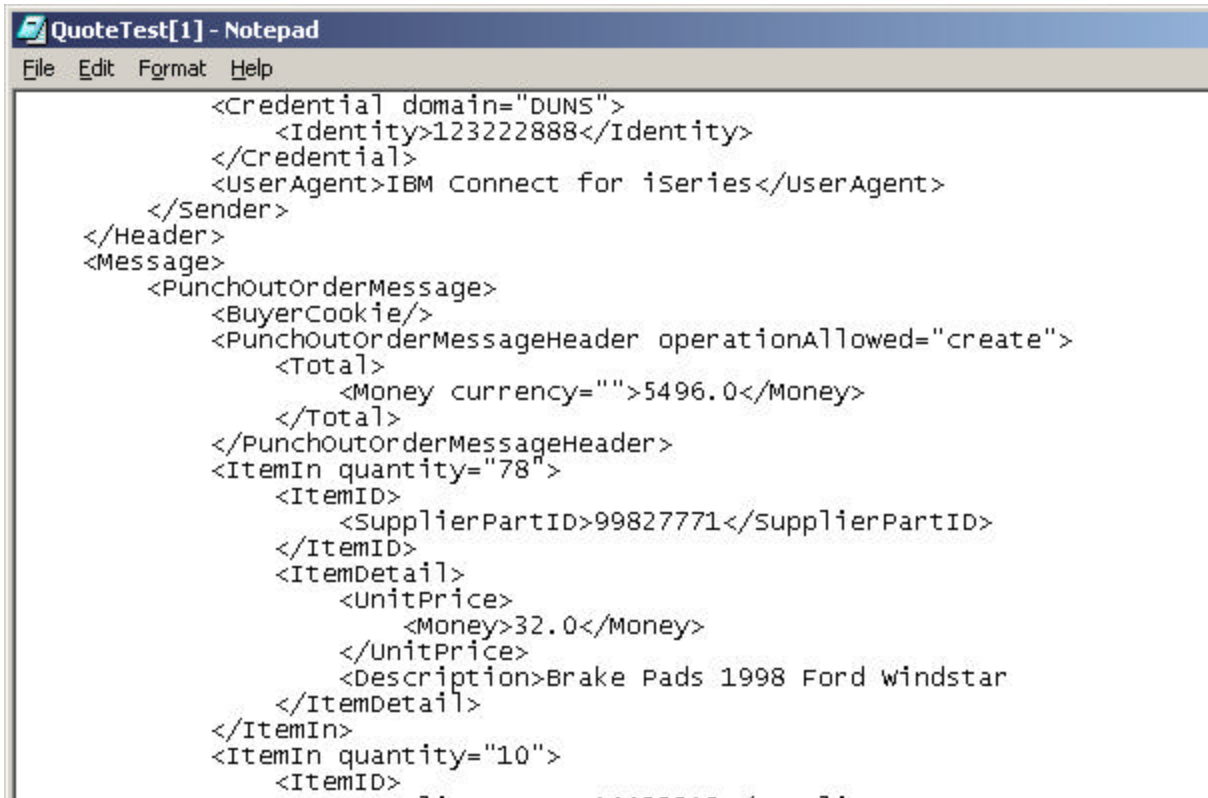


Change the EncodingMethod = Base64 to
 EncodingMethod = URL



Run the PunchoutSetup example again and stream the NewQuote message to the instance. This time the PunchoutOrderRequest Message will be human readable containing the products ordered, prices, and a total..

View the source from a browser to see the xml that would normally get sent to the browser.



```
QuoteTest[1] - Notepad
File Edit Format Help
<Credential domain="DUNS">
  <Identity>123222888</Identity>
</Credential>
<UserAgent>IBM Connect for iSeries</UserAgent>
</Sender>
</Header>
<Message>
  <PunchOutOrderMessage>
    <BuyerCookie/>
    <PunchOutOrderMessageHeader operationAllowed="create">
      <Total>
        <Money currency="">5496.0</Money>
      </Total>
    </PunchOutOrderMessageHeader>
    <ItemIn quantity="78">
      <ItemID>
        <SupplierPartID>99827771</SupplierPartID>
      </ItemID>
      <ItemDetail>
        <UnitPrice>
          <Money>32.0</Money>
        </UnitPrice>
        <Description>Brake Pads 1998 Ford windstar
      </ItemDetail>
    </ItemIn>
    <ItemIn quantity="10">
      <ItemID> .. .. . . . . . . . . . .
```

As indicated earlier, the response is “well formed” but not “valid”. The reason is that this example does not respond with a Currency and Description language field. Turning back on Gateway validation and rerunning this example will cause an error to be displayed in the browser.

Conclusion

The fundamentals of implementing this functionality are not difficult but skills in many areas will be necessary. Studying the example connectors and HTML page should give the reader enough information to formulate how this functionality could be implemented in a real implementation.