Solutions for Access and Integration to HP OpenVMS Applications

Jim Jennis, Integration Architect, WRQ

Manoj Winfred, Senior Consultant, WRQ

WRQ and HP

Pioneers, Partners & Leaders in World Class Computing Solutions

- WRQ—Over 22 years of connecting host applications with emerging technologies
- Multiple options for extending the life of existing OpenVMS & HP applications
  - Host access/emulation
  - Web-enablement
  - Composite applications
  - Itanium® migration
- WRQ is an HP pioneer, and has been a trusted HP partner for over 20 years
- WRQ has more than 10,000 customers on HP platforms worldwide

WRQ Value to HP Customers

- Extend/Enhance use of HP platforms
  - OpenVMS
  - HP-UX
  - Linux
  - HP e3000
- Integrate with third party platforms
  - IBM 3720/5250, OS/390, AS/400
  - UNIX—AIX, Solaris, etc
- A single vendor for legacy extension and integration
  - Terminal emulation — fat & thin
  - Web-refacing of legacy applications
  - Creation of web-services and composite applications from legacy functions and data

SOA Enables Enterprise Legacy Applications

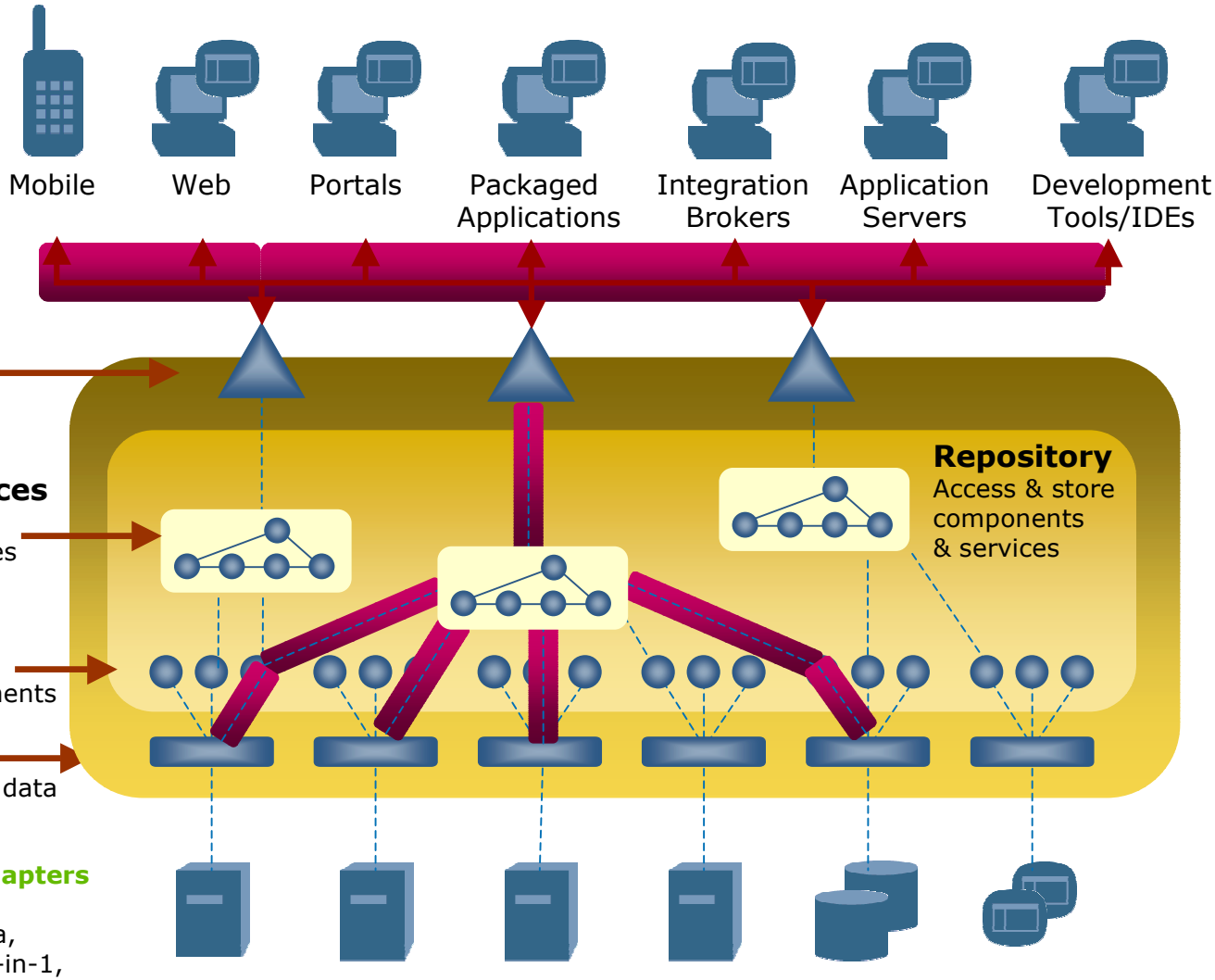
Target Applications & Tools
Reuse legacy functions in new ways

Interfaces
.NET, COM, Java, EJB, Web-services, HTTP/XML, JMS

Composite Services
Combine components into high value services

Components
Represent business functions or data elements

Adapters
Abstracts host logic & data into components



OpenVMS specific adapters
Screen: Any VT-based application (VAX, Alpha, Itanium) including: All-in-1, FMS, FMS/Datatrieve, Cognos Powerhouse, 3GLs
Data: RMS Files, RMS/Cobol, RMS/Datatrieve, Oracle/RDB, CDD, Oracle

HP WORLD 2004
Solutions and Technology Conference & Expo

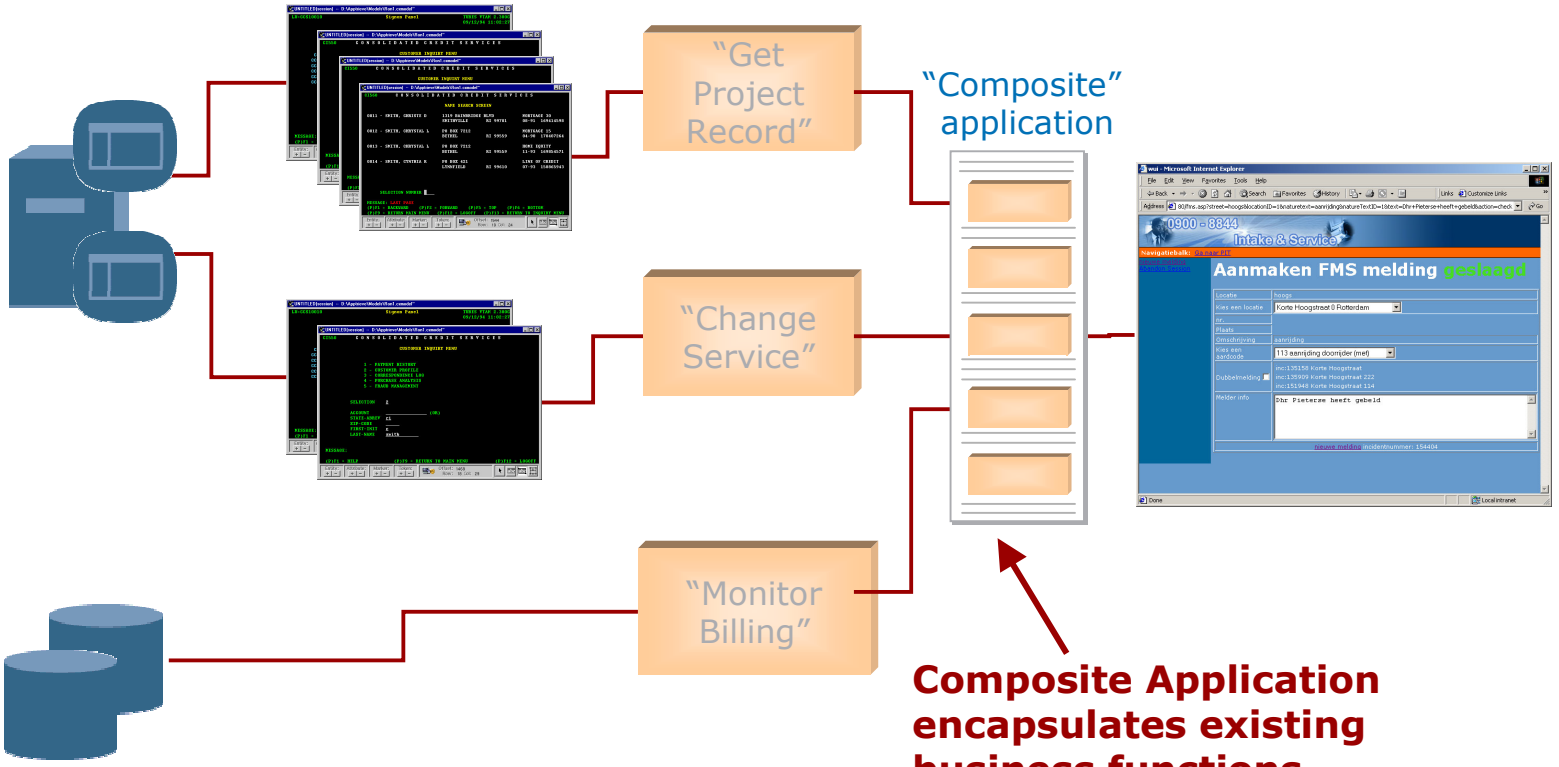
Composite Application Architecture

Existing host applications & databases

User steps to complete a business function

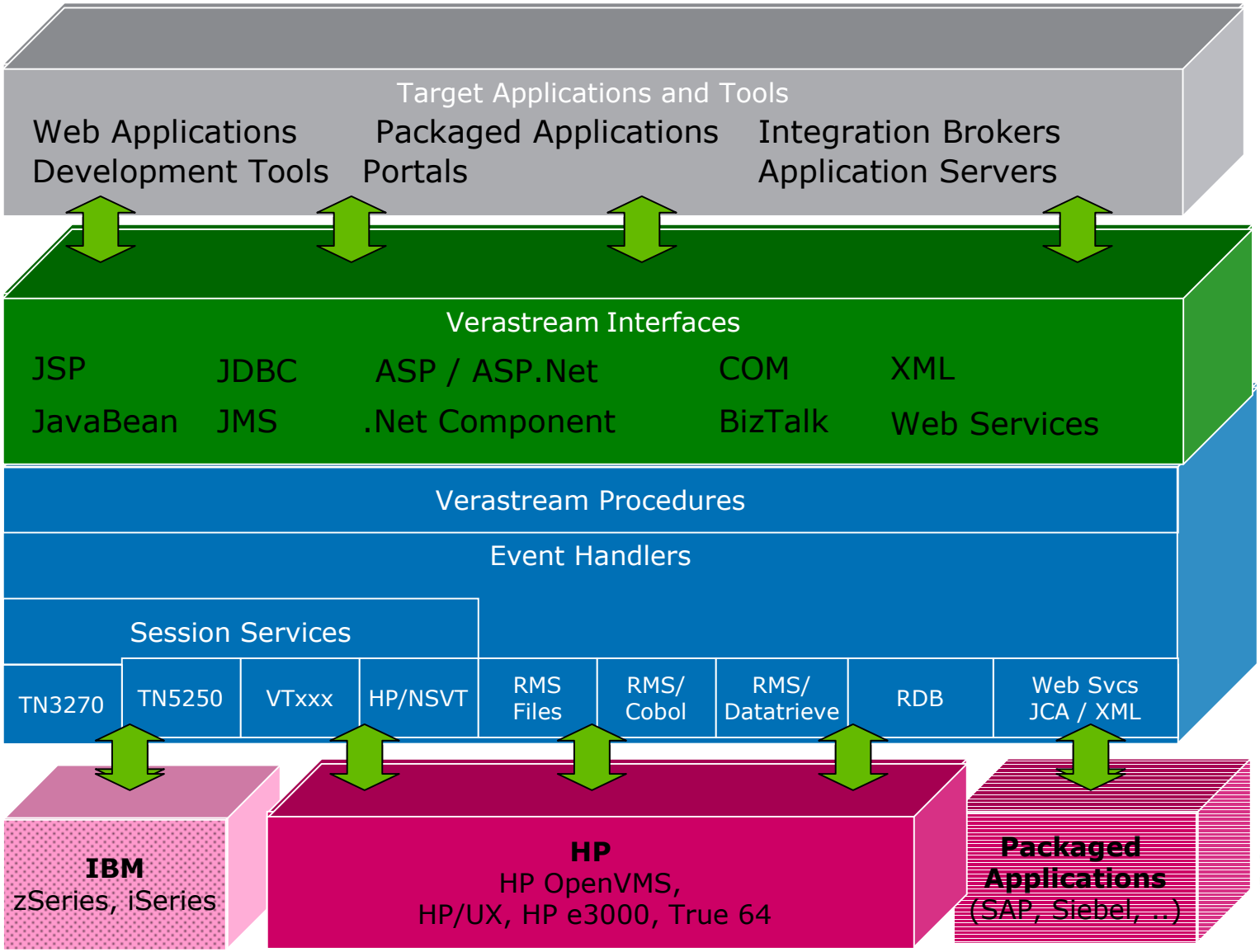
Reusable business components

New call center web interface

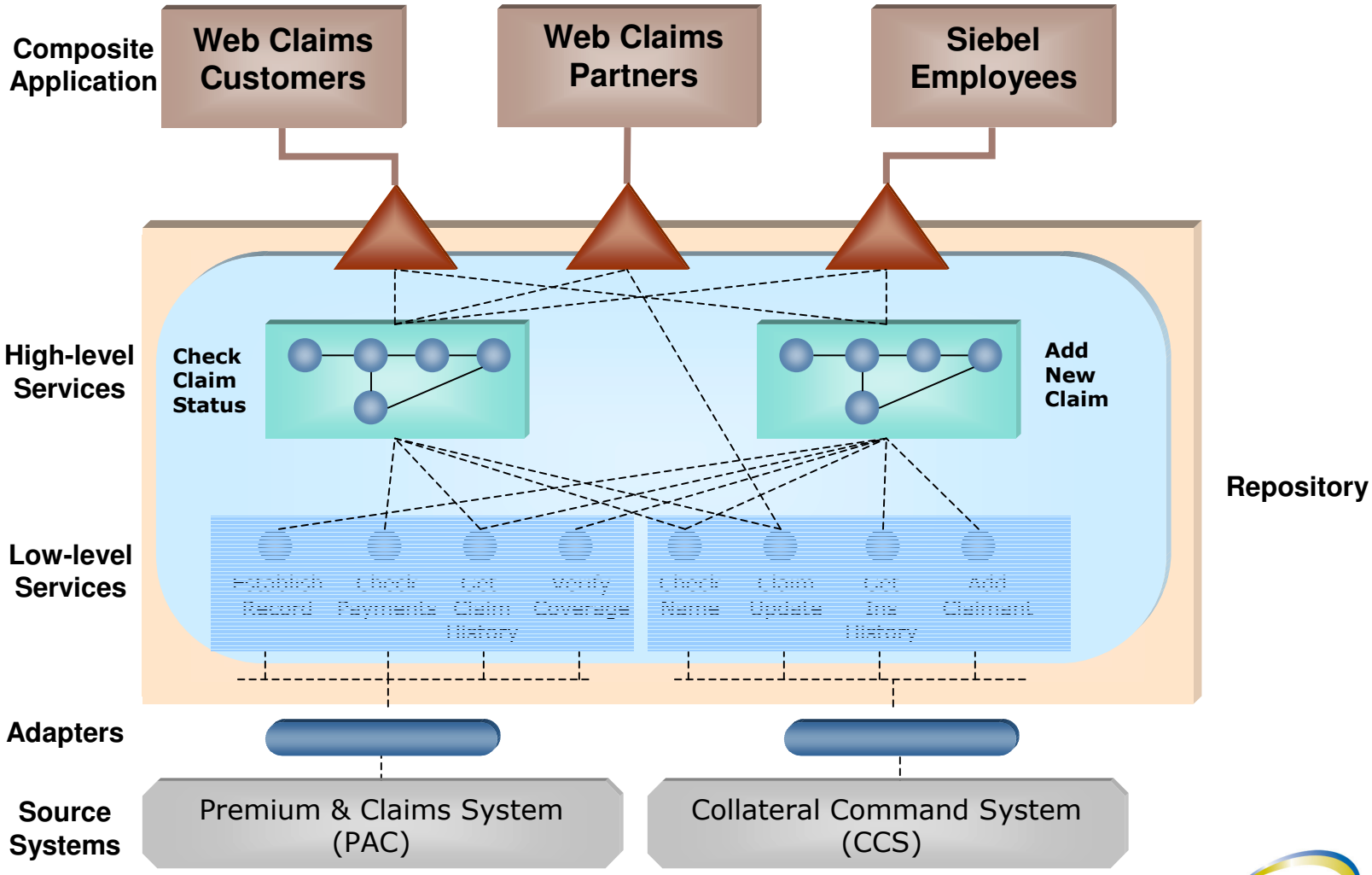


Composite Application encapsulates existing business functions allowing reuse and helping accelerate new application development

Verastream Architecture



Verastream Composite Applications – Customer Example



Verastream Composite Application Tutorial

Freight Claims Payment—Verastream Demo

Combines/reuses:

Data Layer 

- RMS Data used/maintained by existing OpenVMS Cobol Applications

U.I. 

- FMS/Fortran Character Based Checking Account Application wrapped with Verastream Host Integrator

API 

- A legacy OpenVMS application written in C and wrapped with HP Bridgeworks to create a Java callable interface

WEB 

- A modern easy-to-use WEB user interface developed using Java/JSP

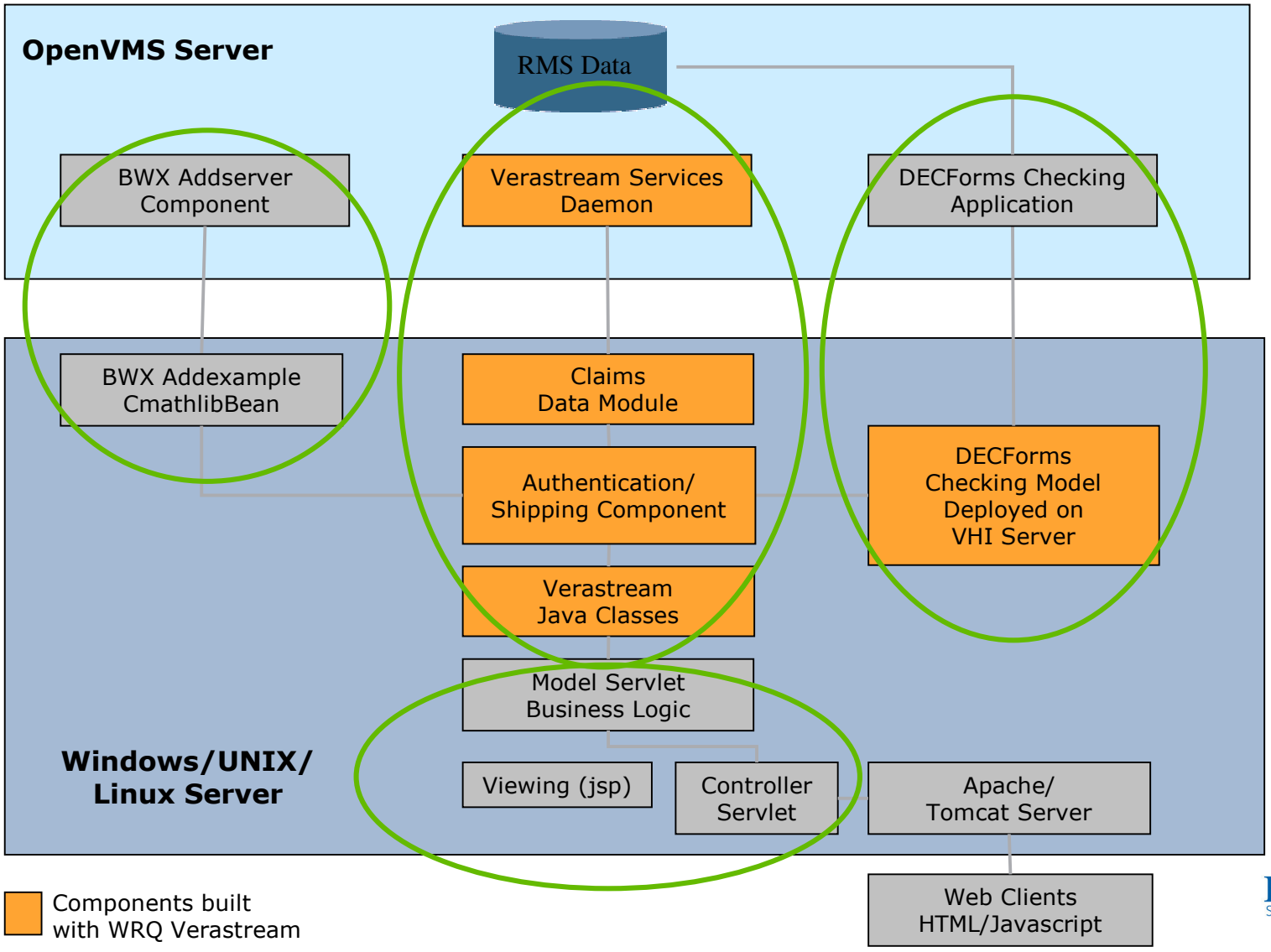
Verastream Composite Application Development Process

Freight Claims Payment Development Process

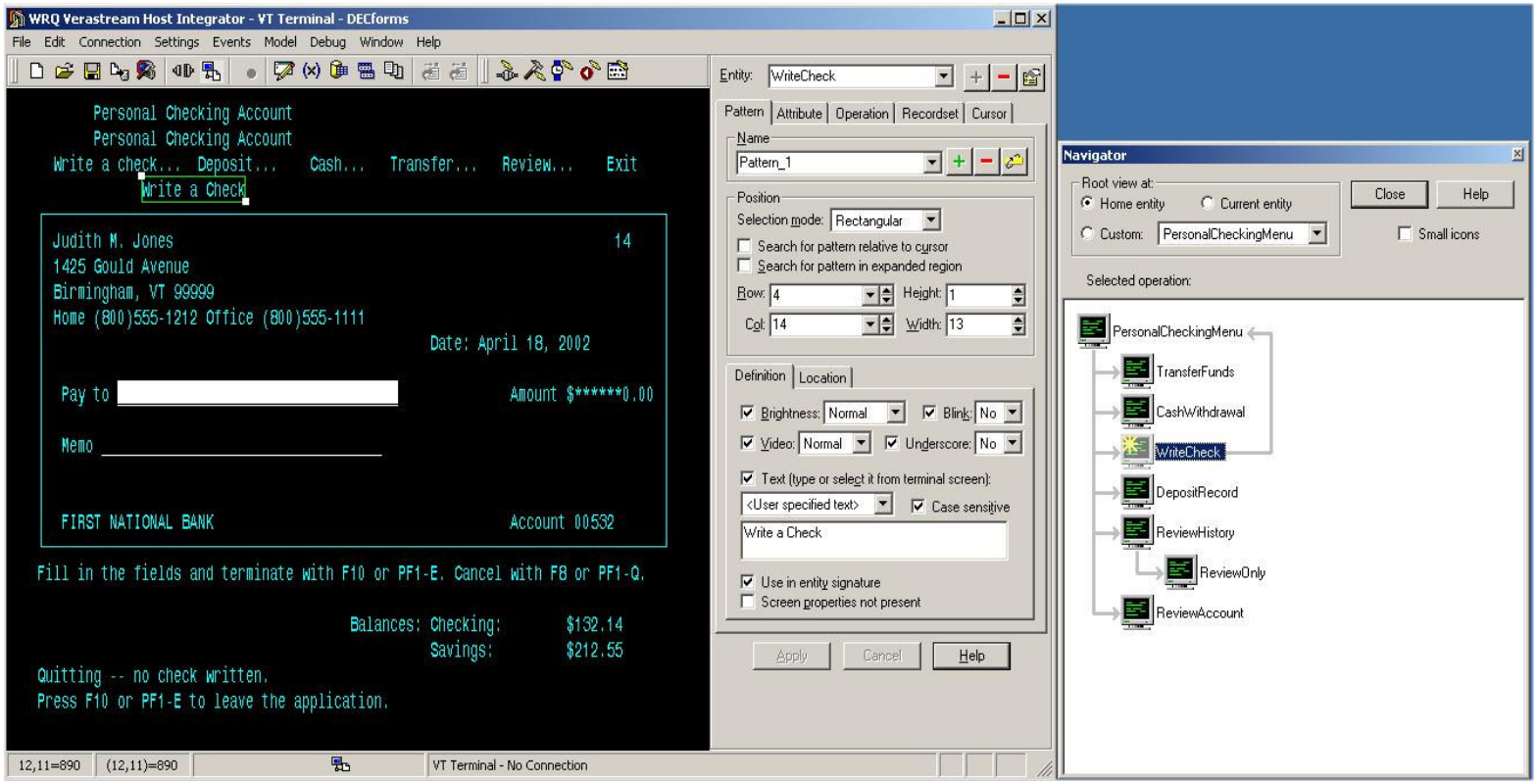
Five simple steps:

- Build the components
- Add the components to the Repository
- Assemble the composite application
- Deploy the composite application
- Test the composite application
- Mission Accomplished!

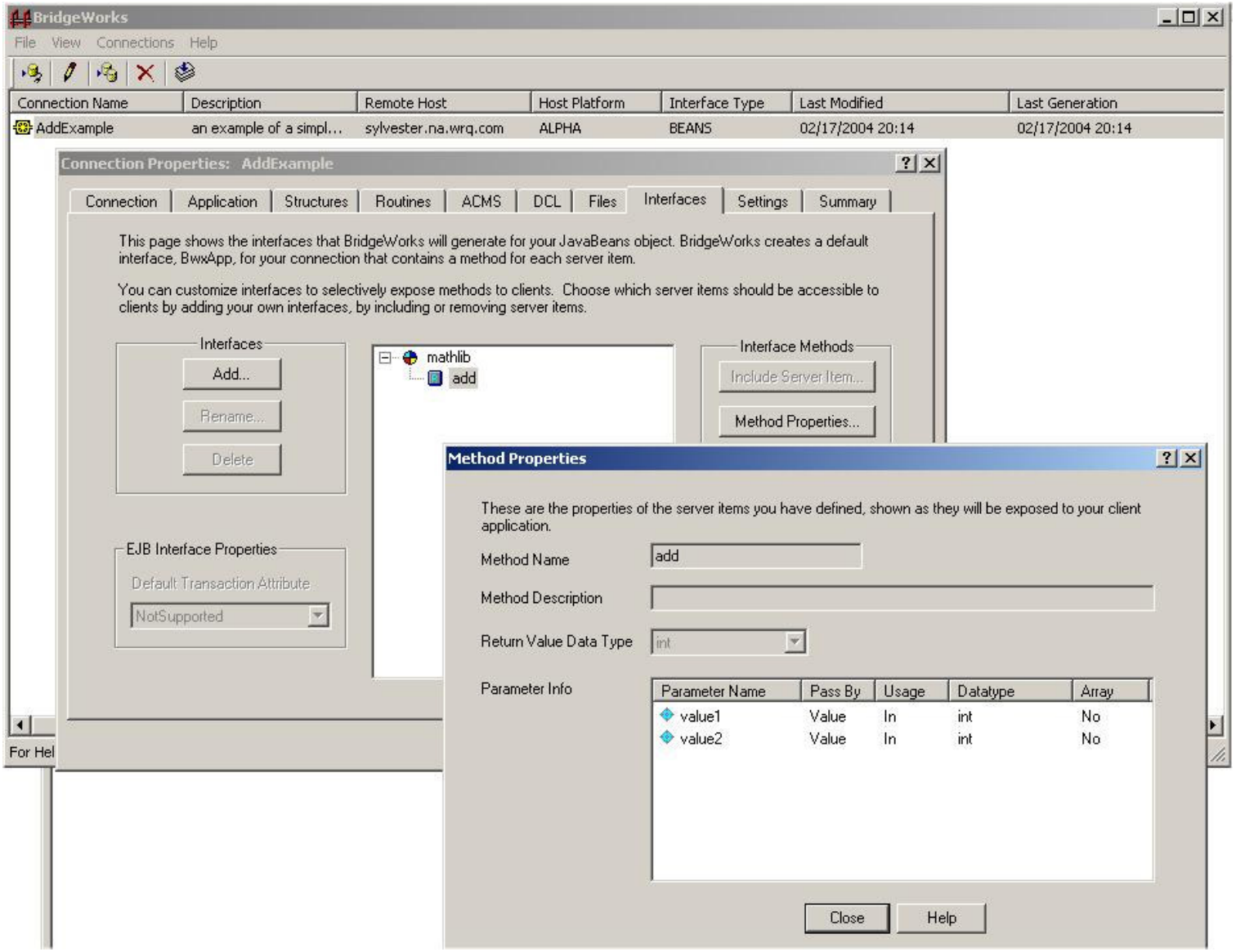
Freight Claims Composite Application Architecture Overview



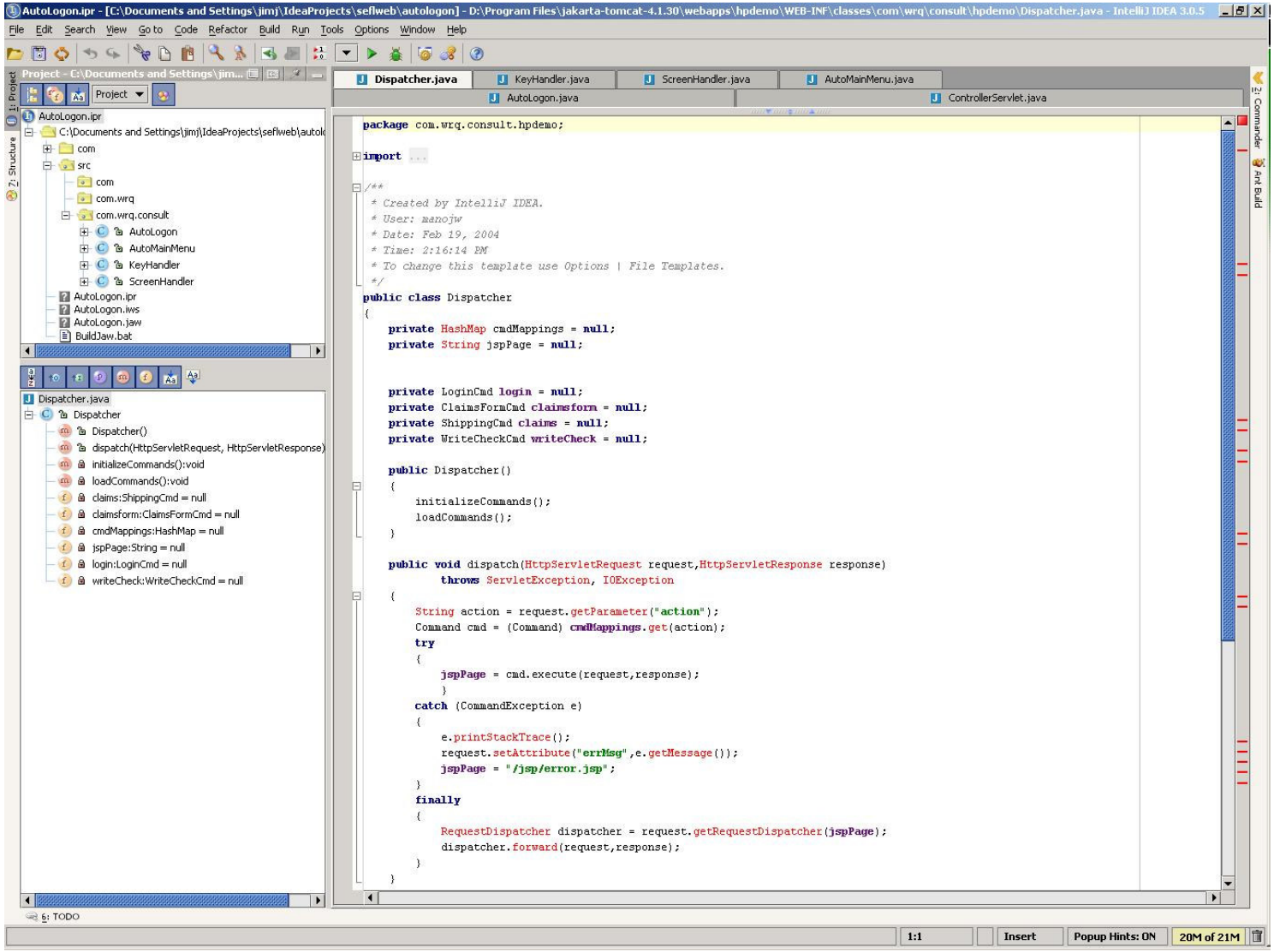
Building the DECforms Application Model



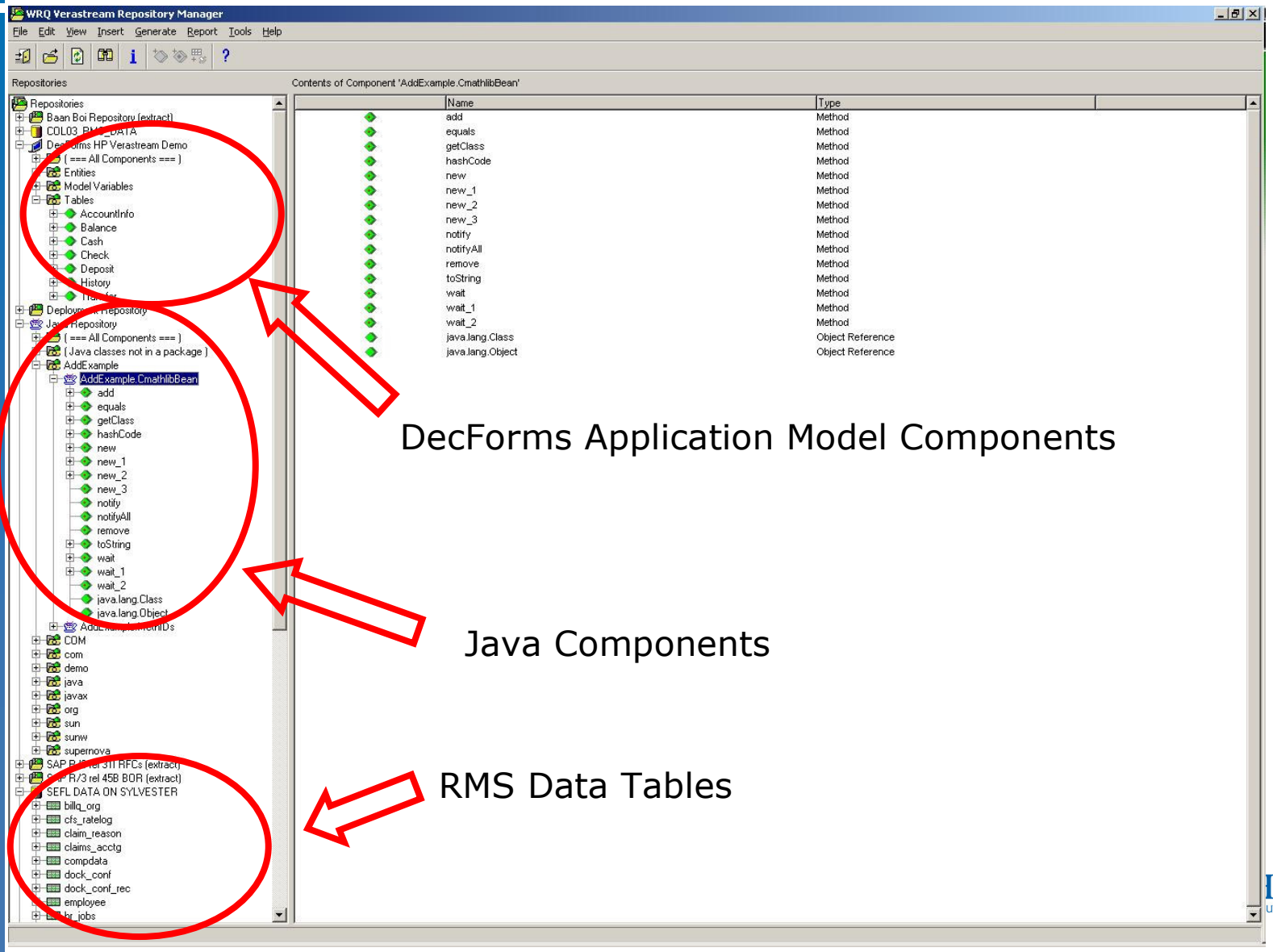
Building the Bridgeworks Component



Building the Java Components



Component Management
WRQ Verastream Repository

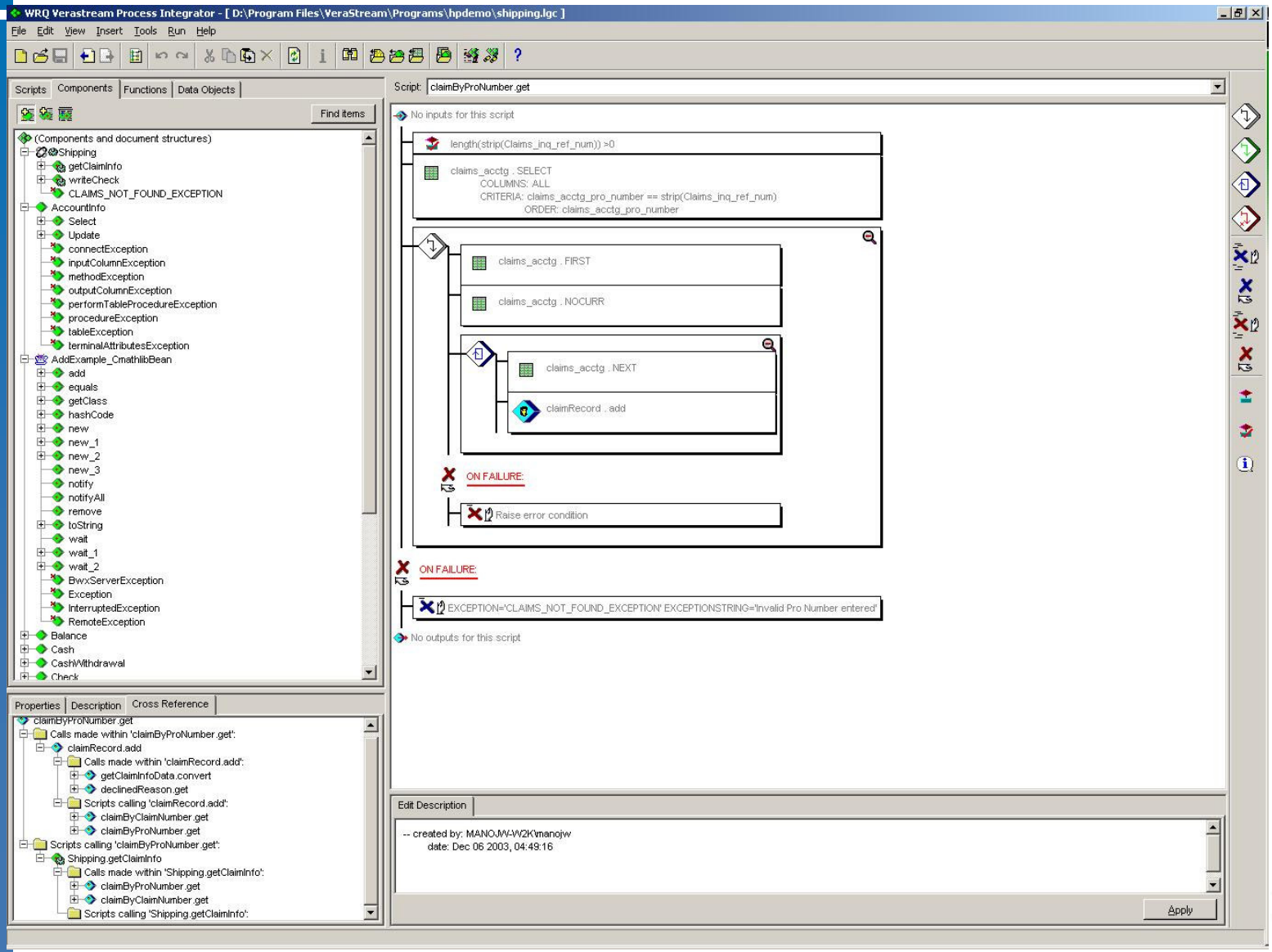


DecForms Application Model Components

Java Components

RMS Data Tables

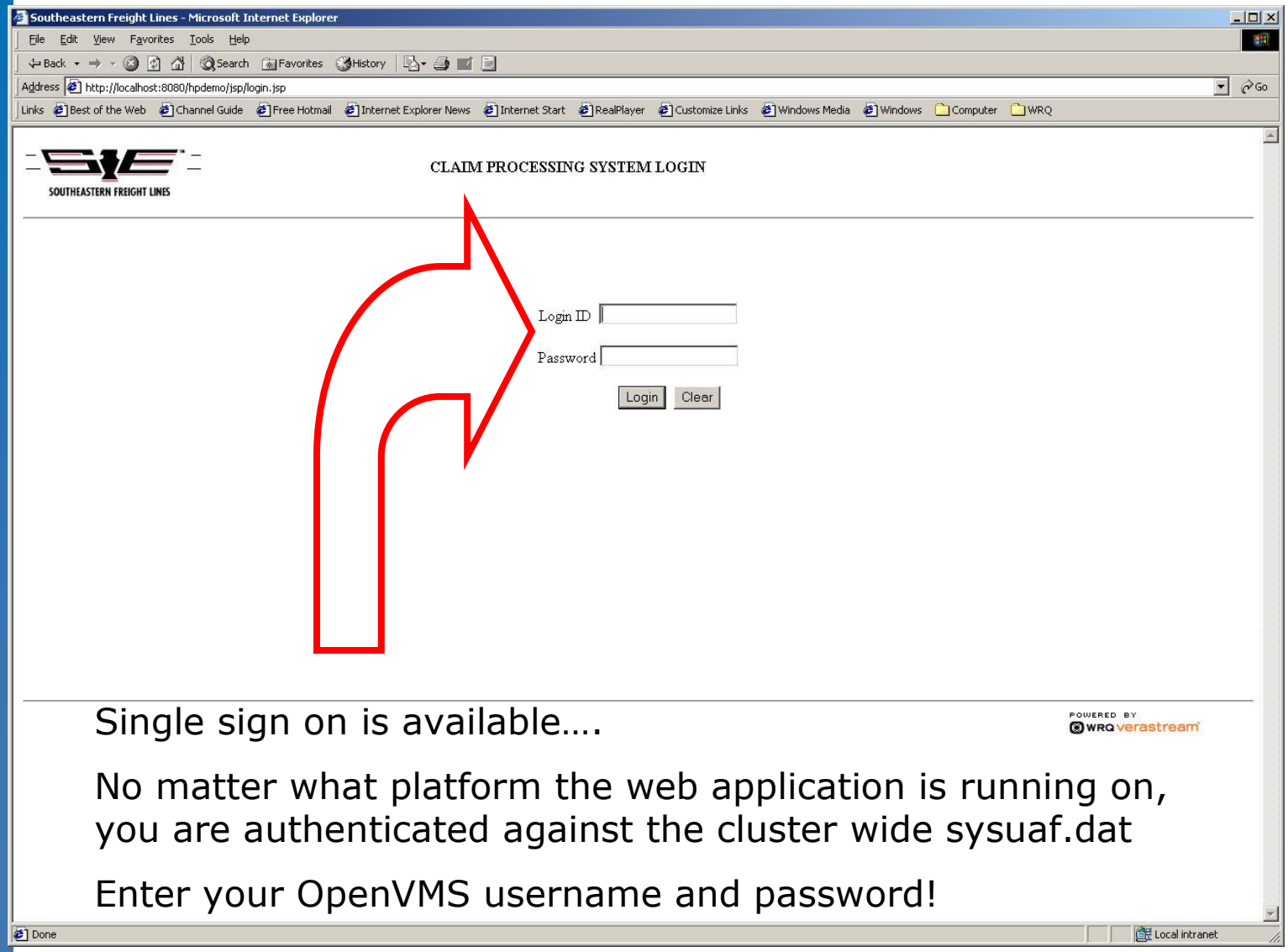
Assembling the Composite Application
Verastream Process Integrator



Ready, Set, Go!!
Let's Get Started!!

Here's what we will do….

OpenVMS Authentication Component

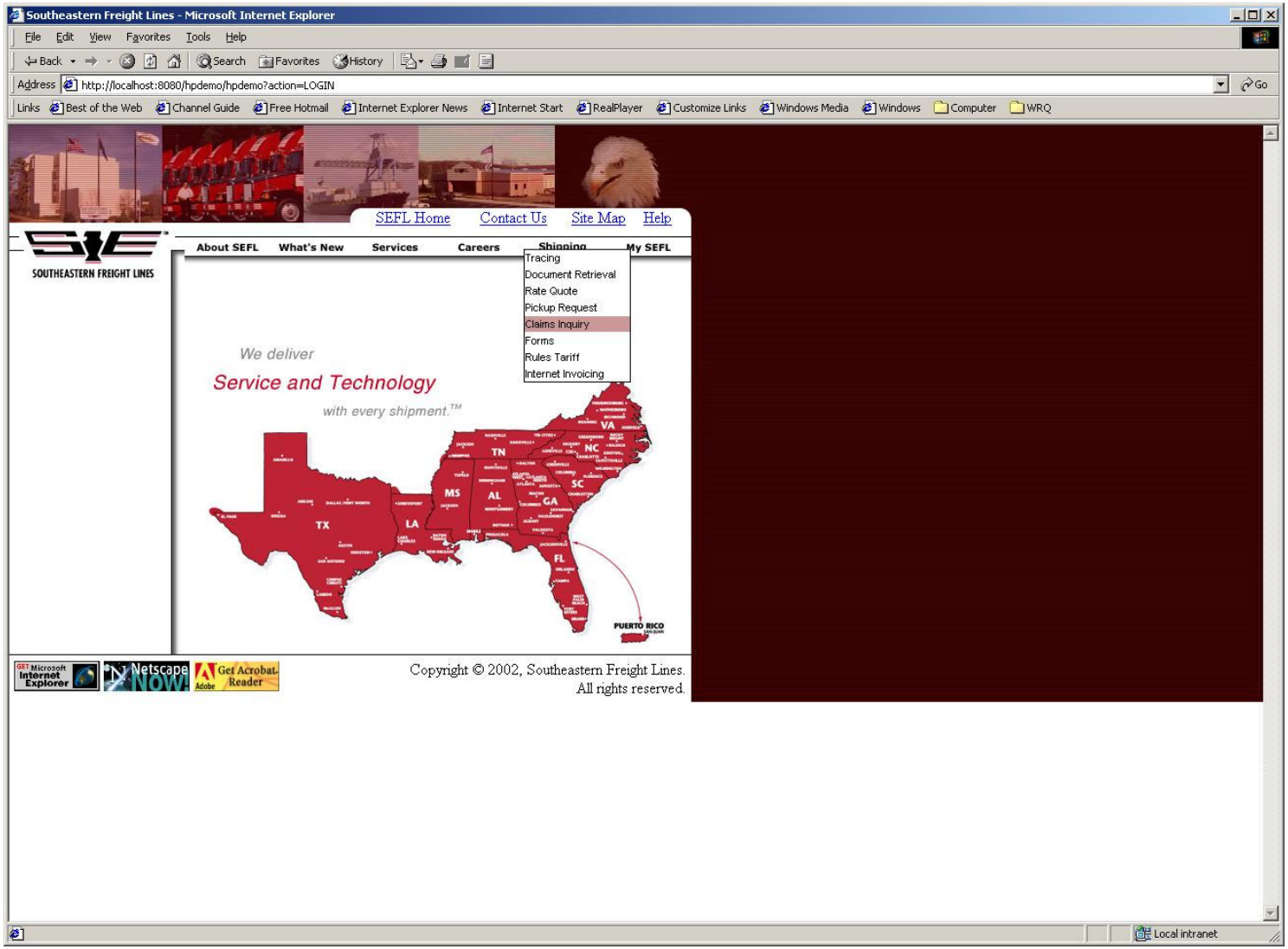


Single sign on is available….

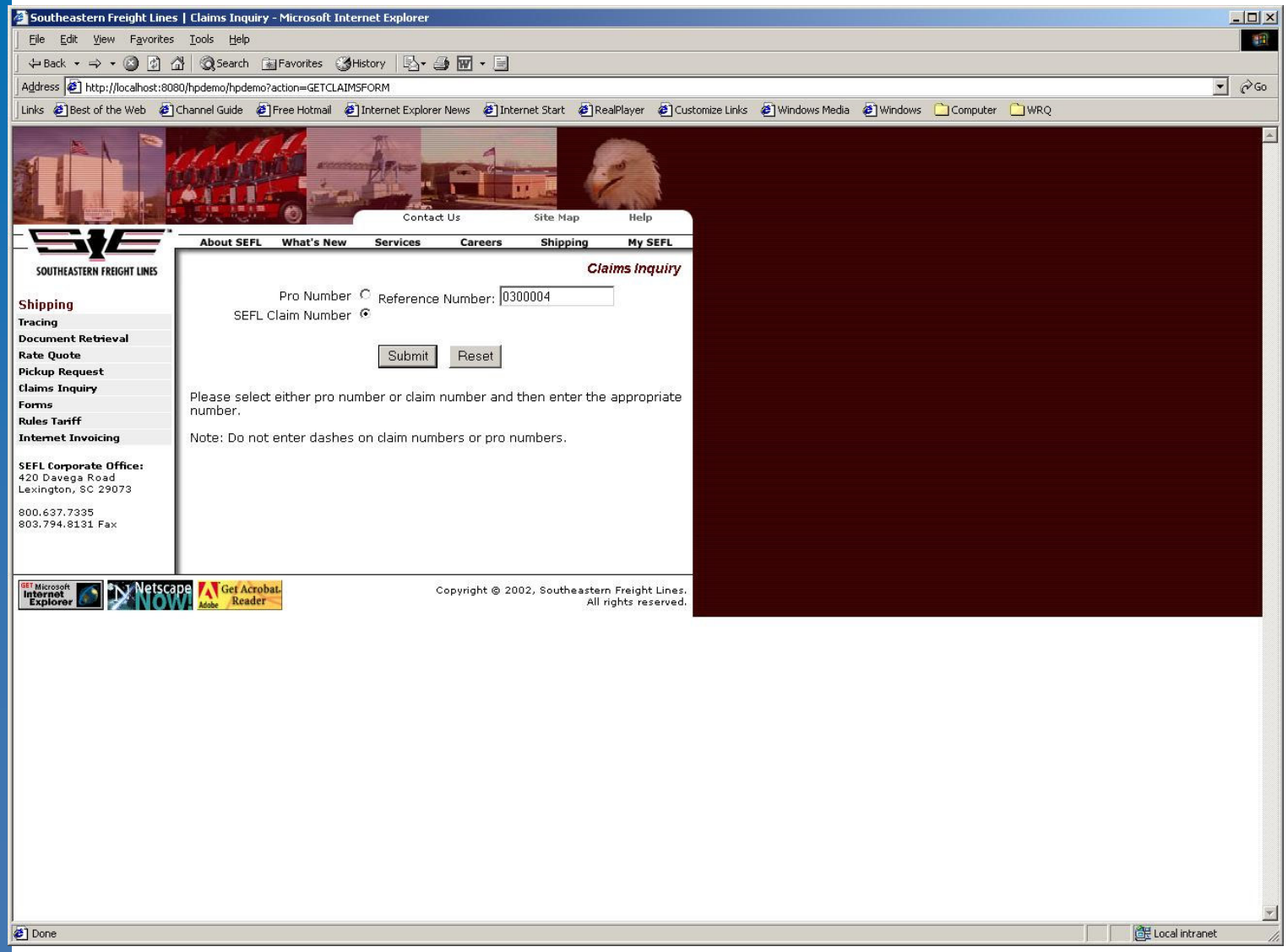
No matter what platform the web application is running on, you are authenticated against the cluster wide sysuaf.dat

Enter your OpenVMS username and password!

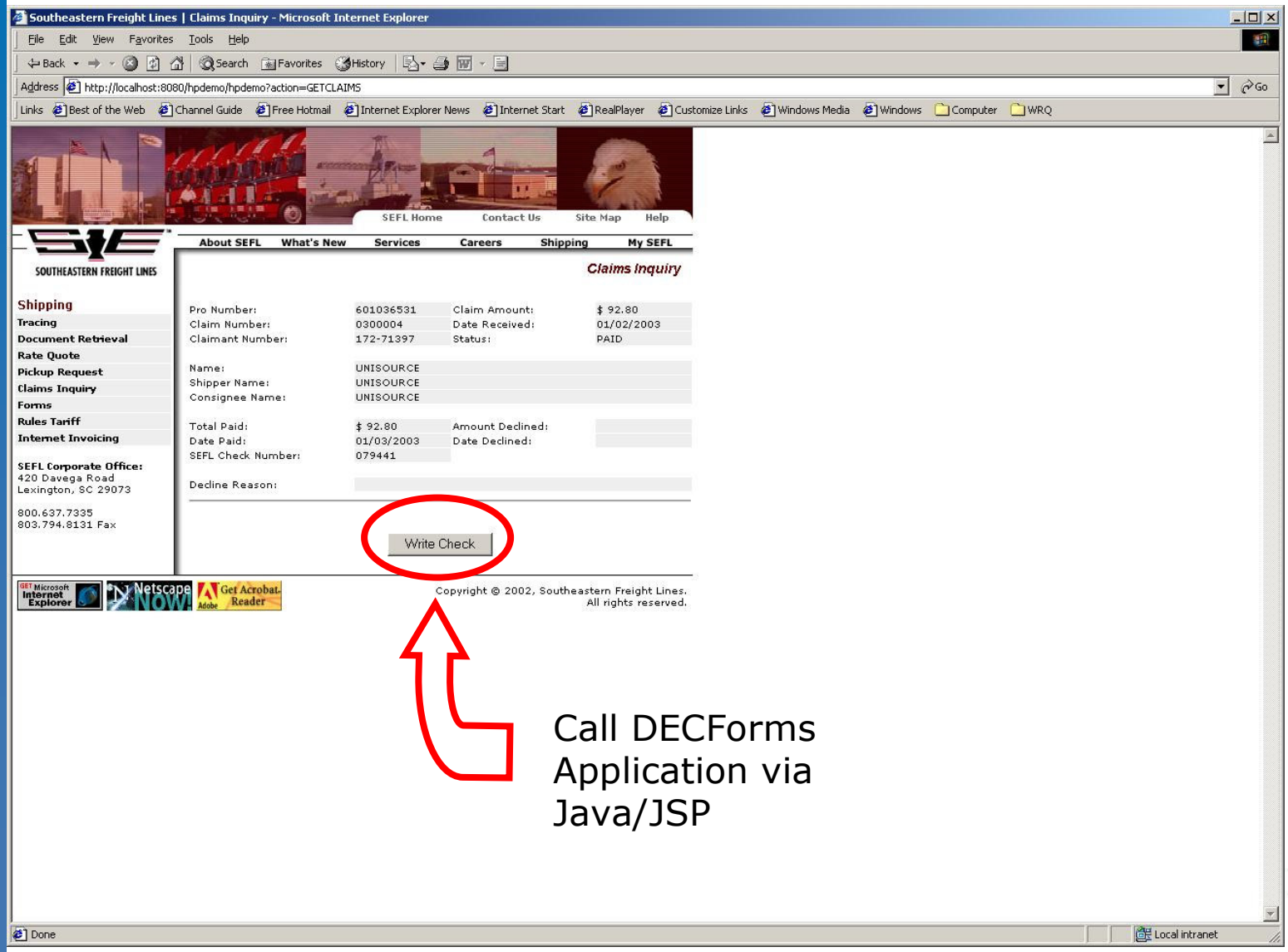
Initiating the Claims Inquiry Process



Query for Damaged Freight Claim Data—OpenVMS RMS Data Files

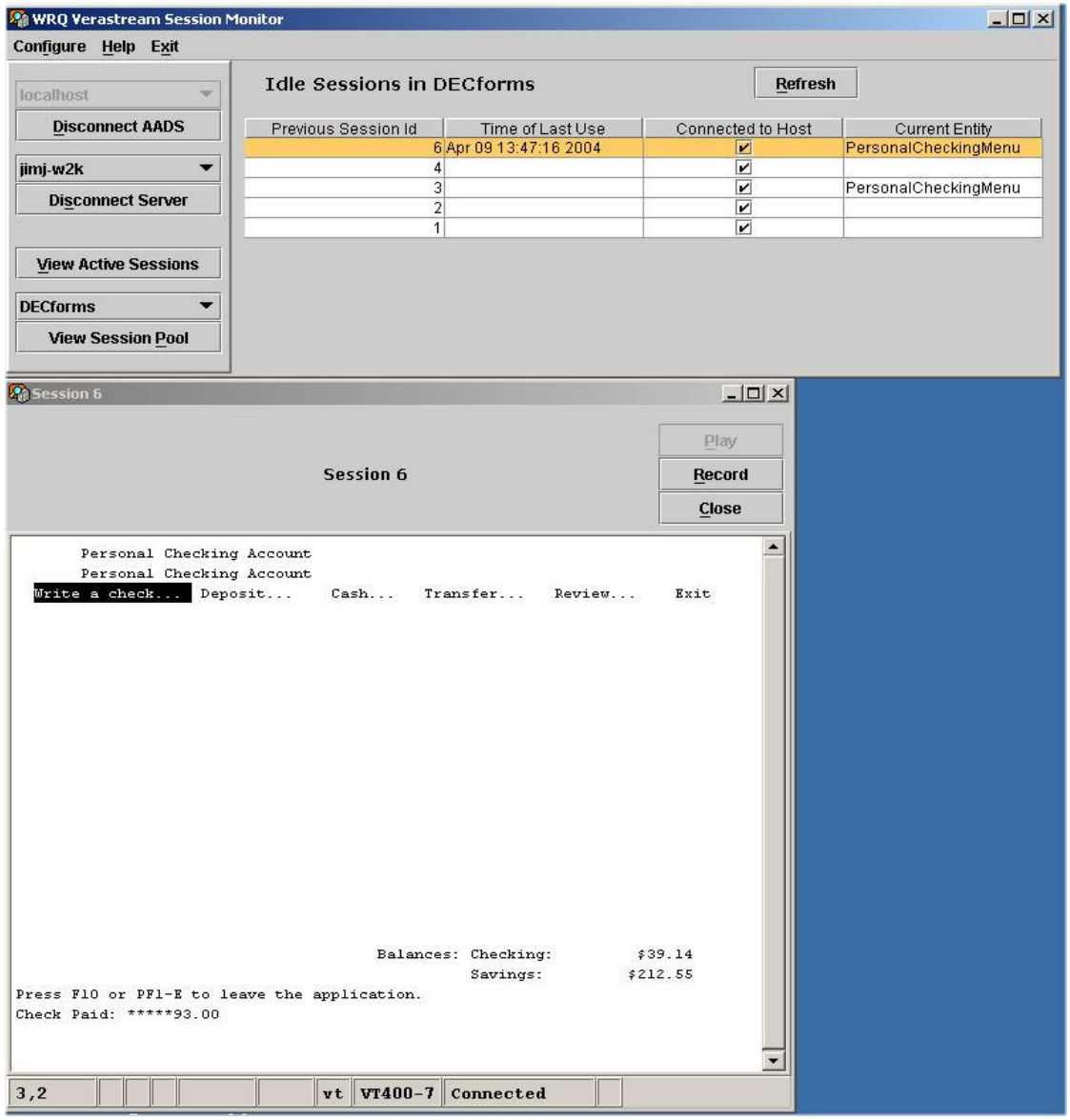


Paying the Damage Claim

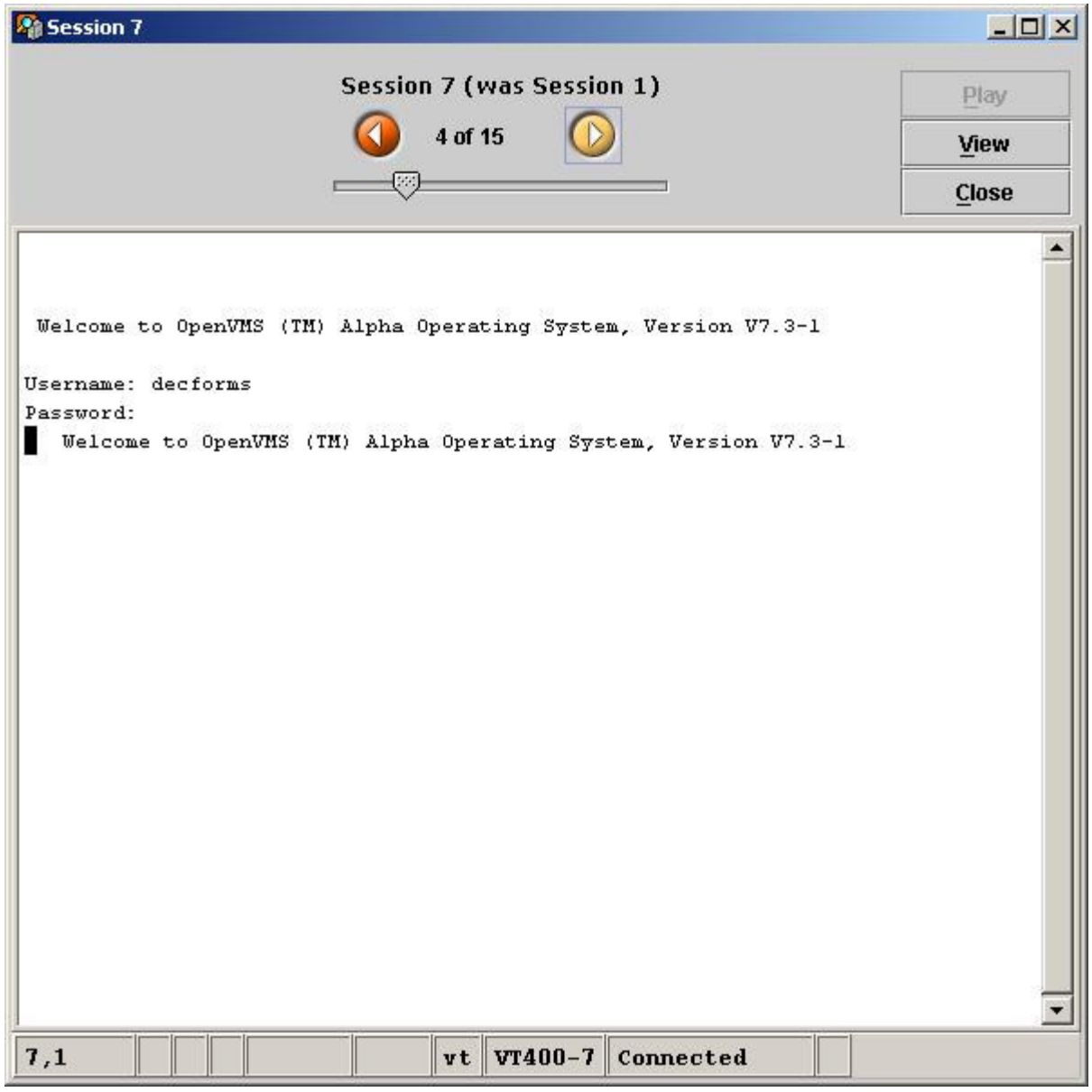


Call DECForms Application via Java/JSP

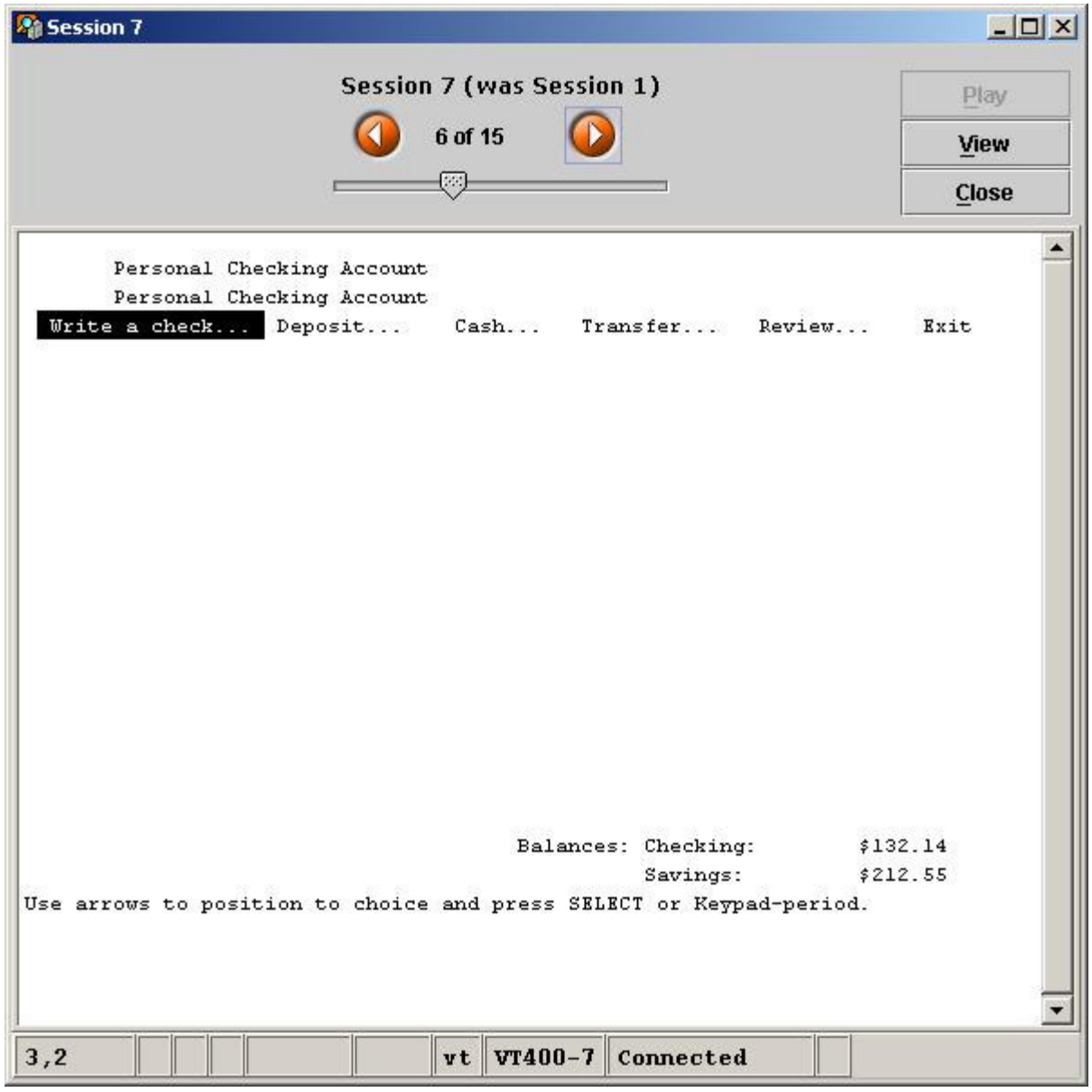
Verastream Host Adapter Session Pools



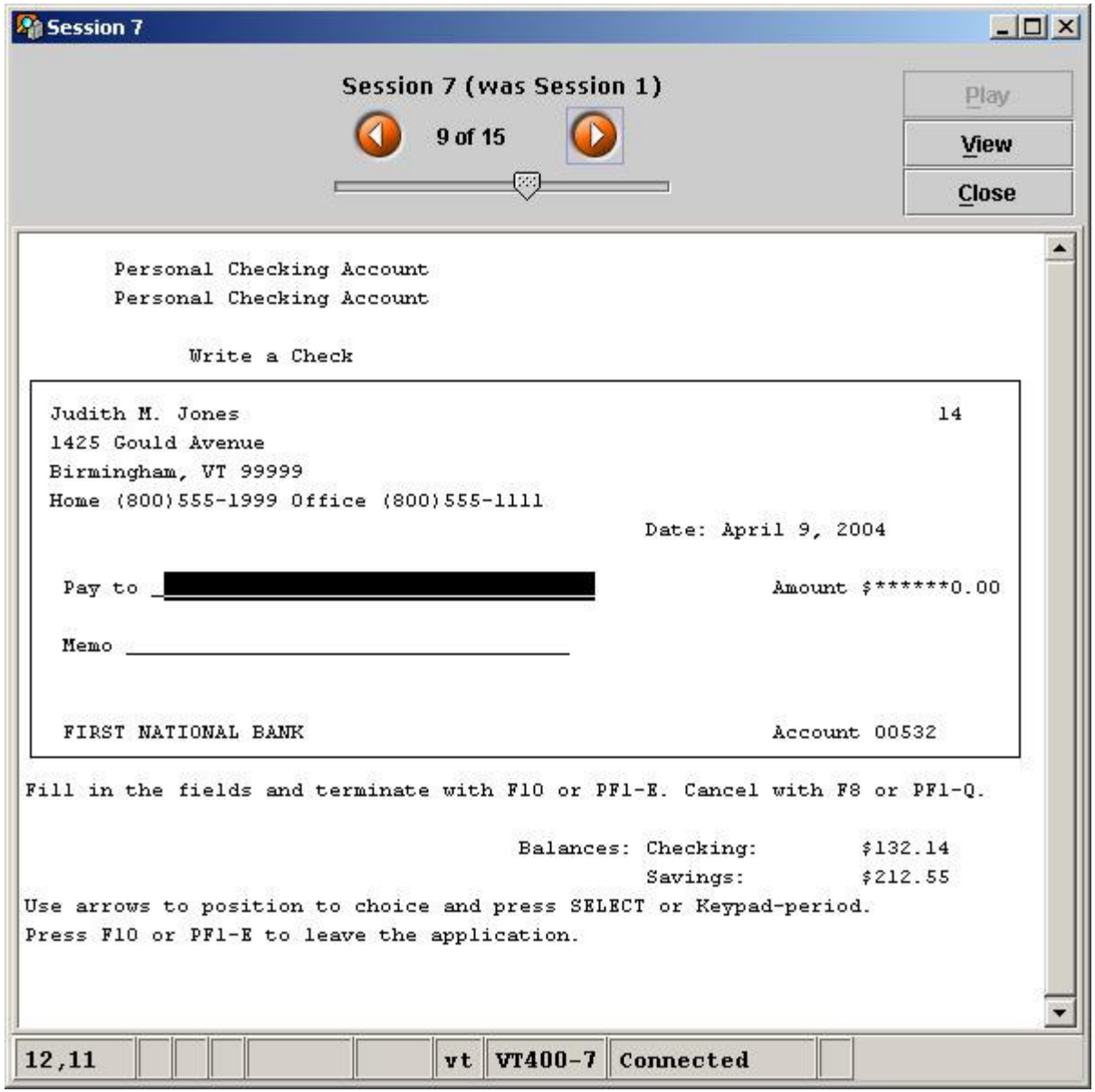
Behind the Scenes—
DECforms Checking Transaction Execution via Verastream Host Adapter Component



Behind the Scenes—
DECforms Checking Transaction Execution via Verastream Host Adapter Component

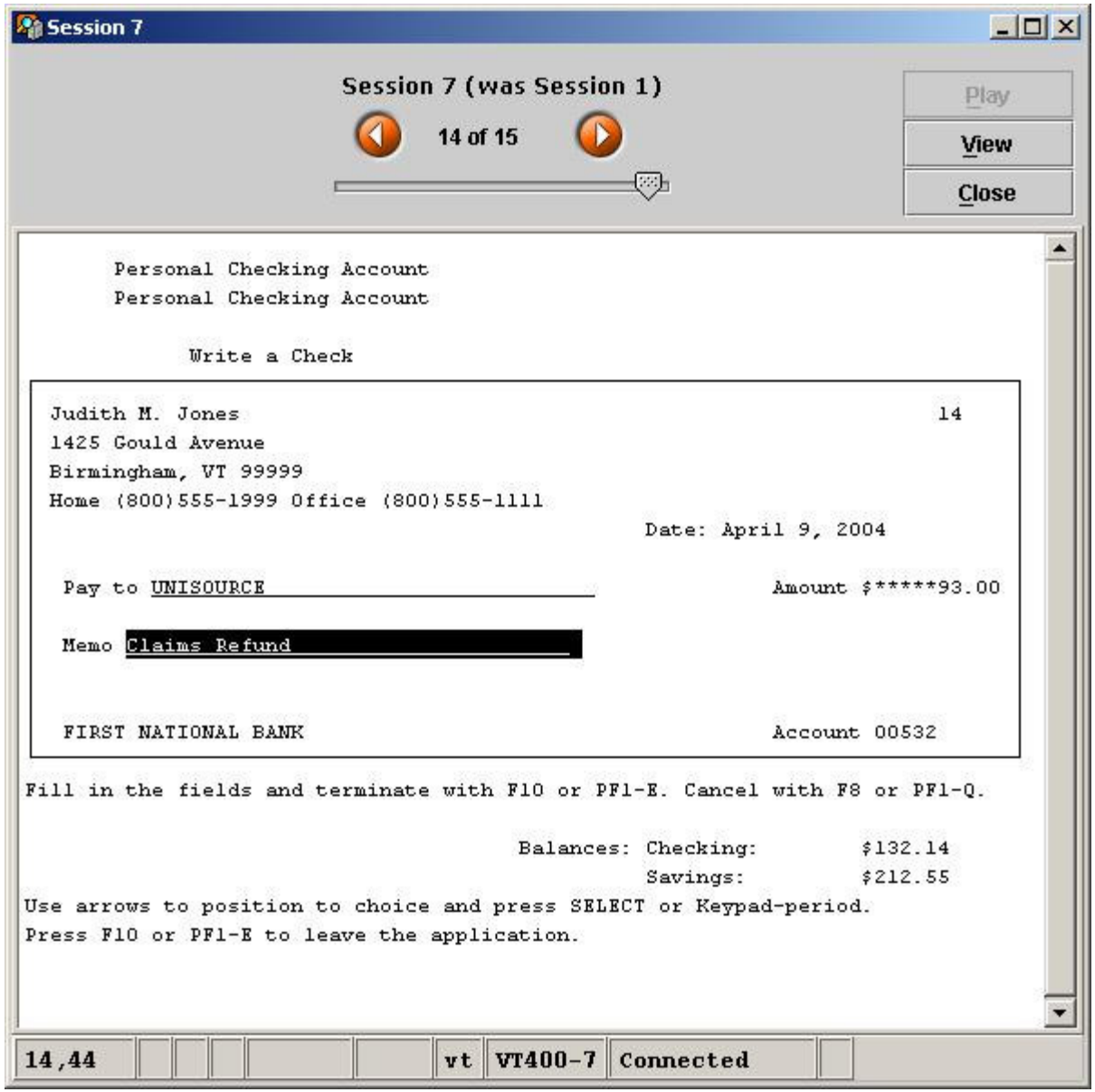


Behind the Scenes—
DECforms Checking Transaction Execution via Verastream Host Adapter Component

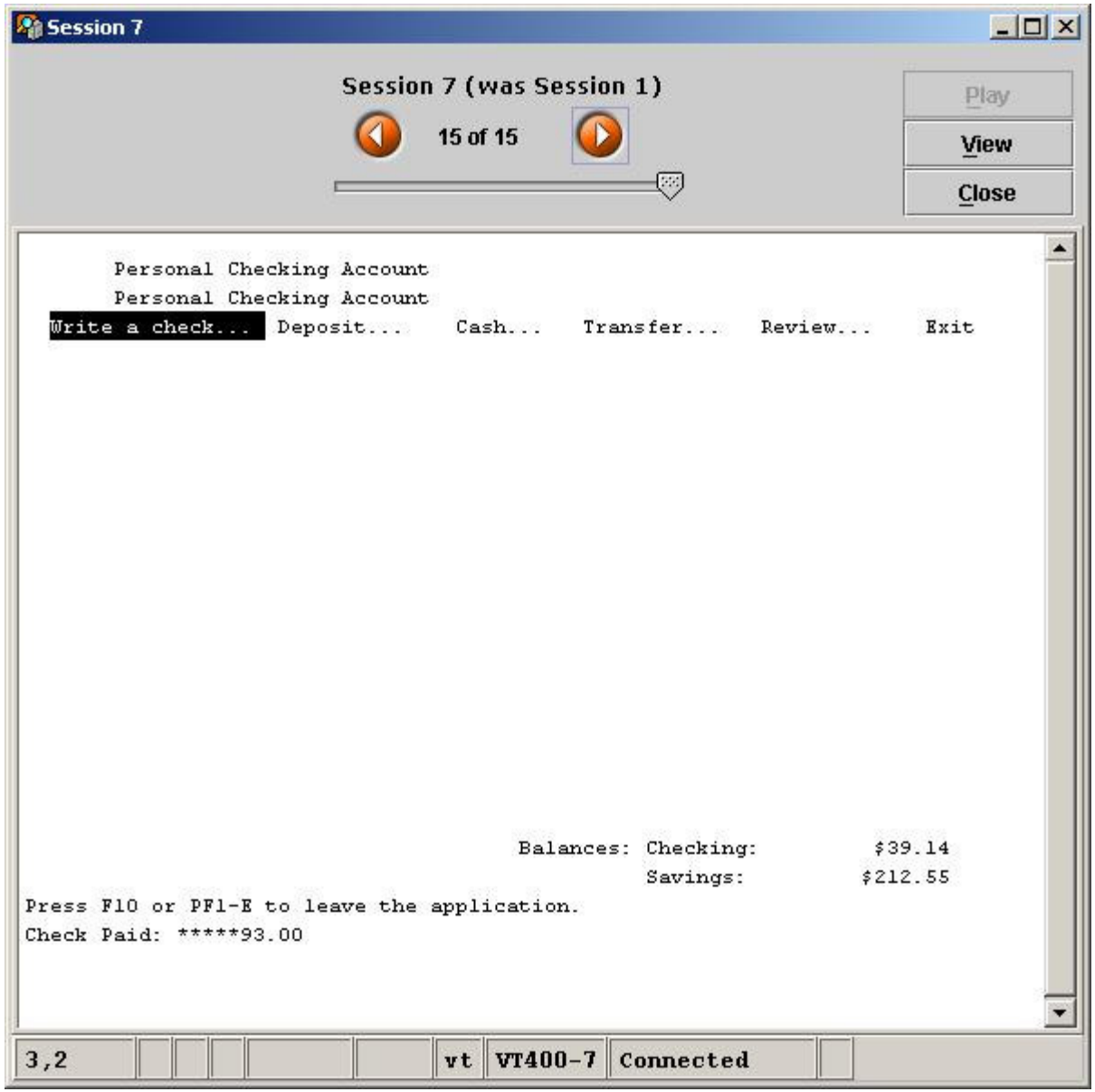


Behind the Scenes—

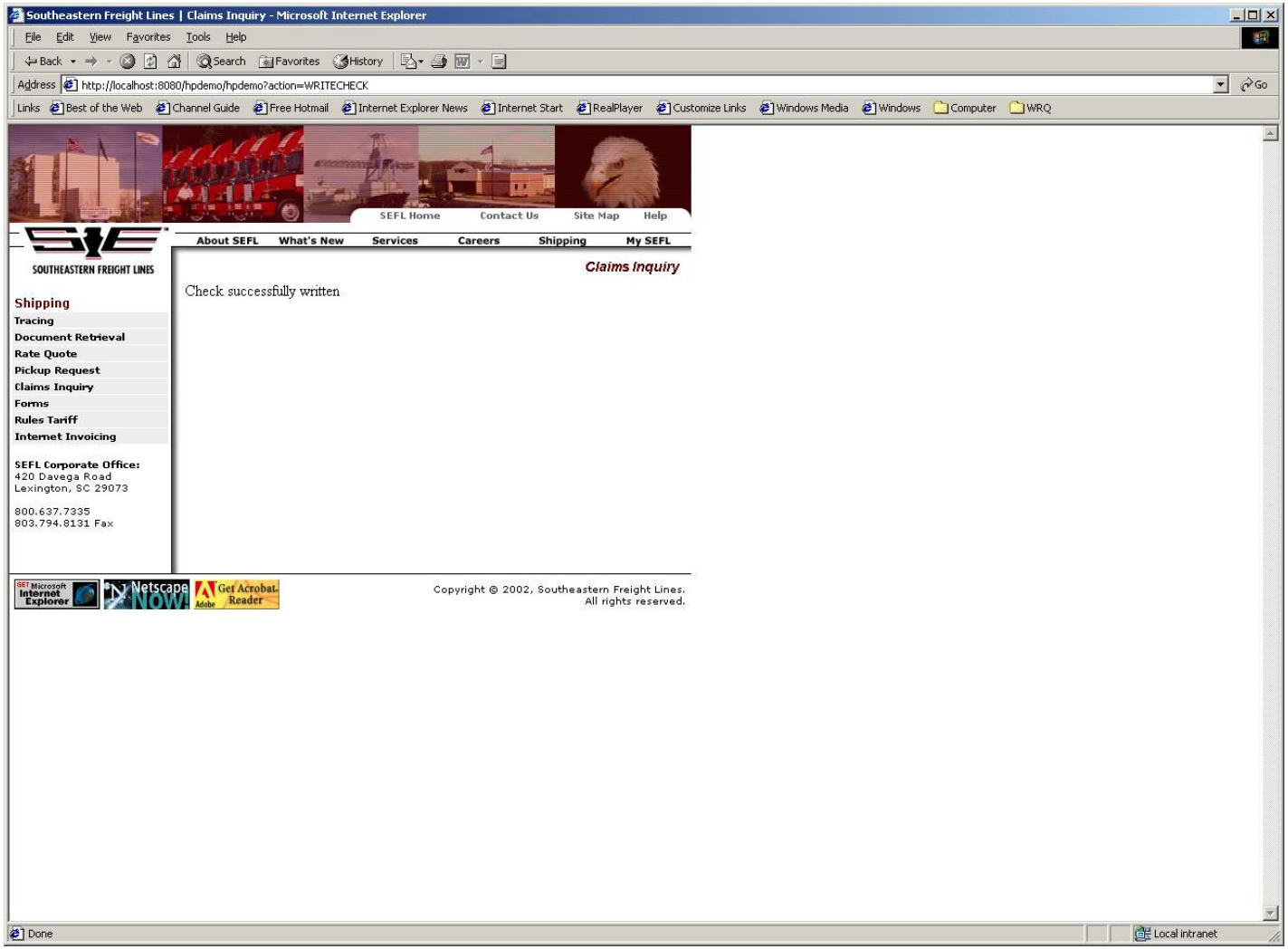
DECforms Checking Transaction Execution via Verastream Host Adapter Component



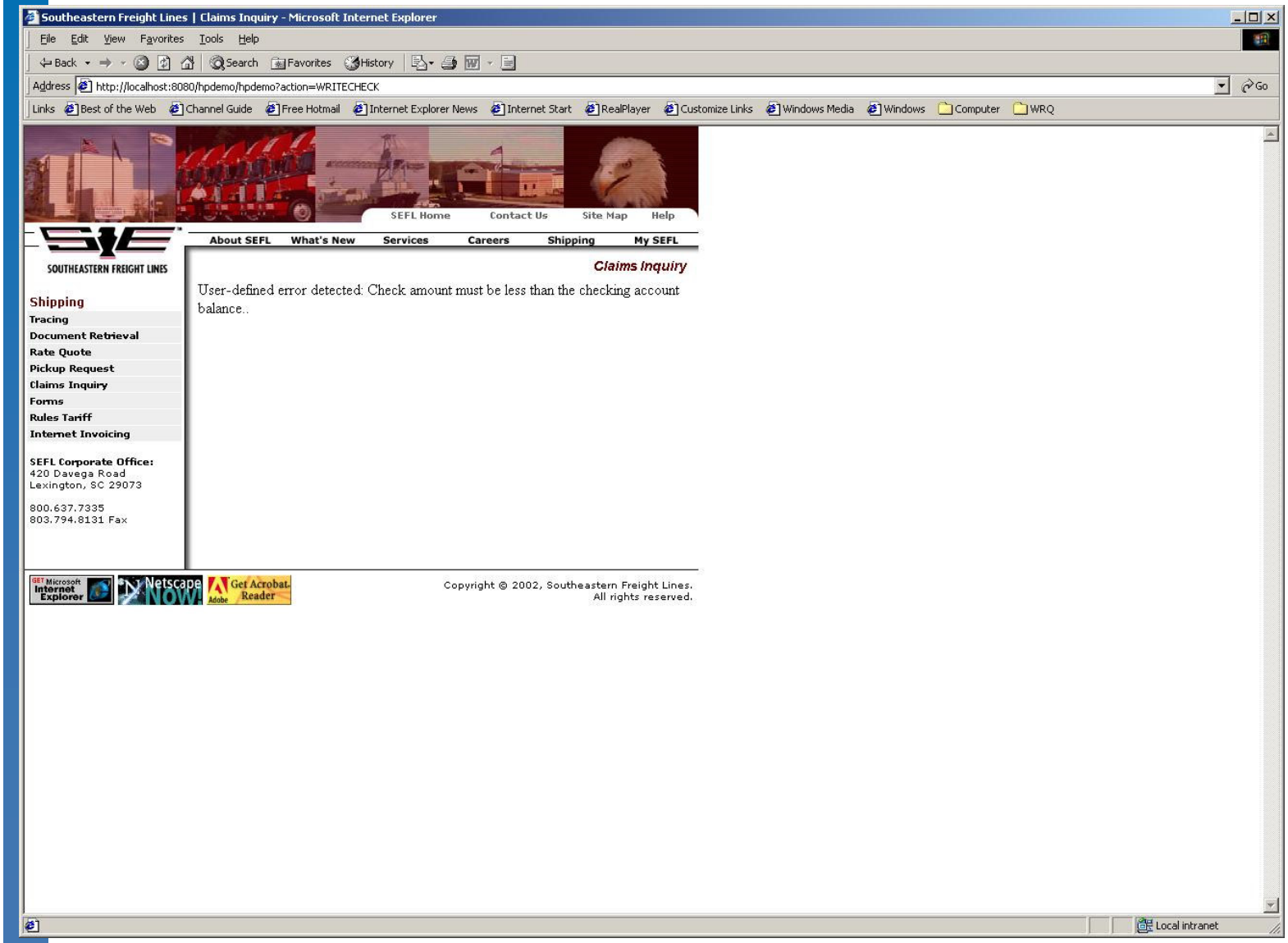
Behind the Scenes—
DECforms Checking Transaction Execution via Verastream Host Adapter Component



A Successful Transaction!



Ooops!!
Robust Error/Exception Handling



Composite Application Advantages

Faster, better results!

- Designed to work the way your business does…Allows you to tailor applications and integration to YOUR business needs instead of tailoring your business to the software!
- Reuse the thousands of business functions/ applications already proven and in production
- Minimize testing and risk…No alterations to existing code
- Highly flexible… Stands alone or complements other EAI infrastructures!
- Highly scalable... Built-in load balancing
- Proven success with many customers

Freight Claims Payment Composite Application

Hands-On Tutorial
Step by Step

Verastream Composite Application Overview

Freight Claims Payment—Composite Application

Combines/reuses/integrates:

- RMS Data used/maintained by existing OpenVMS Cobol Applications
- FMS/Fortran Character Based Checking Account Application wrapped with Verastream Host Integrator
- A legacy OpenVMS application written in C and wrapped with HP Bridgeworks to create a Java callable interface
- A modern easy-to-use WEB user interface developed using Java/JSP

Verastream Development Process

Freight Claims Payment

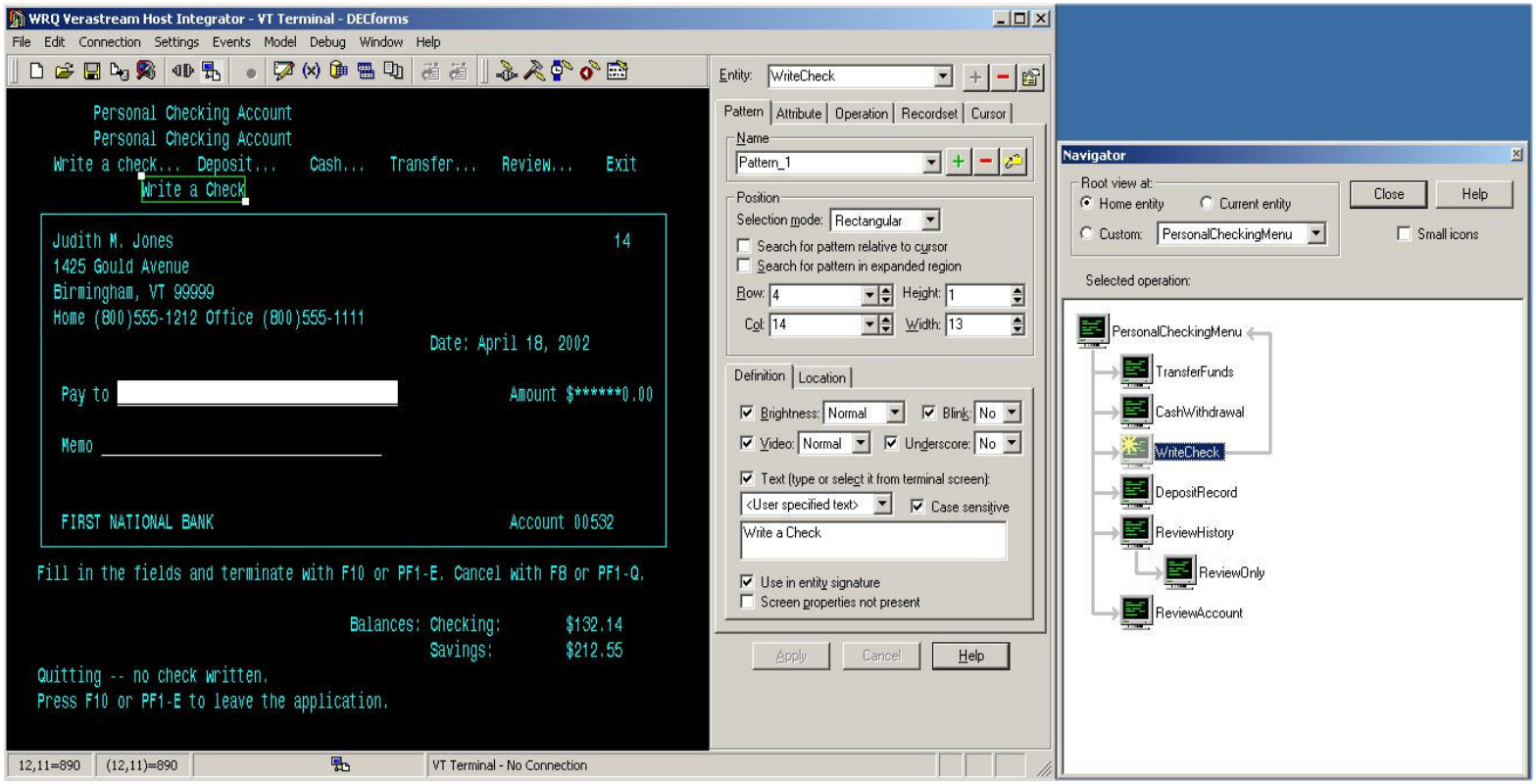
Composite Application Development Process

Five simple steps:

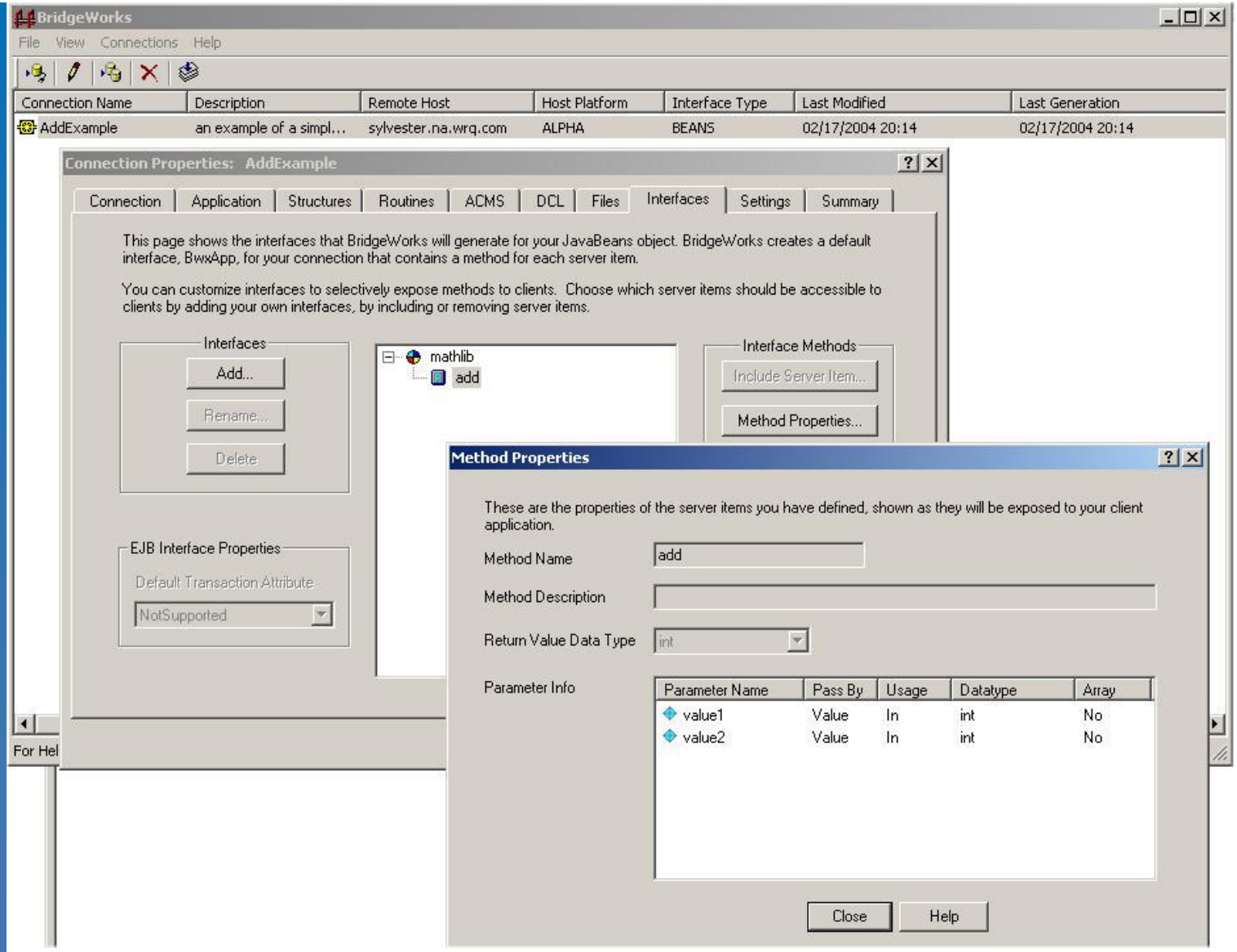
- Build the components (DECForms Verastream Host Integrator model, Bridgeworks Java Bean and Verastream Shipping Component)*
- Add the components to the Repository
- Assemble the composite application
- Deploy the composite application
- Test the composite application
- Mission Accomplished!

* - To save time, for this tutorial an example DECForms Checking model, Bridgeworks Java Bean and Java Servlets/JSP's have been pre-built for you)

Build the components
Verastream Host Integrator Model



Build the components Bridgeworks Java Bean

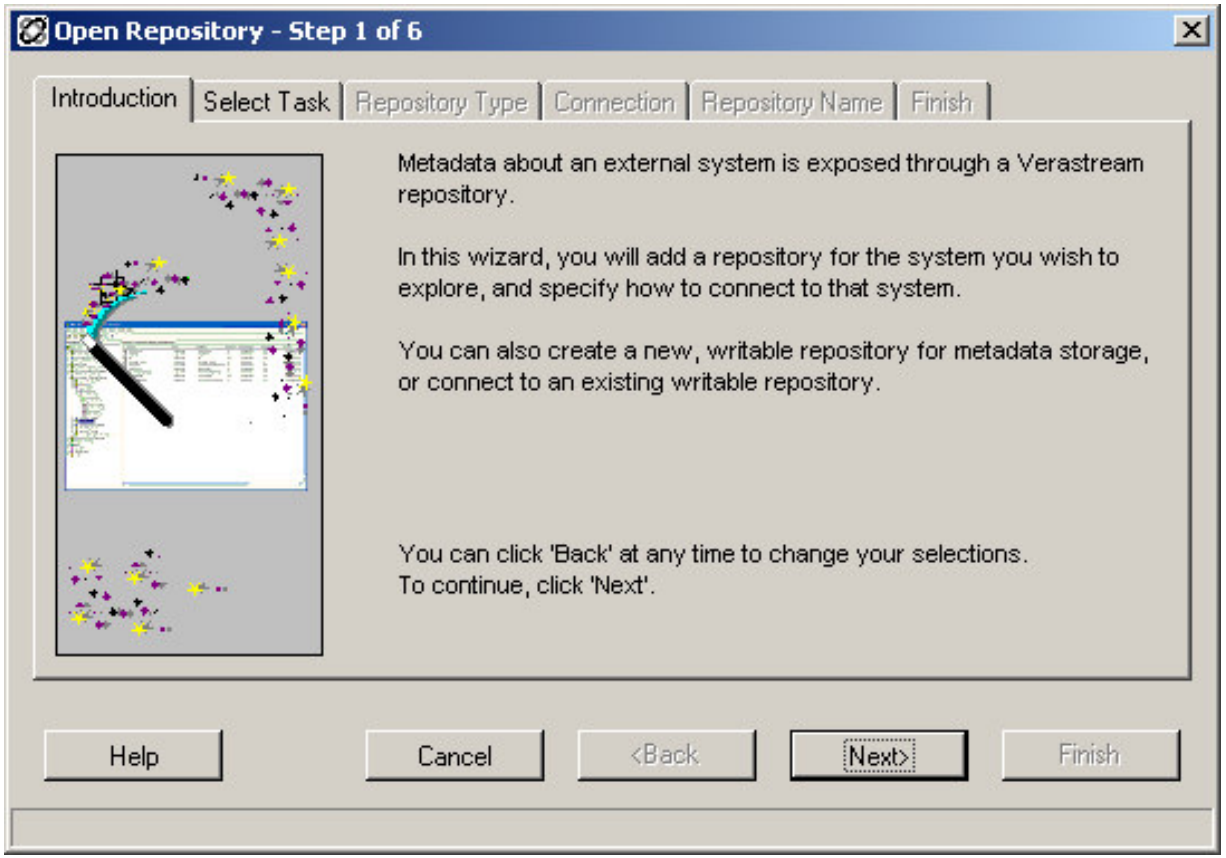


Build the components
Verastream Shipping Component – Composite Application

- Open a Table Repository(To import metadata of RMS data files)
- Open a Verastream Host Integrator Model Repository
- Open a Java Repository
- Create a Verastream Repository
- Define the Shipping Component
- Create “claimsdd” module file
- Realize(Implement) the Component
- Register(deploy) the Component
- Test the Component

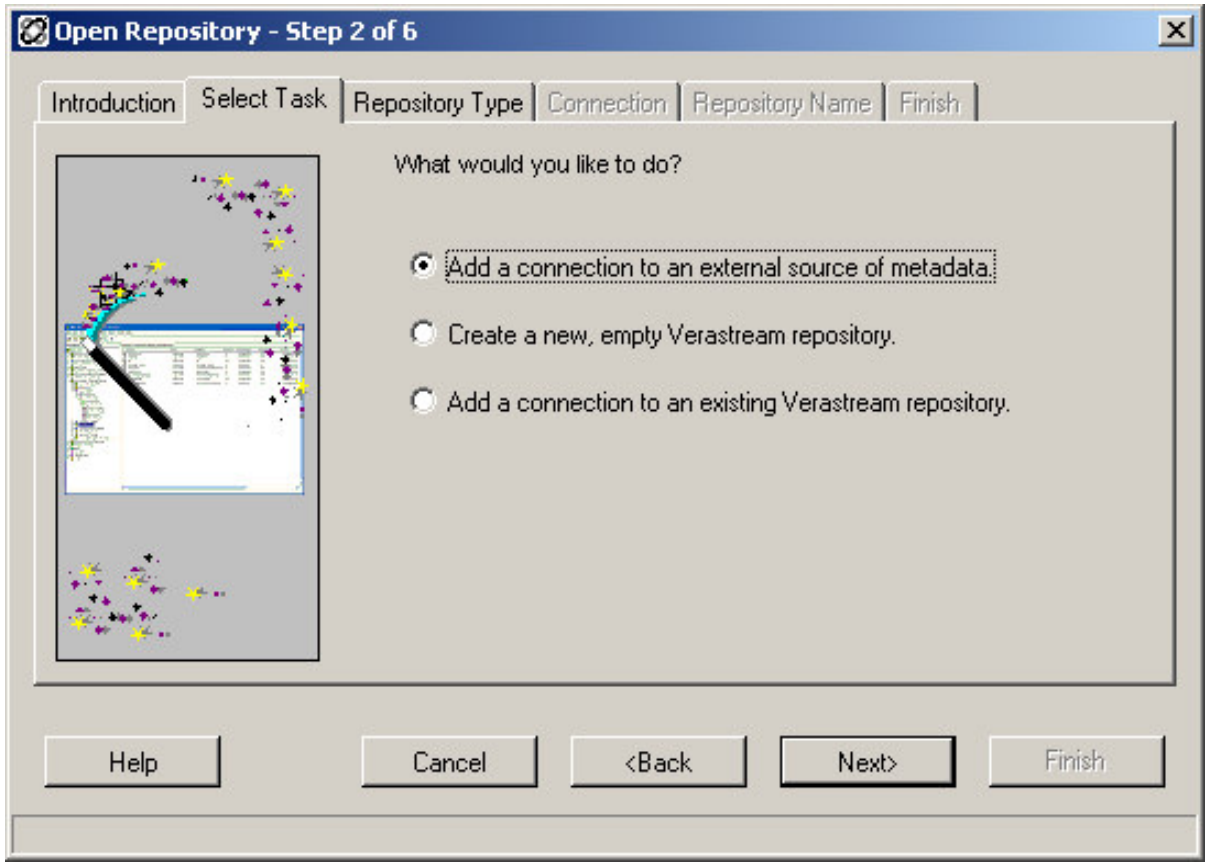
Building the Shipping Component
Open a Table Repository

- Launch Verastream Repository Manager
- Select the menu File → Open Repository



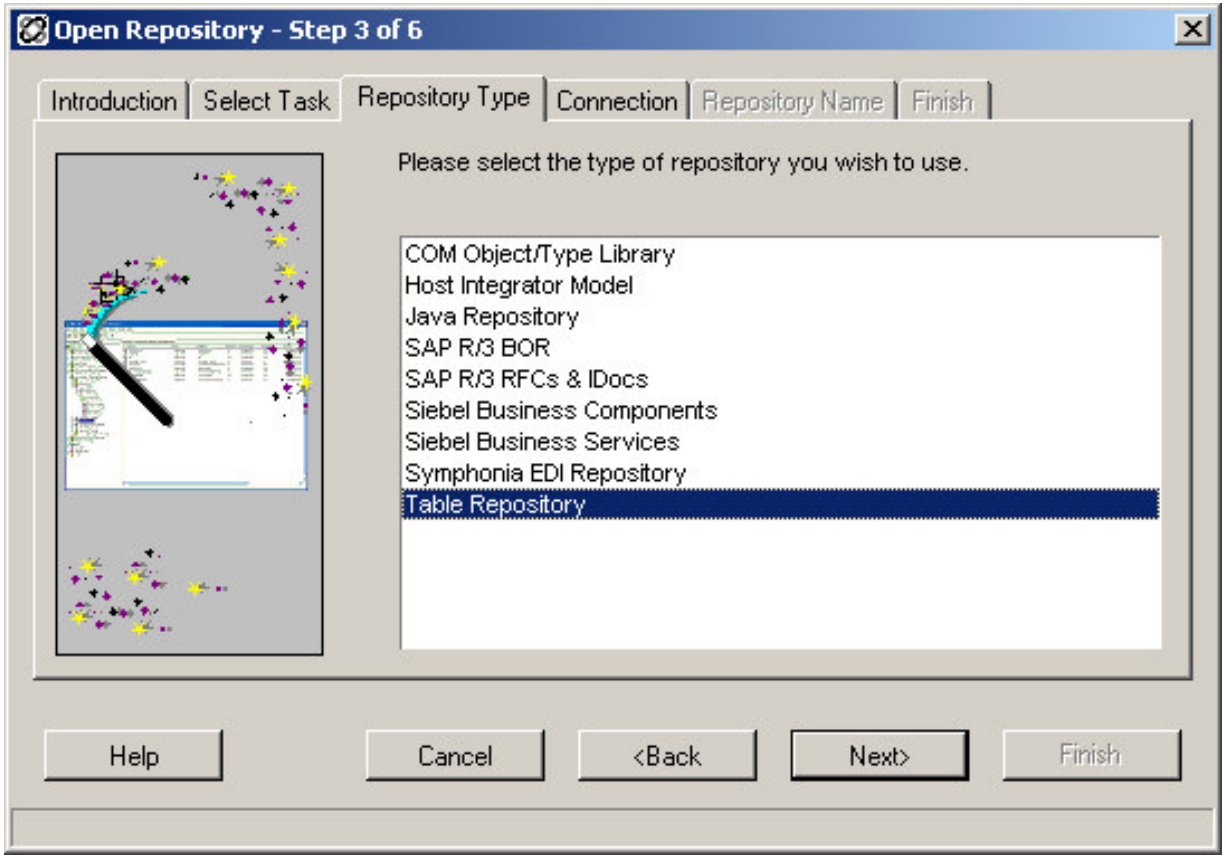
Building the Shipping Component
Open a Table Repository

- Add a connection to an external source of metadata



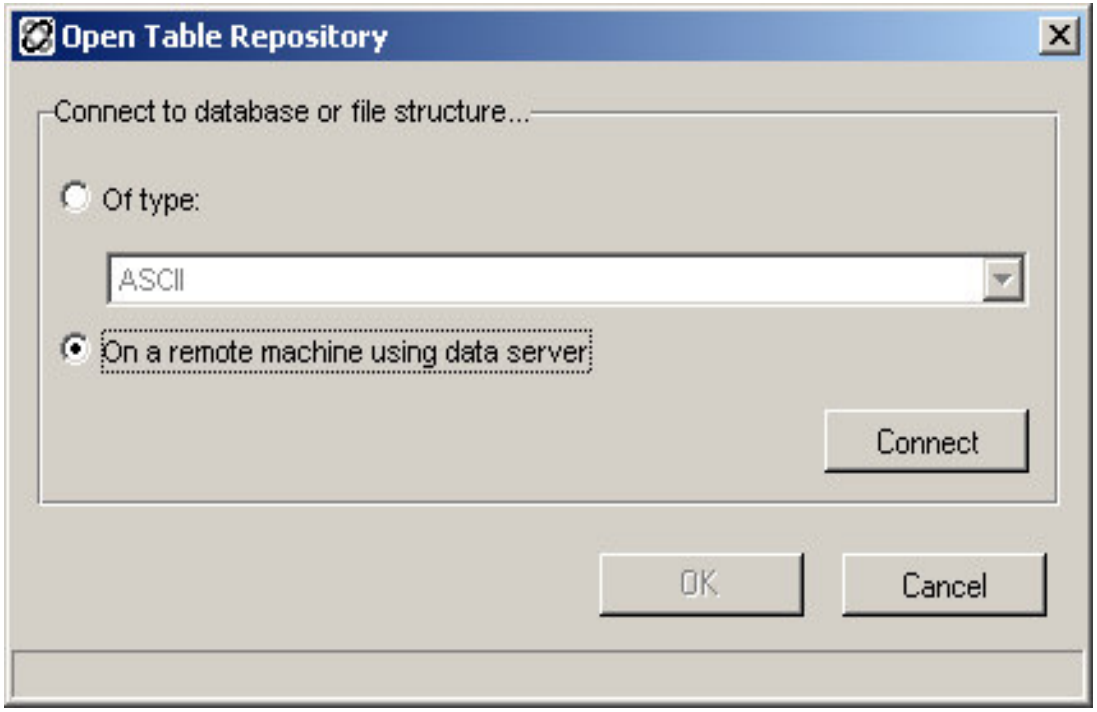
Building the Shipping Component
Open a Table Repository

- Select the Table Repository



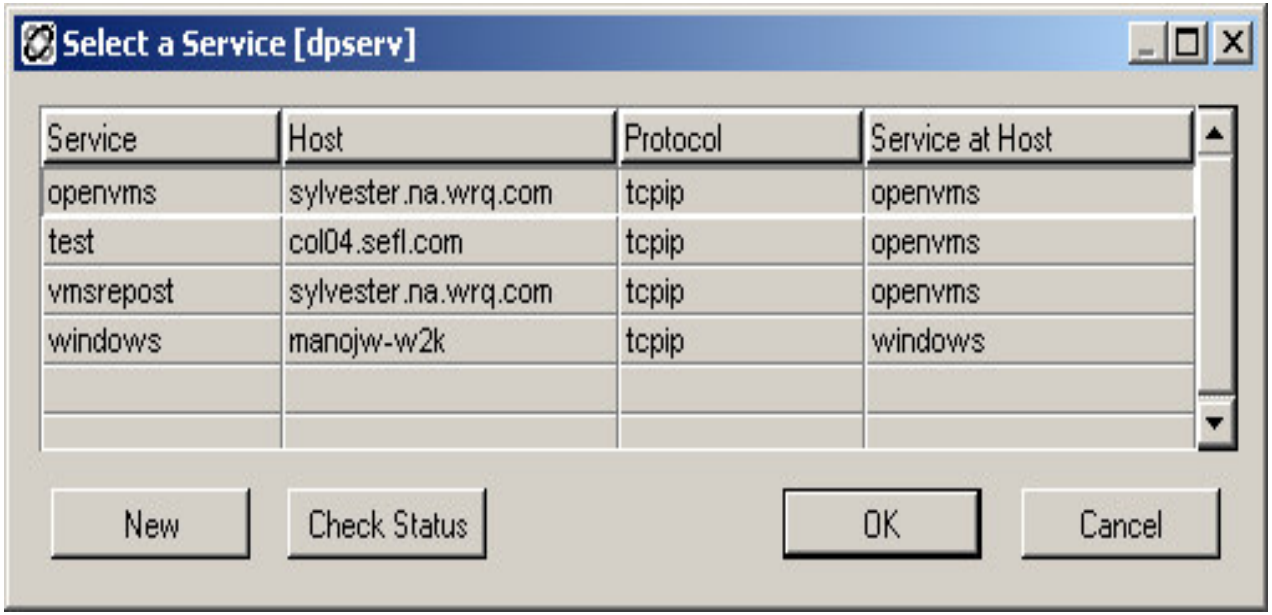
Building the Shipping Component
Open a Table Repository

- Connect to the database using data server



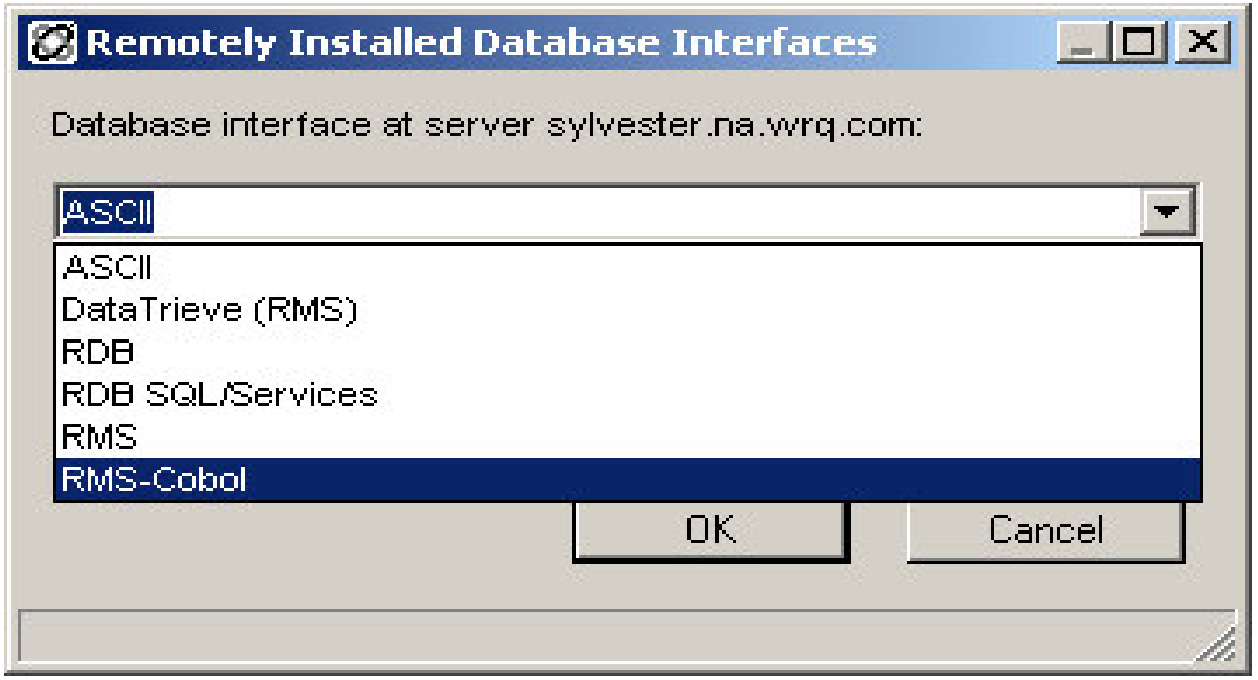
Building the Shipping Component
Open a Table Repository

- Select the client service



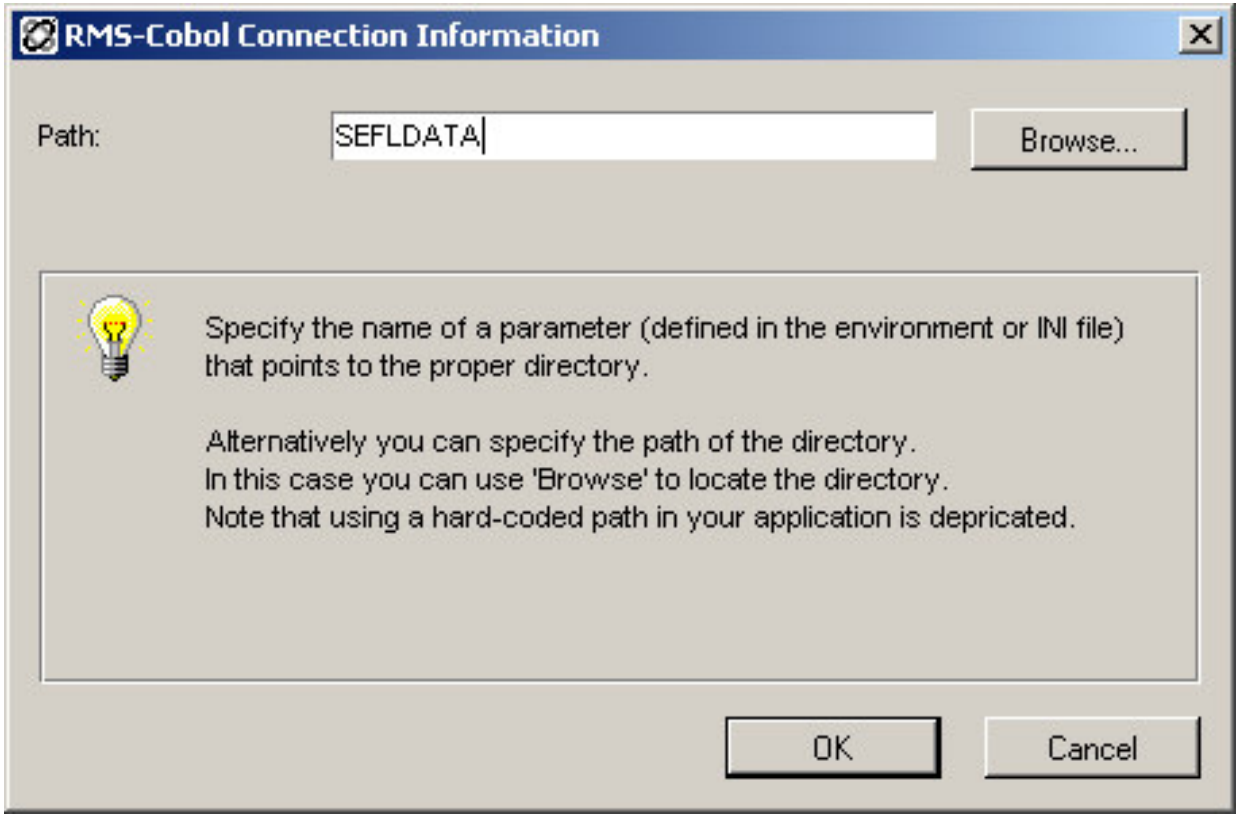
Building the Shipping Component
Open a Table Repository

- Select the database interface type - (RMS-Cobol)



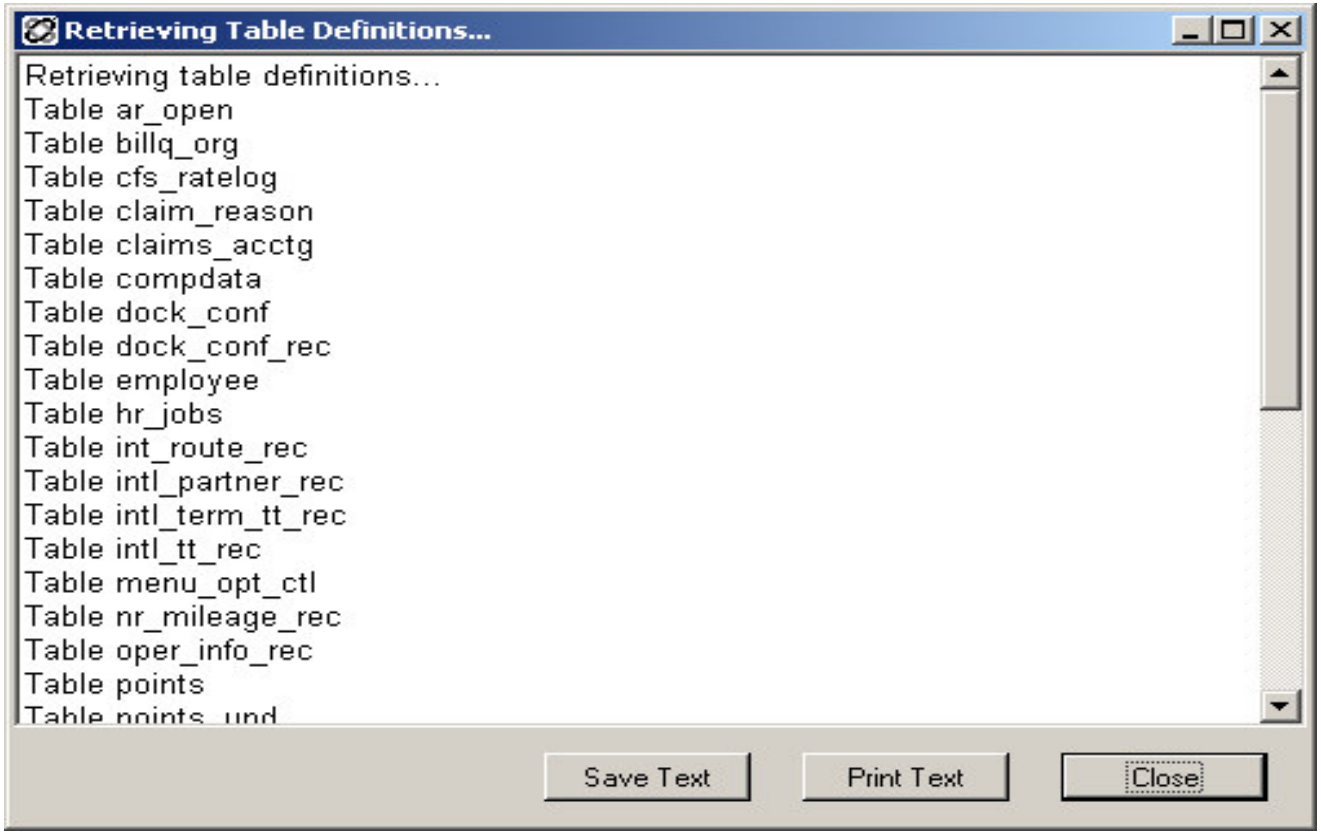
Building the Shipping Component
Open a Table Repository

- Specify path to the Cobol copy book



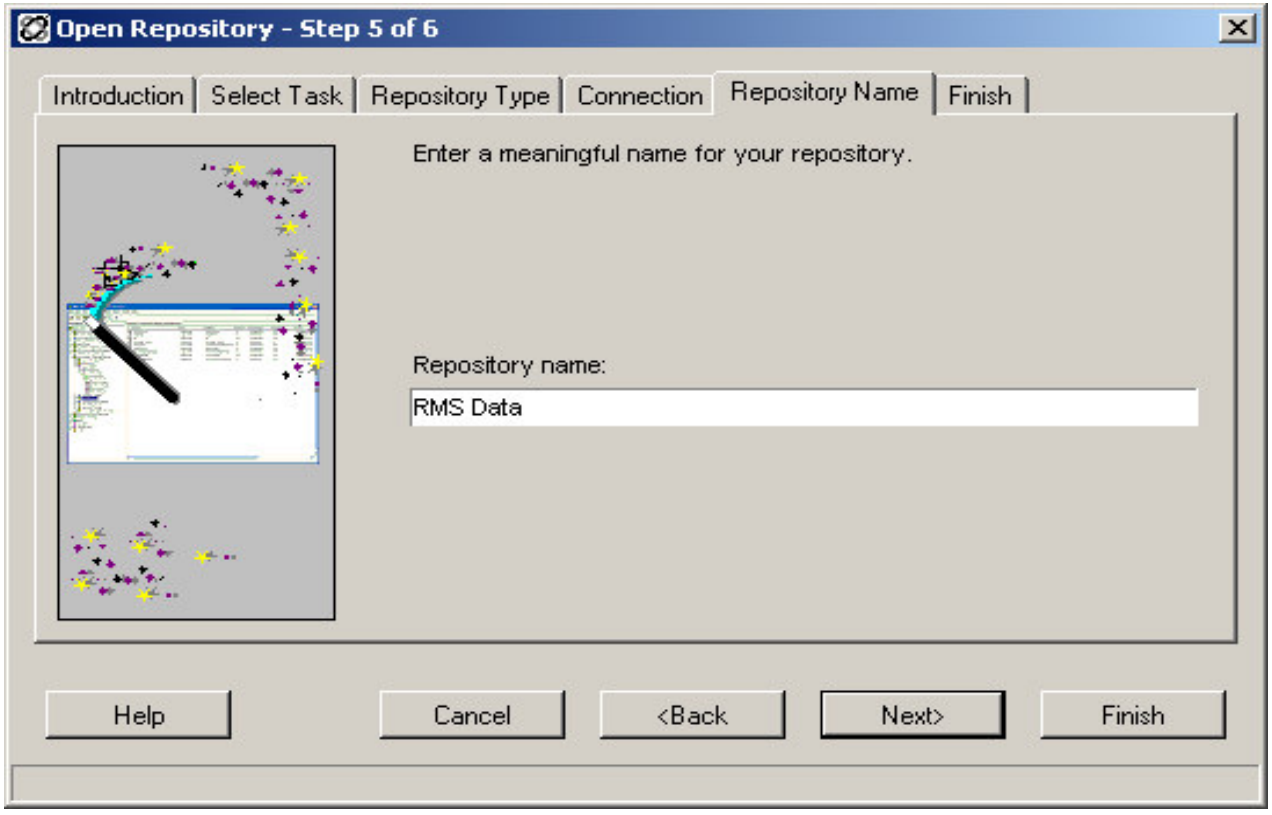
Building the Shipping Component
Open a Table Repository

- View the list of tables whose metadata is being imported



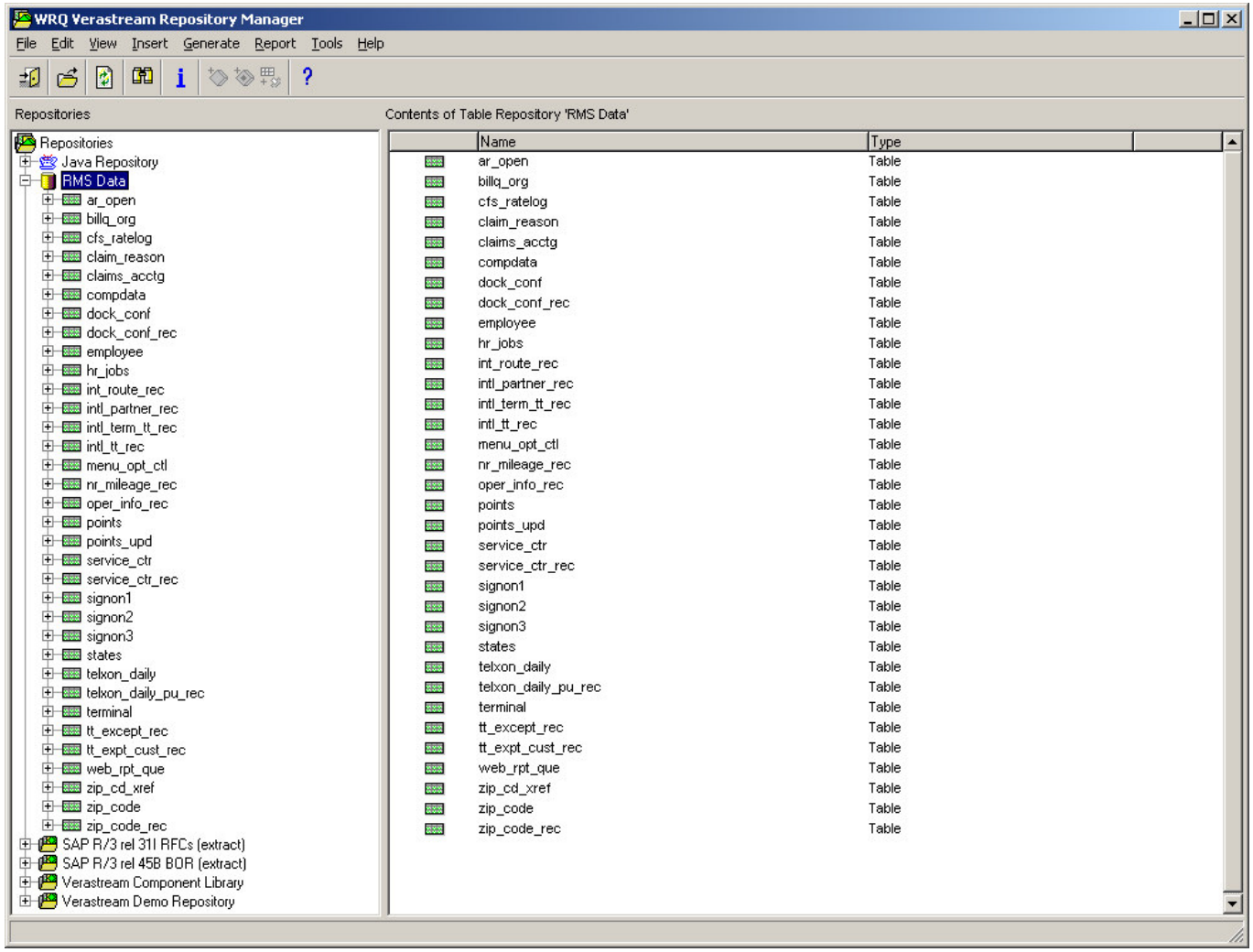
Building the Shipping Component
Open a Table Repository

- Name the Repository
- Click Finish



Building the Shipping Component
Open a Table Repository

- View the table repository

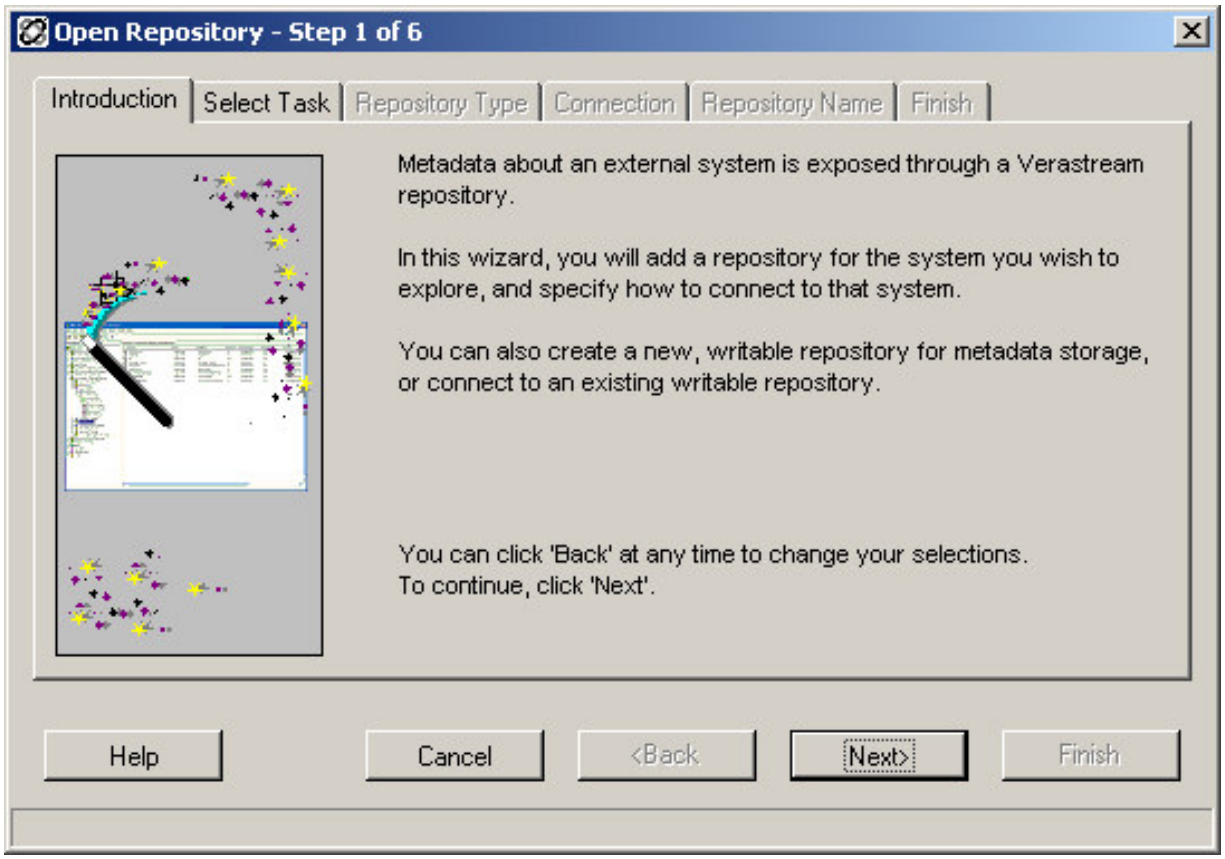


Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ❑ Open a Verastream Host Integrator Model Repository
- ❑ Open a Java Repository
- ❑ Create a Verastream Repository
- ❑ Define the Shipping Component
- ❑ Create “claimsdd” module file
- ❑ Realize(Implement) the Component
- ❑ Register(deploy) the Component
- ❑ Test the Component

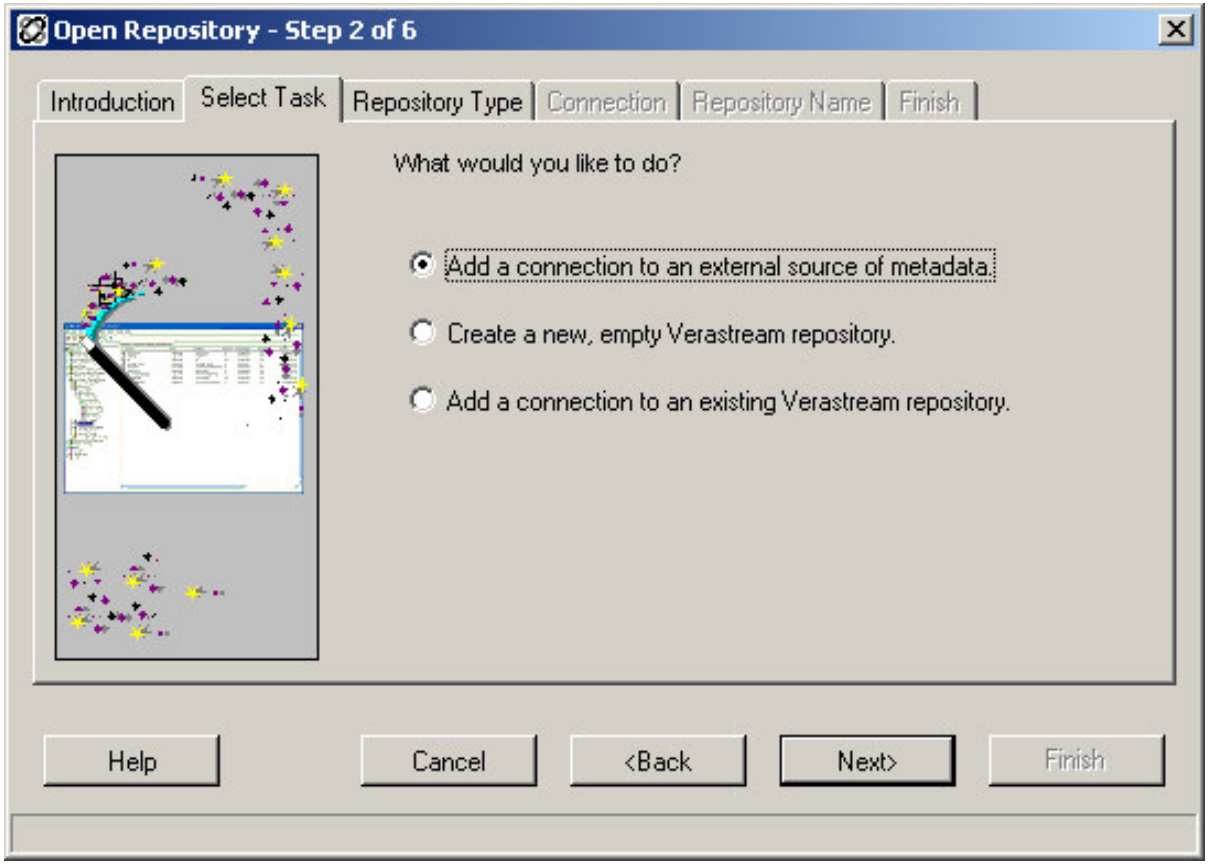
Building the Shipping Component
Open a Host Integrator Model Repository

- Go to Verastream Repository Manager
- Select the menu File → Open Repository



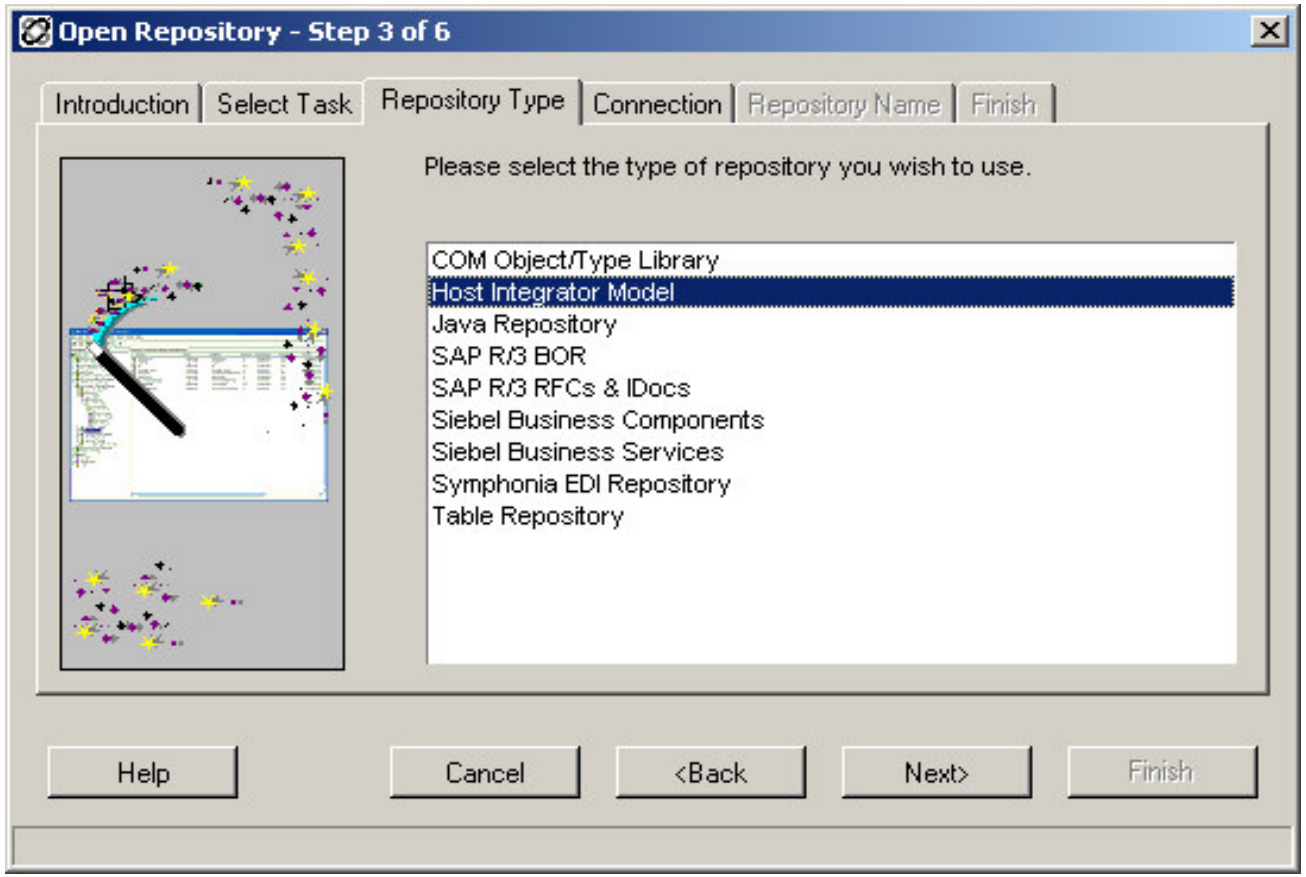
Building the Shipping Component
Open a Table Repository

- Add a connection to an external source of metadata



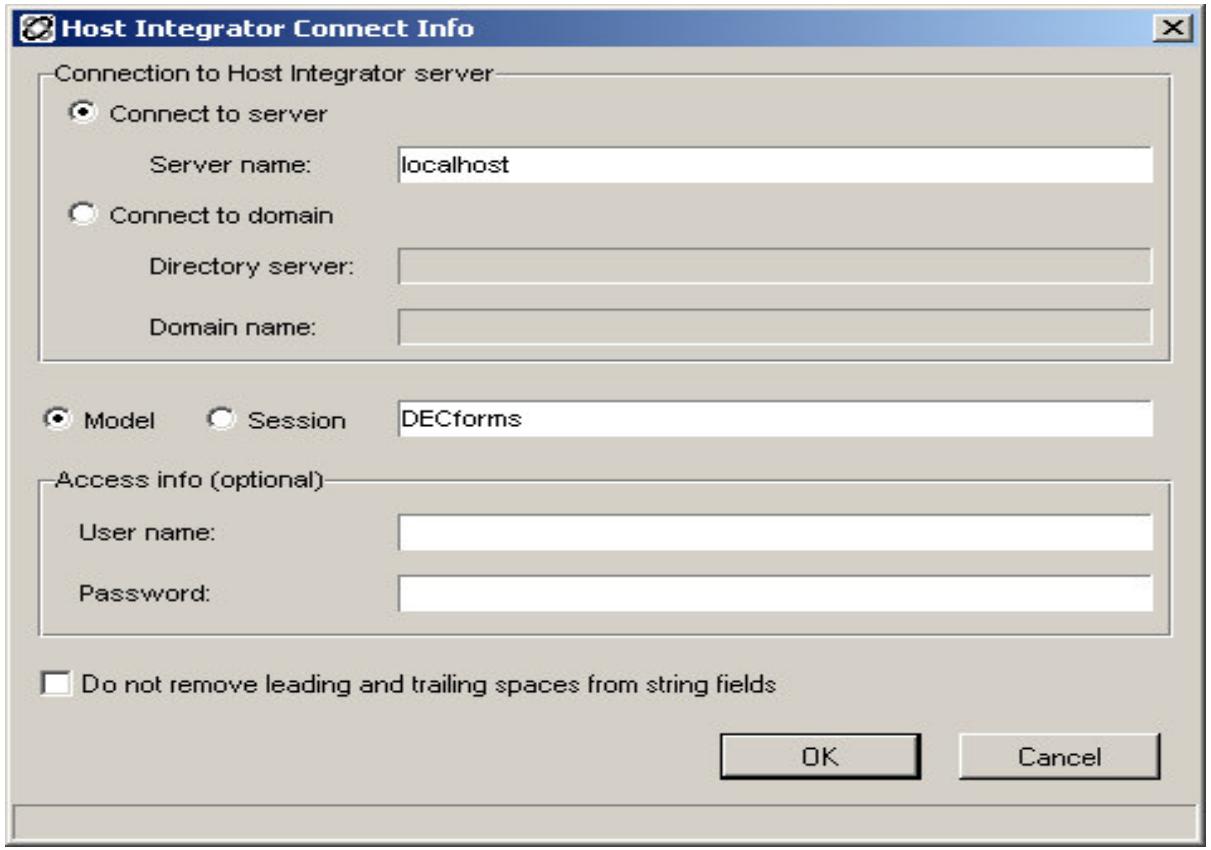
Building the Shipping Component
Open a Table Repository

- Select Host Integrator Model



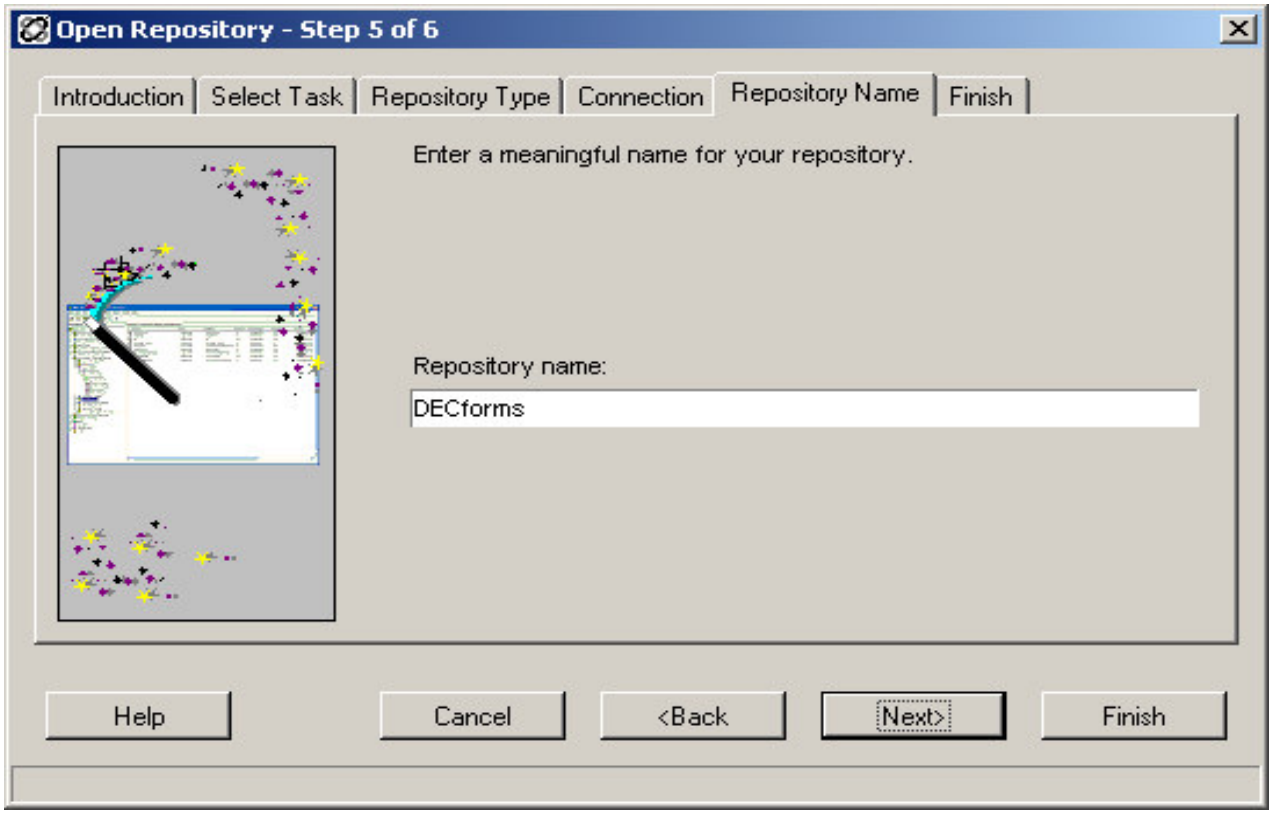
Building the Shipping Component
Open a Table Repository

- Provide the connection information to import the model metadata



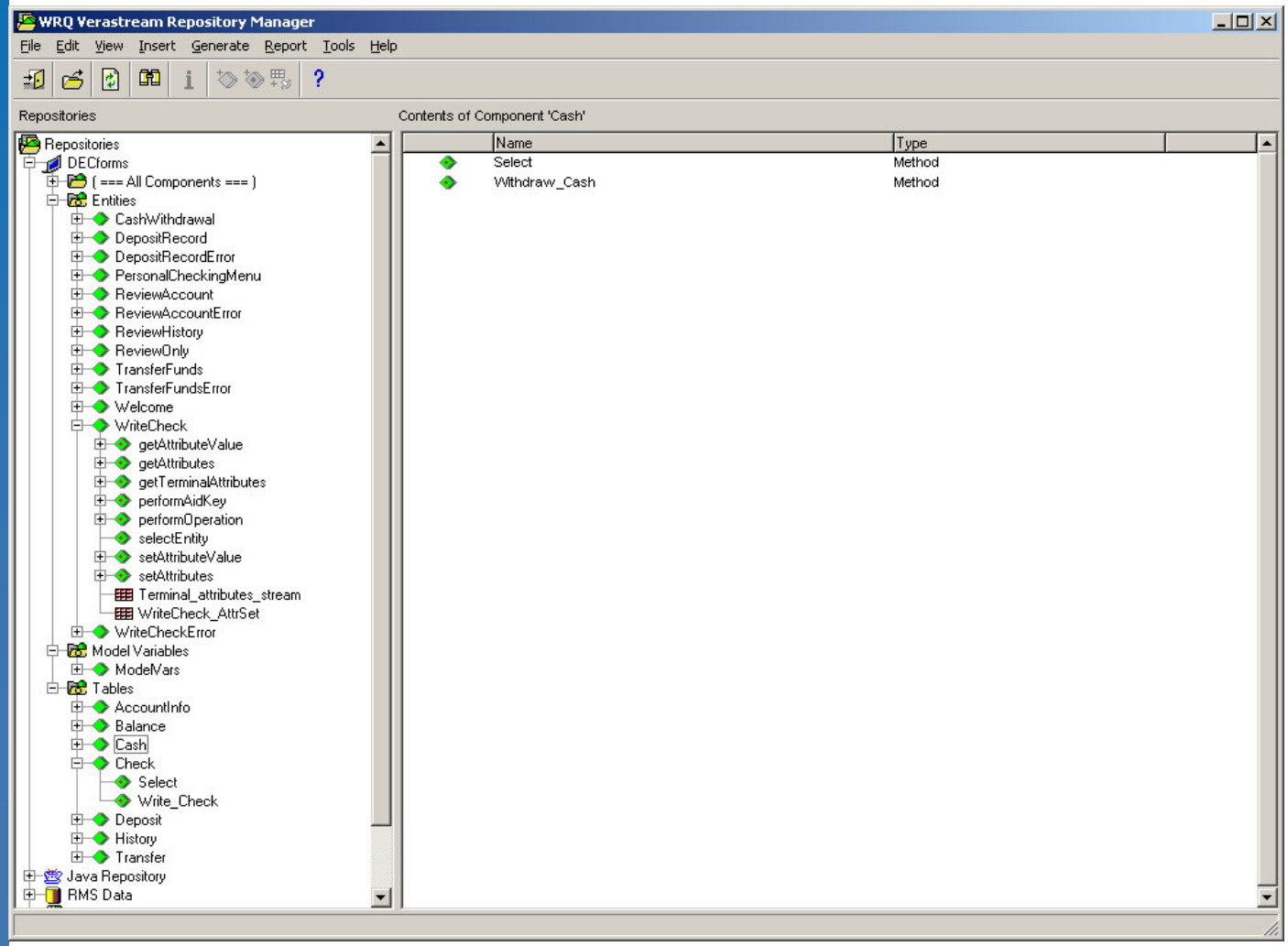
Building the Shipping Component
Open a Table Repository

- Name the Repository
- Click Finish



Building the Shipping Component
Open a Table Repository

- View the Host Integrator Model Repository

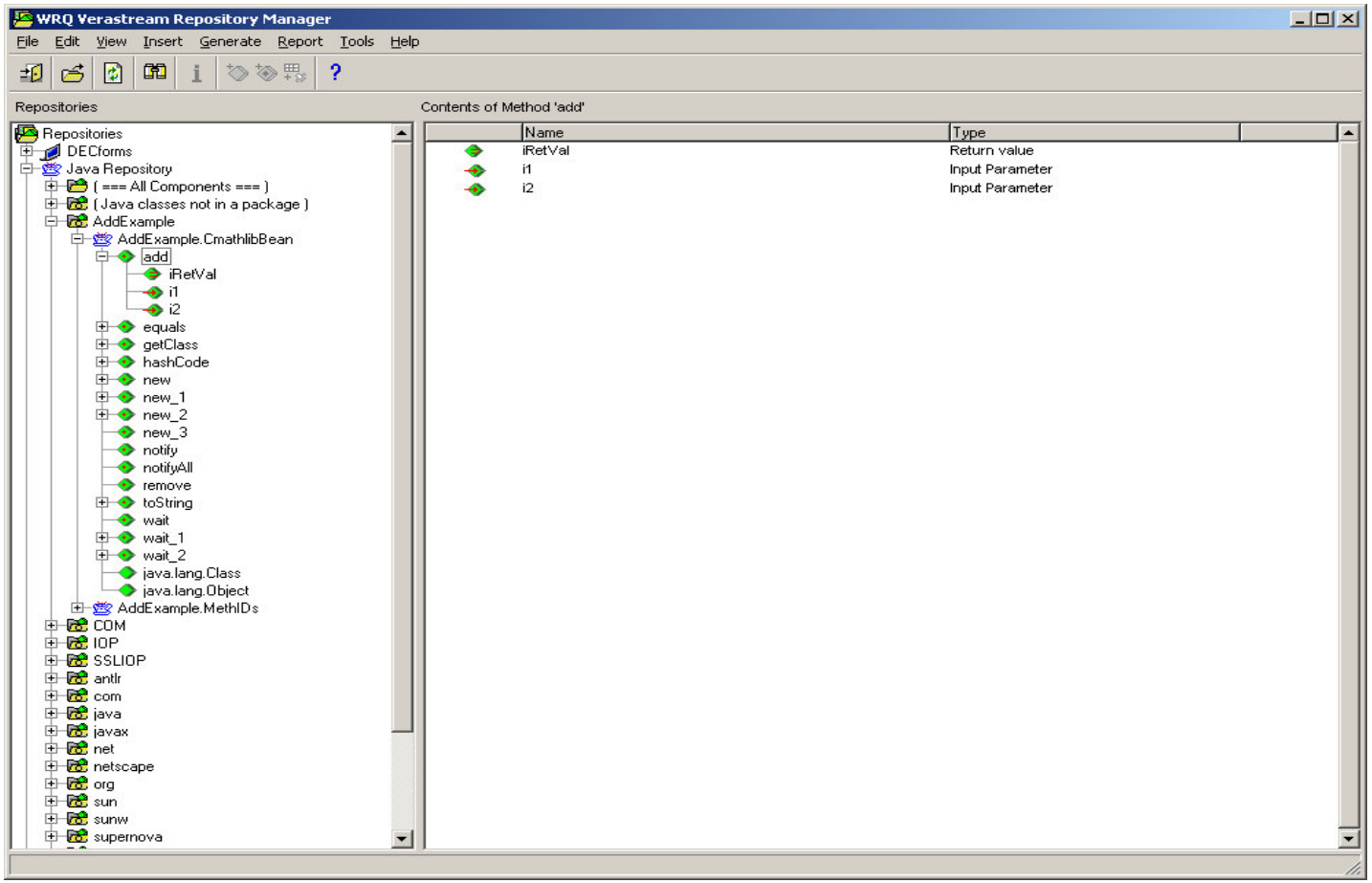


Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ❑ Open a Java Repository
- ❑ Create a Verastream Repository
- ❑ Define the Shipping Component
- ❑ Create “claimsdd” module file
- ❑ Realize(Implement) the Component
- ❑ Register(deploy) the Component
- ❑ Test the Component

Building the Shipping Component
Open a Java Repository

- Go to Verastream Repository Manager
- Expand the node Java Repository

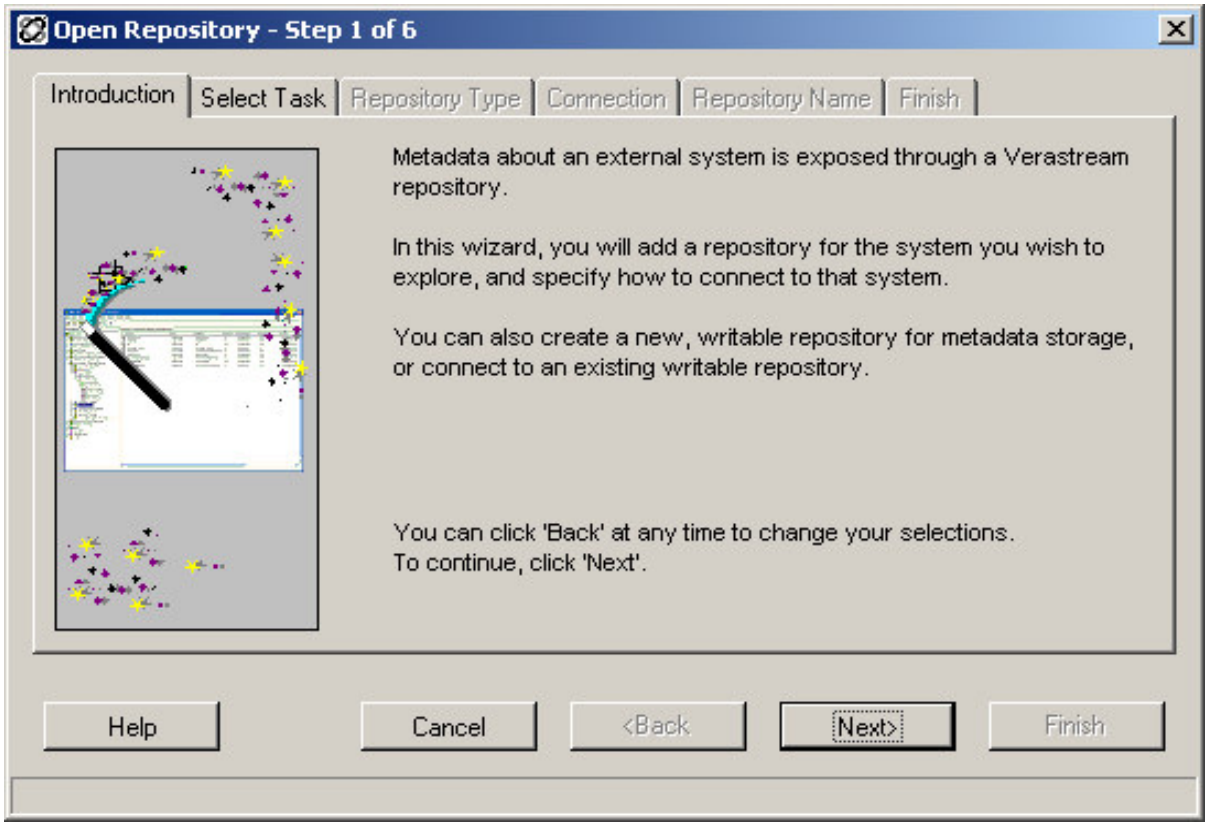


Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ☑ Open a Java Repository
- ❑ Create a Verastream Repository
- ❑ Define the Shipping Component
- ❑ Create “claimsdd” module file
- ❑ Realize(Implement) the Component
- ❑ Register(deploy) the Component
- ❑ Test the Component

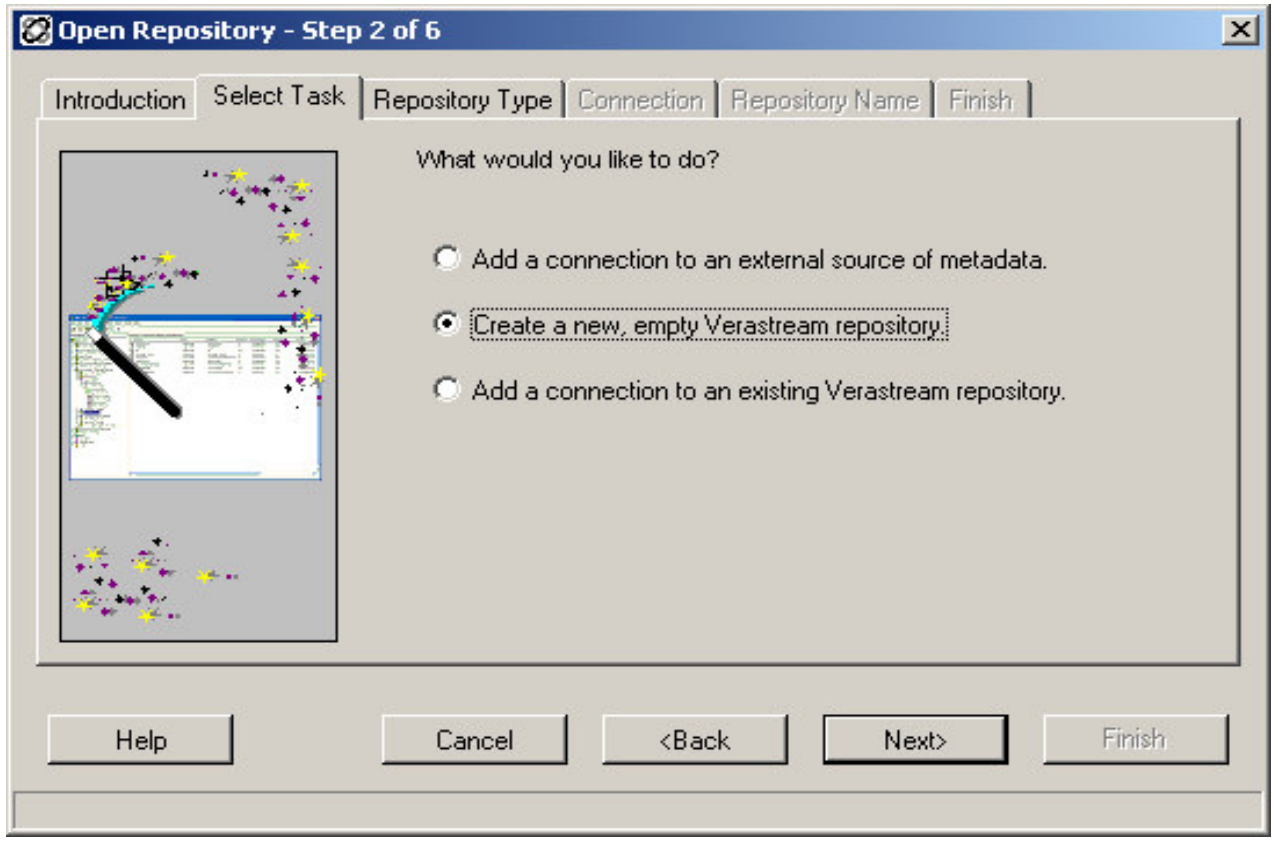
Building the Shipping Component
Create a Verastream Repository

- Go to the Repository Manager
- Select the menu File → Open Repository



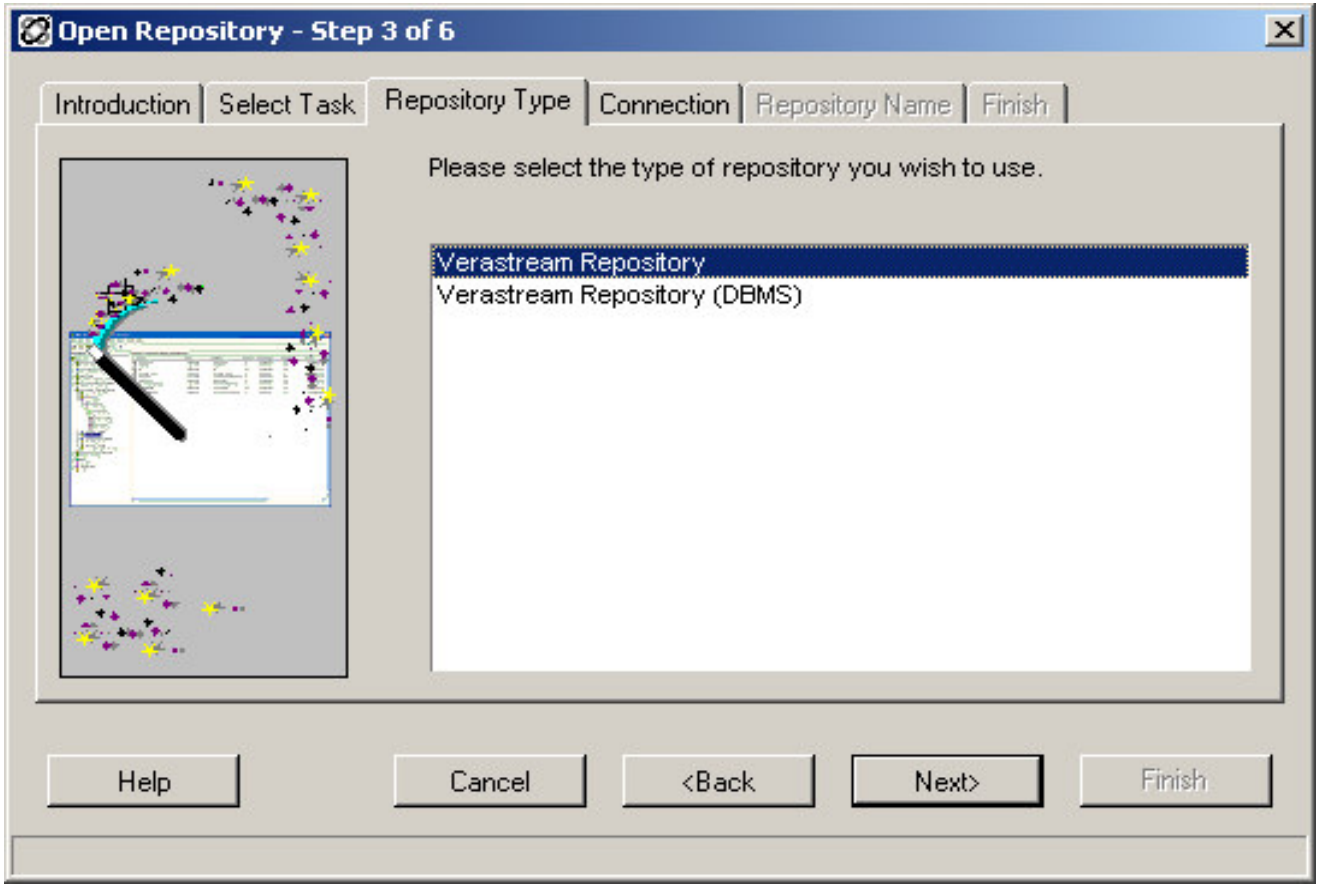
Building the Shipping Component
Create a Verastream Repository

- Create a new, empty Verastream Repository



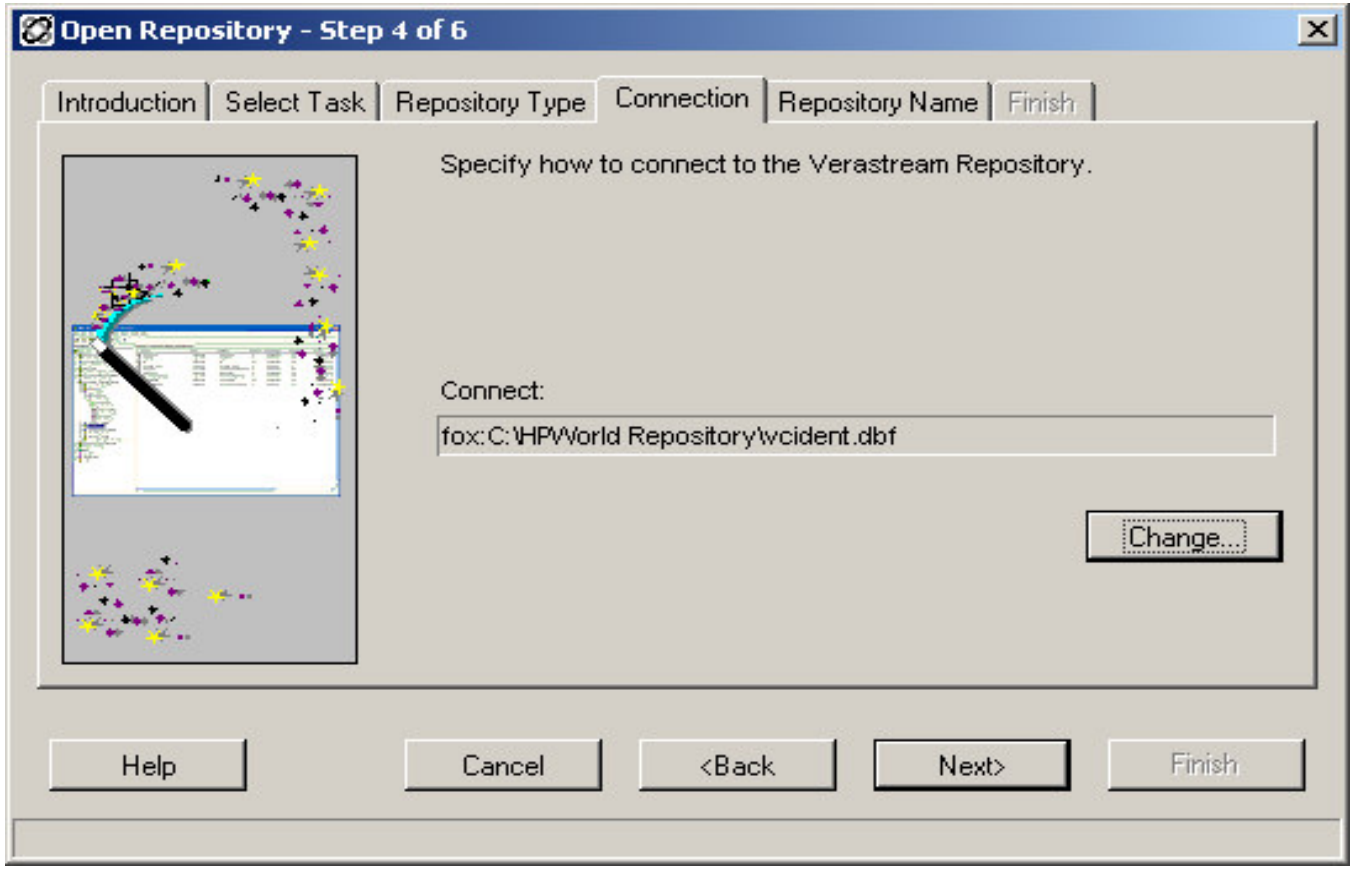
Building the Shipping Component
Create a Verastream Repository

- Select Verastream Repository



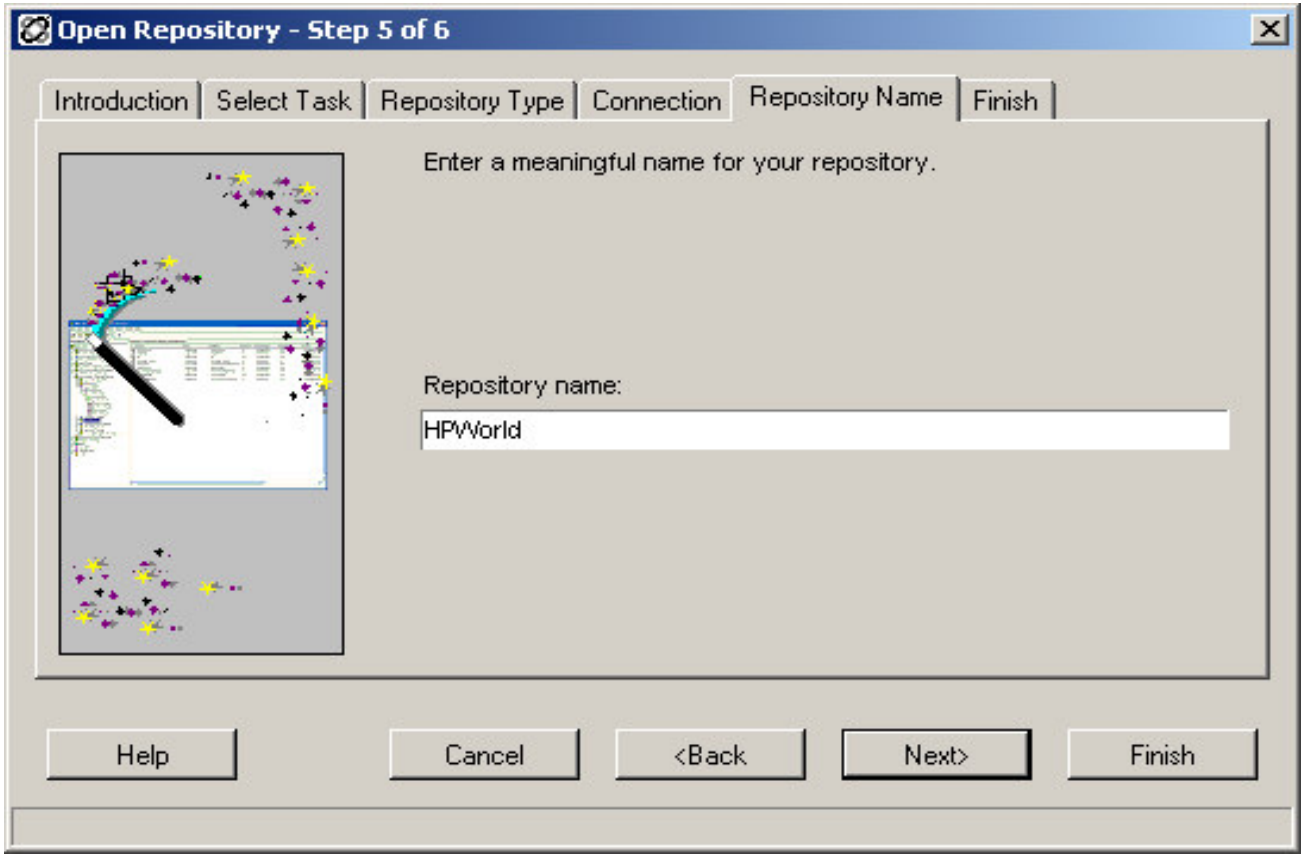
Building the Shipping Component
Create a Verastream Repository

- Specify a directory



Building the Shipping Component
Create a Verastream Repository

- Name the Repository
- Click Finish



Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ☑ Open a Java Repository
- ☑ Create a Verastream Repository
- ❑ Define the Shipping Component
- ❑ Create “claimsdd” module file
- ❑ Realize(Implement) the Component
- ❑ Register(deploy) the Component
- ❑ Test the Component

Define the Shipping Component

- Go to Repository Manager
- Select the Verastream Repository(HPWorld)
- Right click and select New → Component
- Follow instructions in Section I of ShippingComponent_Instructions.pdf file

Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ☑ Open a Java Repository
- ☑ Create a Verastream Repository
- ☑ Define the Shipping Component
- ❑ Create “claimsdd” module file
- ❑ Realize(Implement) the Component
- ❑ Register(deploy) the Component
- ❑ Test the Component

Create “claimsdd” module file

- Follow instructions in Section II of ShippingComponent_Instructions.pdf file

Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ☑ Open a Java Repository
- ☑ Create a Verastream Repository
- ☑ Define the Shipping Component
- ☑ Create “claimsdd” module file
- ❑ Realize(Implement) the Component
- ❑ Register(deploy) the Component
- ❑ Test the Component

Realize the Shipping Component

- Follow instructions in Section III of ShippingComponent_Instructions.pdf file

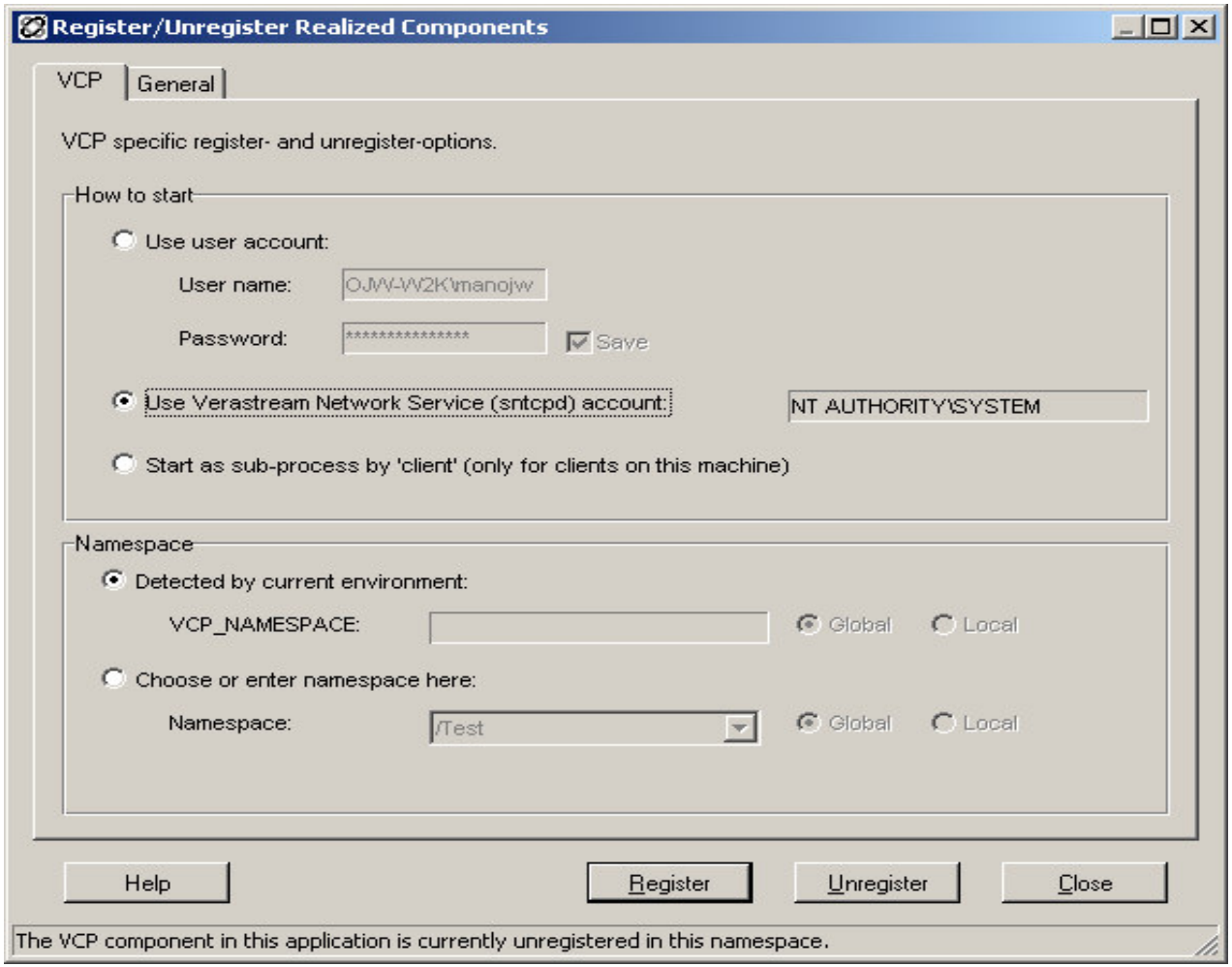
Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ☑ Open a Java Repository
- ☑ Create a Verastream Repository
- ☑ Define the Shipping Component
- ☑ Create “claimsdd” module file
- ☑ Realize(Implement) the Component
- ❑ Register(deploy) the Component
- ❑ Test the Component

Deploy the Shipping Component

- Launch Process Integrator and open the shipping.lgc file
- Select the menu Register → Run → Register/Unregister Realized Components …
- In the Register/Unregister Realized Components window, select the radiobuttons “Use Verastream Network Service(sntcpd) account and “Detected by current environment”
- Click the “Register” button

Deploy the Shipping Component



Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ☑ Open a Java Repository
- ☑ Create a Verastream Repository
- ☑ Define the Shipping Component
- ☑ Create “claimsdd” module file
- ☑ Realize(Implement) the Component
- ☑ Register(deploy) the Component
- ❑ Test the Component

Test the Component

- Follow instructions in Section IV of ShippingComponent_Instructions.pdf file

Build the components
Verastream Shipping Component – Composite Application

- ☑ Open a Table Repository(To import metadata of RMS data files)
- ☑ Open a Verastream Host Integrator Model Repository
- ☑ Open a Java Repository
- ☑ Create a Verastream Repository
- ☑ Define the Shipping Component
- ☑ Create “claimsdd” module file
- ☑ Realize(Implement) the Component
- ☑ Register(deploy) the Component
- ☑ Test the Component

“ We had numerous and complex demands, ranging from very high performance requirements to the fundamental need to retain our OpenVMS-based systems. WRQ was the only vendor with products and services that met or exceeded these demands, WRQ provides us with significant value, not only enabling us to modernize our infrastructure and applications but also to maximize the value of our OpenVMS systems.”

Dave Robinson

Vice President, MIS, Southeastern Freight Lines

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP