WebSphere Message Broker
Version 8 Release 0

Scenario: Using a message map to enrich a message with data from a database

Note
Before using this information and the product it supports, read the information in "Notices" on page 43.

Published date: 02 October 2013

Contents

Figures

Chapter 1. Introduction to the "Using a message map to enrich a message with data from a database" scenario

This scenario shows how you can enhance a message in WebSphere® Message Broker by using a message map. The data is available in a database system. The data is stored across multiple database tables. All the tables are located within the same database schema.

About this task

WebSphere Message Broker Version 8.0 introduces graphical data maps. These message maps replace the previous message map format, and are created as `.map` files. Message maps offer the ability to transform a message without the need to write code, providing a visual image of the transformation, and simplifying its implementation and ongoing maintenance.

WebSphere Message Broker Version 8.0 also introduces the **Select** transform that allows you to enrich a message by accessing data located in an external database system. This new feature simplifies the programming model. It eliminates the requirement to use a Database node, a JavaCompute node, a .NETCompute node, or a Compute node to access data located in a database. You can design simpler message flows by using a single Mapping node to complete graphically a message transformation that requires data from an external database system.

Note: If you migrate from an earlier version of WebSphere Message Broker Version 8.0, you can continue using your legacy maps. However, if you need to modify any of your legacy maps, or if you want to use the **Select** transform, you must convert these legacy message maps into `.map` message maps. For more information about converting maps, see Converting a message map from a .msgmap file to a .map file.

Read the following topics to understand the scenario and the concepts the scenario is intended to demonstrate:

- "Context"
- "Technical solution" on page 3

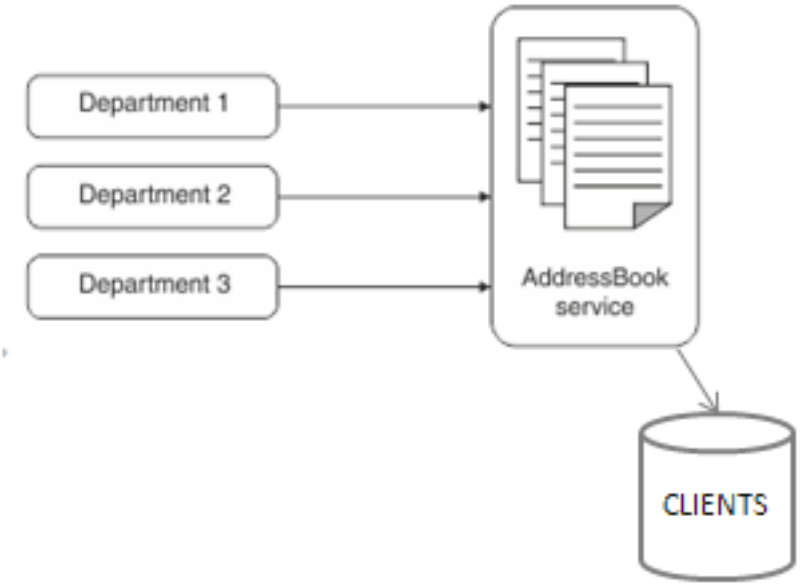
What to do next

Implement the solution. For more information, see Chapter 2, "Implementing the solution," on page 5.

Context

This scenario shows how you can enhance a message in WebSphere Message Broker by using a message map. In this scenario, the data is available in an external database system. The data is stored across multiple database tables, all of which are located within the same database schema.

Your company has implemented an AddressBook service that is used by different departments in different countries in your organization. This service allows your employees to obtain a client's mail address or to save a new client's mail address.



The company uses WebSphere Message Broker to develop and manage a number of integration solutions that transform and communicate data between source and target systems. In order to make the service reusable by multiple applications, you design an application responsible for the transformation of the different address formats between the requesting application and the AddressBook service. The AddressBook service is a SOAP-based service that stores a new address or returns an address to the user. You use a message map to define how to transform the SOAP message based on the operation that your user requests.

The company uses DB2 Version 9.7 as the external database system that hosts client's details and addresses.

The scenario uses the following database tables:

- *Person*: This table contains an entry per client. The client ID element is used to link information for this client across all tables in the database. The database automatically assigns the ID value when a new record is created. This table contains all clients from all countries.
- *Address*: This table contains an entry per client with the address details.
- *Phone*: This table contains an entry per client with the phone details.

In WebSphere Message Broker, you have the following choices to implement a message flow that connects to a database, and retrieves information to enrich the message:

- You can use a Mapping node to graphically connect to a database and retrieve data to use in the node and later on in the message flow.
- You can use a Database node in a message flow to connect to a database and retrieve data that you can use later on in the message flow.
- You can program a Compute node, JavaCompute node, or .NETCompute node to connect to a database and retrieve data to use in the message flow.

This scenario demonstrates how to use a Mapping node to connect to a database, retrieve data from multiple tables, and graphically populate elements in a SOAP message with this information in the WebSphere Message Broker Toolkit.

Technical solution

You can use a message map to enhance an existing message with data from one or more database tables. Data from the database can then be used to enrich, route, and transform messages within WebSphere Message Broker.

In WebSphere Message Broker, to connect to a database, you must configure the development environment and the WebSphere Message Broker runtime environment:

1. To have visibility of the database resources during the development phase, you must connect the WebSphere Message Broker Toolkit to the development database.
2. To enable the deployed map to execute in the run time, you must create a JDBC provider configurable service that defines the connection to the runtime database. This database is normally a different database server from the one you use for development, and the artifacts could be in a different database schema.

To configure the WebSphere Message Broker Toolkit to connect to a database, you must create a database definition file in a data design project, and configure a database connection.

- Data design project: A specialized type of project where you store your database resources.
- Database definition file: A configuration file where you specify the database physical details such as database type and version, and a connection.
- Database connection: Configuration that details the database resources, that is, the schema, the tables, the store procedures, the indexes, and other resources, that you need access to from within your WebSphere Message Broker project resources.

To access information stored in a database from resources in a WebSphere Message Broker project, you must include a reference to the data design project in your application, service, or Message Broker project.

In WebSphere Message Broker, you can use a message map to access information in a database, and then use this information to perform transformations on the message or to enrich a message.

During the design phase, you must complete the following steps in the WebSphere Message Broker Toolkit to access graphically database information in a message map:

1. Add a reference to each database table from where you must retrieve data.
2. Use a **Select** transform to define how to use the database information in the message map. The **Select** transform has embedded a nested map. You must define the transforms in this nested map.
3. Use a **Failure** transform to handle database failures. The **Failure** transform has embedded a nested map. You can define the transforms in this nested map if you wish to provide specialized handling of any database exceptions that are hit running the generated SQL statements when the map executes. If you take

the default of not adding a **Failure** transform, WebSphere Message Broker will handle the error, reporting it to the system log, and then rolling back the current message transaction.

To configure the WebSphere Message Broker run time to connect to a database, you must establish a connection with the database to fulfill the operations that are performed by the Mapping node. You must define a JDBC provider configurable service.

Use this scenario to learn how to use a Mapping node to connect to a database, retrieve data from multiple tables, populate elements in a SOAP message with this information, and handle a database SQL exception. use this scenario to also learn how to configure the JDBC provider configurable service.

Chapter 2. Implementing the solution

During the development phase, you can use a Mapping node to connect to a database, retrieve data from multiple tables, and then populate elements of a SOAP message. During the design of the map, you can connect the WebSphere Message Broker Toolkit to a database to discover the tables. To enable the run time to establish a connection with the database to fulfill the operations that are performed by the Mapping node, you must create a JDBCProvider configurable service.

Before you begin

1. Create the initial configuration. For more information, see "Creating the scenario graphical data map configuration."
2. Create the database resources. For more information, see "Creating the scenario database configuration" on page 8.

Procedure

Complete the following steps to implement a Mapping node that connects to a database and enriches a message with the database information:

1. Configure the database physical model in WebSphere Message Broker by running discovery. For more information, see "Configuring a database in the WebSphere Message Broker Toolkit" on page 9.
2. Configure your integration solution to include the database connection details. For more information, see "Configuring an integration solution to access database resources" on page 21.
3. Add the relevant database tables to your message map. For more information, see "Adding database tables your message map" on page 23.
4. Configure the **Select** transform in your message map to retrieve the database information. For more information, see "Configuring the Select transform in a message map" on page 29.
5. Optional: Handle database failures. For more information, see "Handling database failures in a Select transform" on page 34.
6. Configure the run time to enable it to establish a connection with the database to fulfill the operations that are performed by the Mapping node. For more information, see "Configuring a database to be available at run time" on page 39.

Creating the scenario graphical data map configuration

This scenario was developed by using a sample initial configuration. You can either follow the instructions to add your own database to a message map, or set up the sample final configuration to try out the scenario in the same way as it was originally developed.

Before you begin

- Download a copy of the `FindAddressInitialConfiguration.zip` file.
- Download a copy of the `FindAddressFinalConfiguration.zip` file to set up the final scenario configuration and see the result of following the steps that are documented in the scenario.

- Make sure you have access to a WebSphere Message Broker runtime environment and a WebSphere Message Broker Toolkit installation with the default configuration deployed. For more information on installing WebSphere Message Broker components, see Installing in theWebSphere Message Broker information center.

Procedure

Complete the following steps to set up the sample initial configuration that was used to develop the scenario:

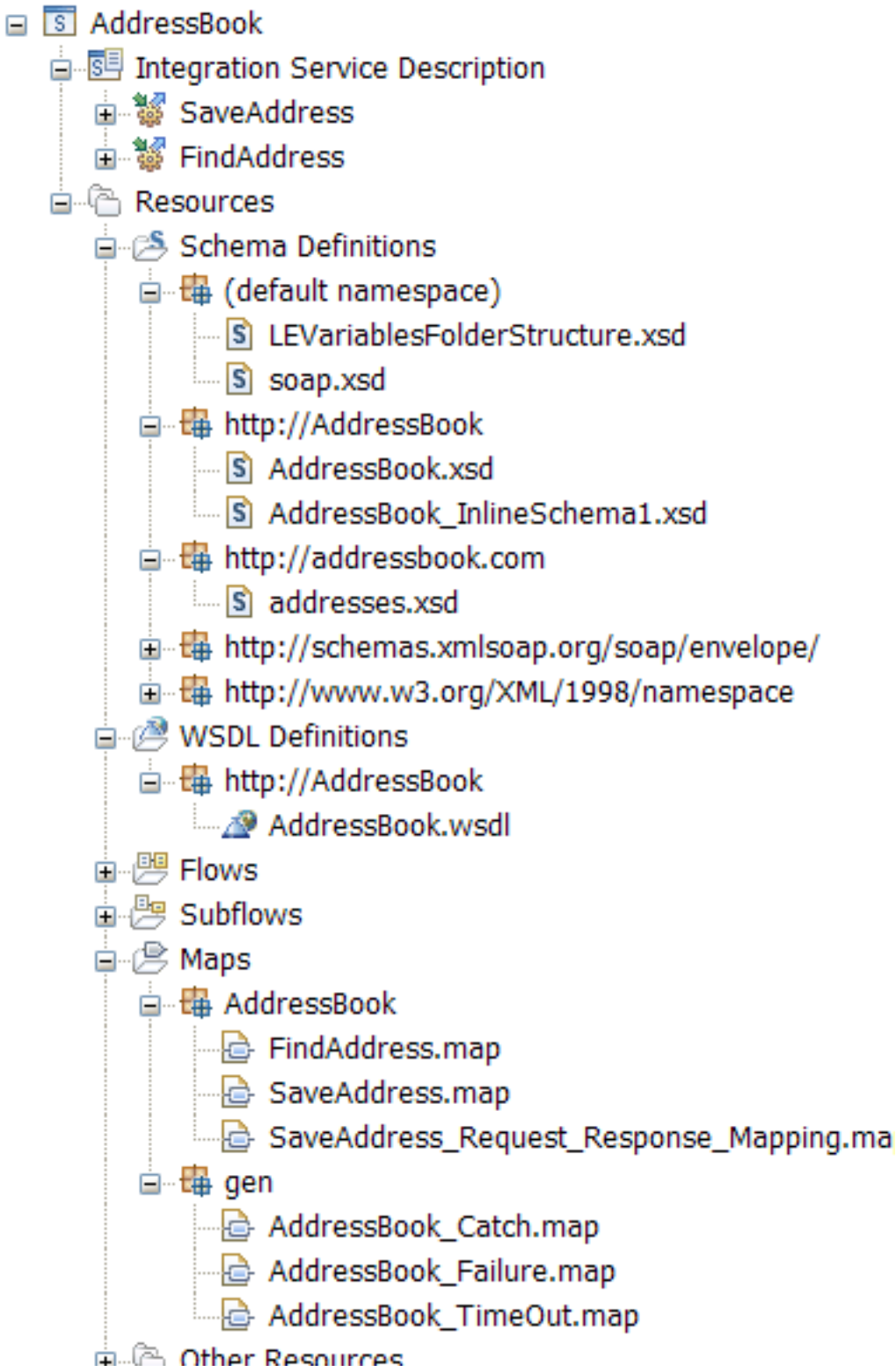
1. Install WebSphere Message Broker Toolkit. For more information, see Installing in theWebSphere Message Broker information center.
2. Import the `AddressBookInitialConfiguration.zip` file:
   a. Click **File** > **Import**. The Import wizard opens.
   b. Expand **Other**, click **Project Interchange**, then click **Next**.
   c. Specify the location of the `AddressBookInitialConfiguration.zip` file.
   d. Specify the location of the open workspace.
   e. Select the projects that you want to import into your workspace. For this scenario, select all projects. Then, click **Finish**.

Results

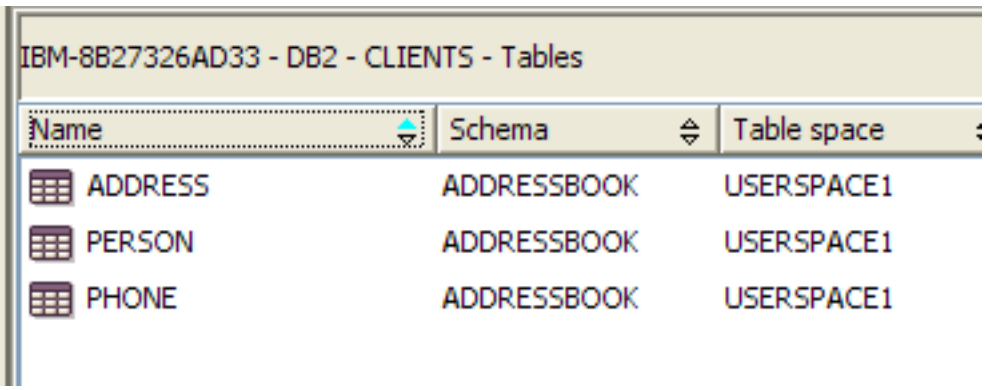
You imported the scenario source files.

In the **Application Development** view, you should see the following resources:

- A SaveAddress operation
- A FindAddress operation
- Multiple xsd structures such as addresses.xsd
- The AddressBook wsdl file
- The FindAddress.map file
- The SaveAddress.map file
- The SaveAddress_Request_Response_Mapping.map file
- The Catch.map file
- The Failure.map file
- The TimeOut.map file



In your database, you see the ADDRESS, PERSON, and PHONE tables:



What to do next

Follow the steps for "Configuring a database in the WebSphere Message Broker Toolkit" on page 9.

Creating the scenario database configuration

The message flow used in this scenario requires an external database, which is used in one of the message maps that enriches the message data as it runs the transformation. The database must be set up in advance.

Before you begin

- Download a copy of the `createdbtablesclients.zip` file.

Procedure

Complete the following steps to set up the sample DB2 database configuration that was used to develop the scenario:

1. Create a database named **CLIENTS**.
   a. Open a DB2 command prompt and create the database. Click **Start** > **All Programs** > **IBM DB2** > ***DatabaseInstance*** > **Command Line Tools**, and select Command Window. *DatabaseInstance* is your DB2 instance name, which by default is the following: **DB2COPY1 (default)**.

      A DB" - CLP window opens.

   b. Create the **CLIENTS** database. Run the following command: `DB2 CREATE DB CLIENTS`

      You receive the following message: `DB20000I The CREATE DATABASE command completed successfully.`

   c. Test the database connection. Run the following command: `DB2 CONNECT TO CLIENTS`
2. Create the tables using the SQL **createdbtablesclients.sql** script provided in the scenario.
   a. Unzip the file **createdbtablesclients.zip**.
   b. From the DB2 command prompt, run the following command: `db2 -vf` *`Sqlscriptdirectory`*`\createdbtablesclients.sql`, where *Sqlscriptdirectory*\ is the directory where you unzip **createdbtablesclients.zip**.

Results

You have a database named CLIENTS, and the following database tables created under the **ADDRESSBOOK** schema:

- PERSON
- ADDRESS
- PHONE

What to do next

Follow the steps for "Configuring a database in the WebSphere Message Broker Toolkit."

Configuring a database in the WebSphere Message Broker Toolkit

To make database information accessible from a Mapping node during the development phase, you must define a database definition file with extension `.dbm`. This file is contained in a data design project. You must define one database definition file per database.

Before you begin

To start the scenario, create the initial configuration. For more information, see "Creating the scenario graphical data map configuration" on page 5.

About this task

You can use WebSphere Message Broker to access a database and manipulate your business data.

During the development phase, you must configure the database before you can access the data from your a message flow in the WebSphere Message Broker Toolkit. WebSphere Message Broker supports the databases that are listed in Supported databases.

WebSphere Message Broker can access databases that are set up on the local computer or on a remote server, subject to restrictions. For more information, see Database locations.

This scenario demonstrates how to configure a local database in the WebSphere Message Broker Toolkit.

Procedure

To configure the **CLIENTS** database, complete the following steps:

1. Create a data design project as a container for a database definition file. See "Creating a data design project."
2. Define your database and how it interacts with your integration node. See "Creating the database definition file" on page 11.

Creating a data design project

Create a data design project to contain the database definition file that describes your database.

About this task

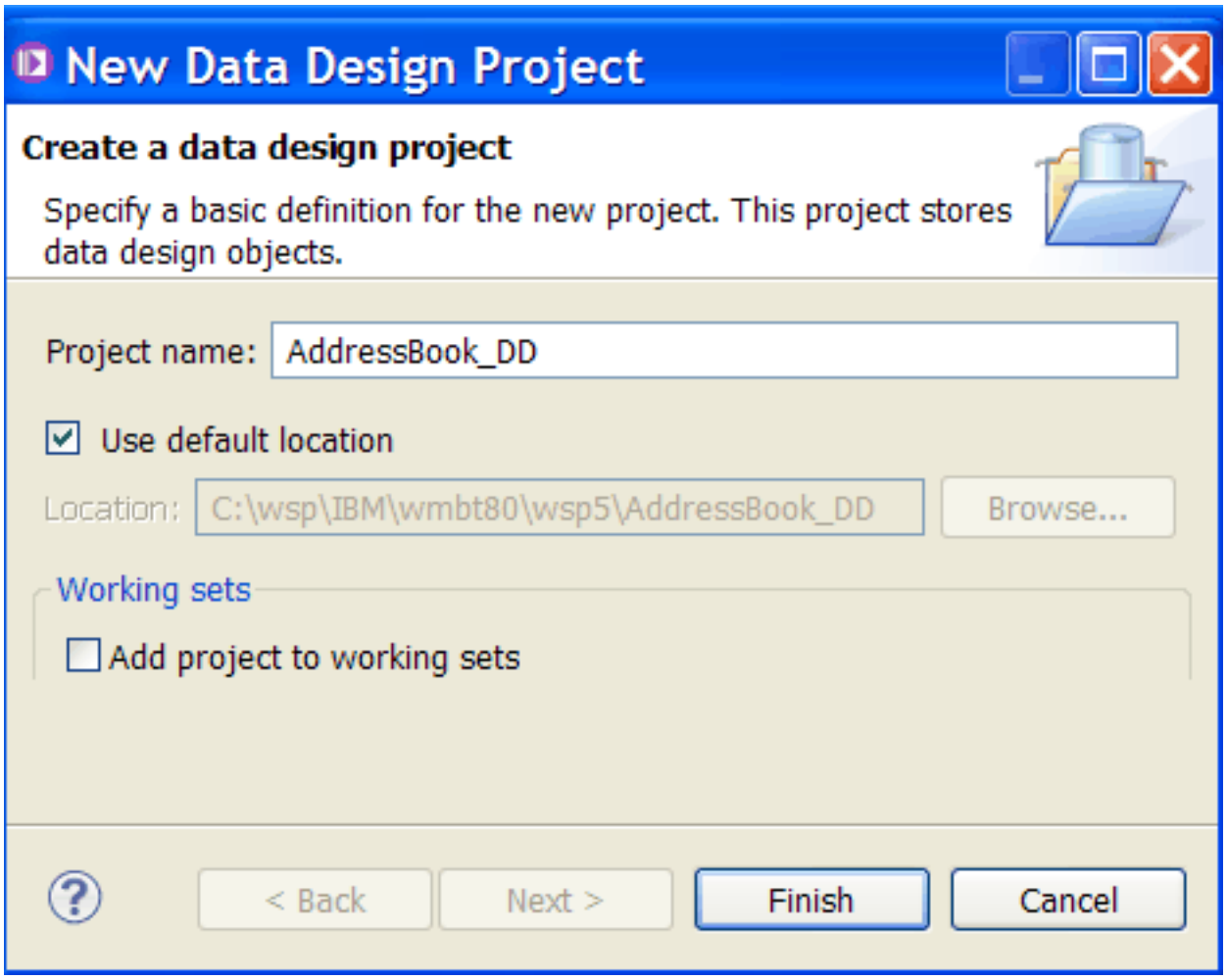
Data design projects are used in the WebSphere Message Broker Toolkit to contain a definition of your database, such as its structure, available fields, and data types, so that the data can be interpreted correctly by the WebSphere Message Broker runtime.

In this scenario, you create the **AddressBook_DD** data design project.

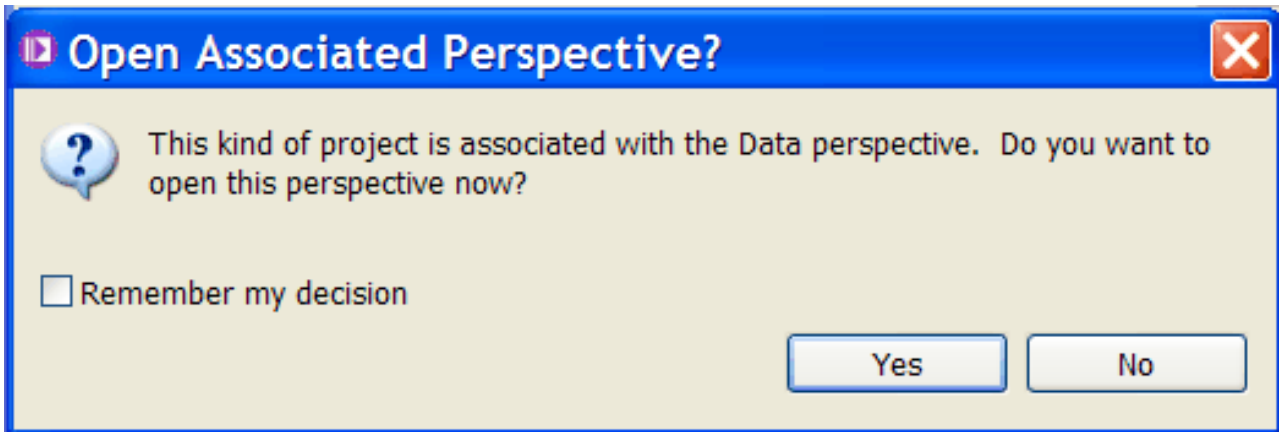
Procedure

Create the **AddressBook_DD** data design project in the Application Development view:

1. Click **File** > **New** > **Other** > **Data Design Project**. The New Data Design Project wizard opens. If this is the first database design project you create within a new workspace you might see the Confirm Enablement window first.
2. Enter **AddressBook_DD** as your *Project name*, and then click **Finish**.



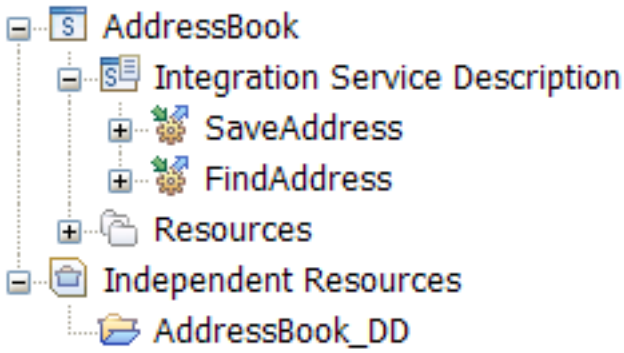
The Open associated perspective? window is displayed.



3. Click **No**.

Results

Your data design project is created, and is displayed in the Application Development view, under **Independent resources**.



What to do next

After you create a data design project, you must add a database definition (`.dbm`) file for your database. For more information, see "Creating the database definition file."

Creating the database definition file

To create database mappings by using the Mapping node, you must have a database definition file (`.dbm`) that is contained in a data design project.

Before you begin

Create the **AddressBook_DD** data design project. For more information, see "Creating a data design project" on page 9.

About this task

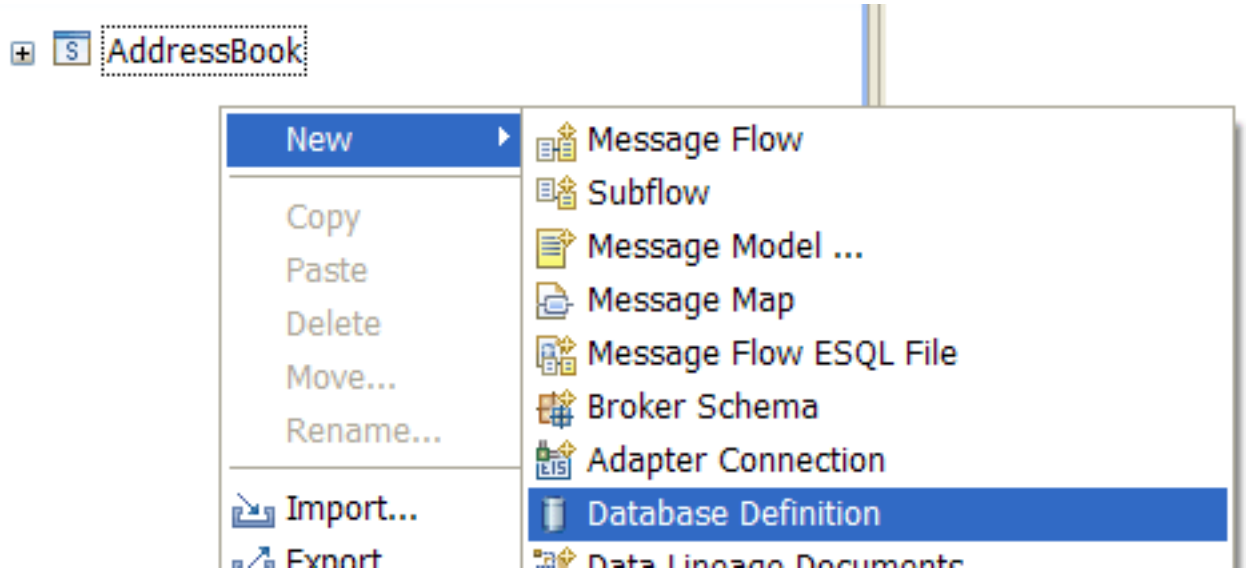
A database definition file contains information about a connection to a database.

Note: Database definition files in the WebSphere Message Broker Toolkit are not automatically updated. If you modify your database, you must recreate the database definition file describing the connection to the database.

Procedure

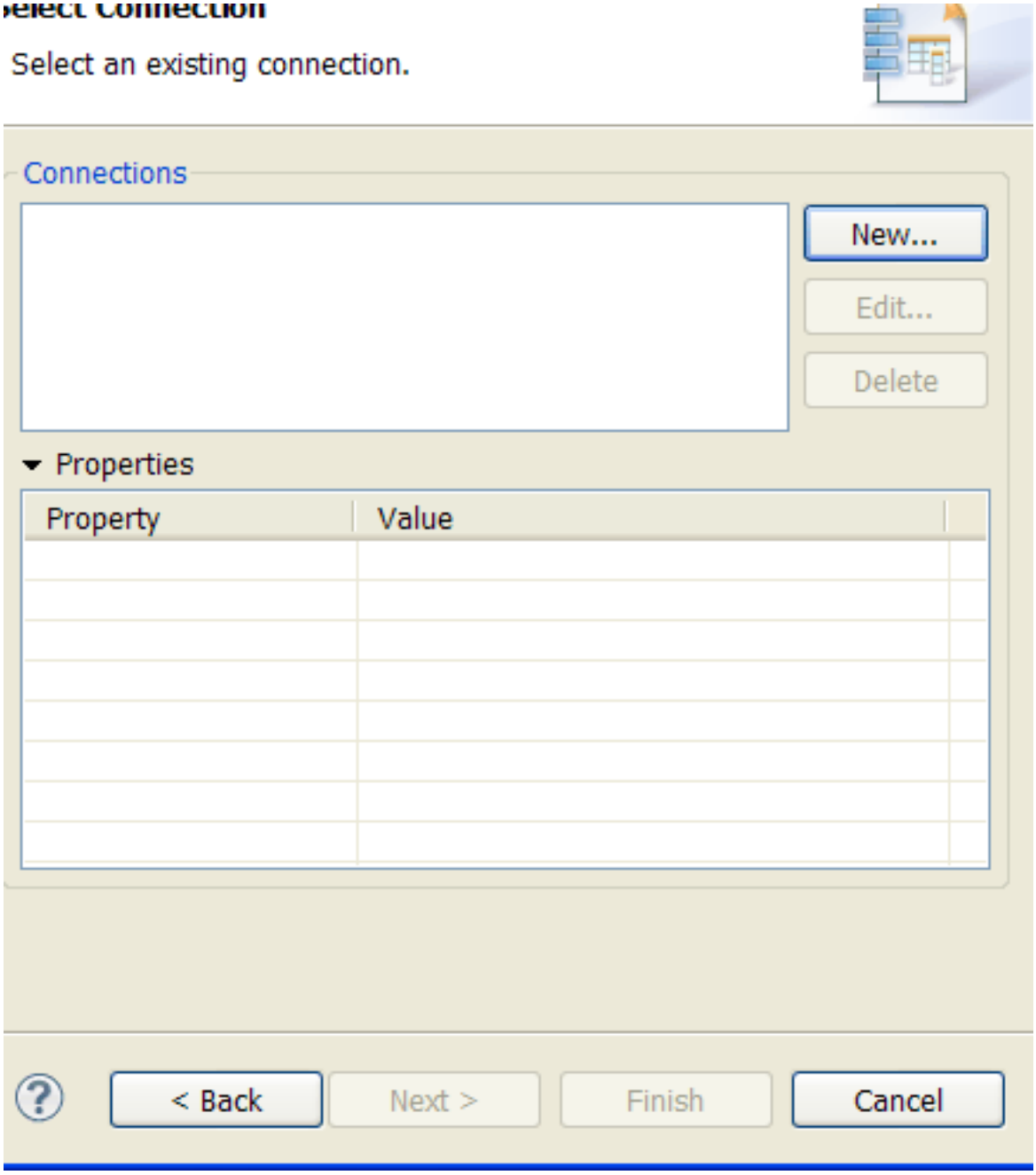
Complete the following steps to create the **CLIENTS.dbm** database definition file by using the New Database Definition File wizard:

1. In the Application Development view, right-click and select **New** > **Database Definition**.



   The New Database Definition File wizard is displayed.

2. From the *Data design project* drop-down list, select **AddressBook_DD**. From the **Database** drop-down list, select **DB2 for Linux, Unix, and Windows**. From the *Version* drop-down list, select **V9.7**. Then, click **Next**.
3. Create a database connection. In the New Database Definition File - Select connection, select **New**.

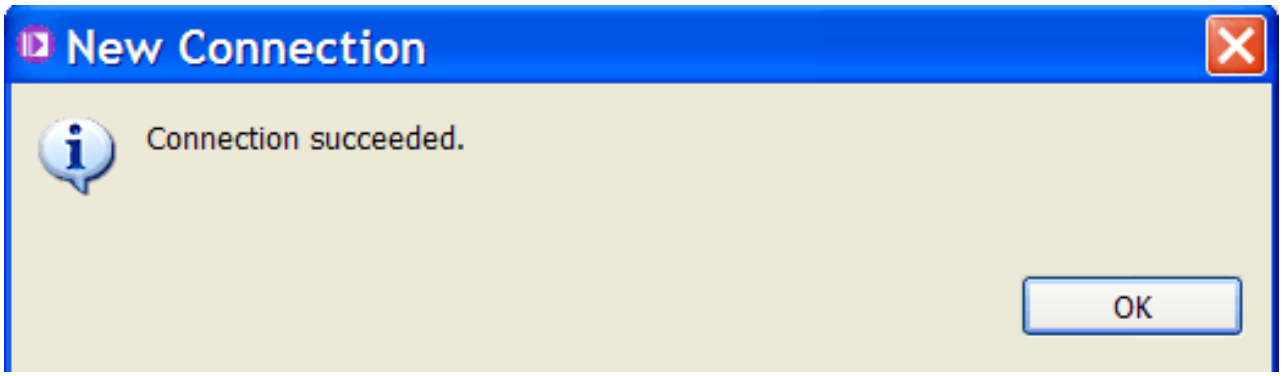


A New Connection window opens.

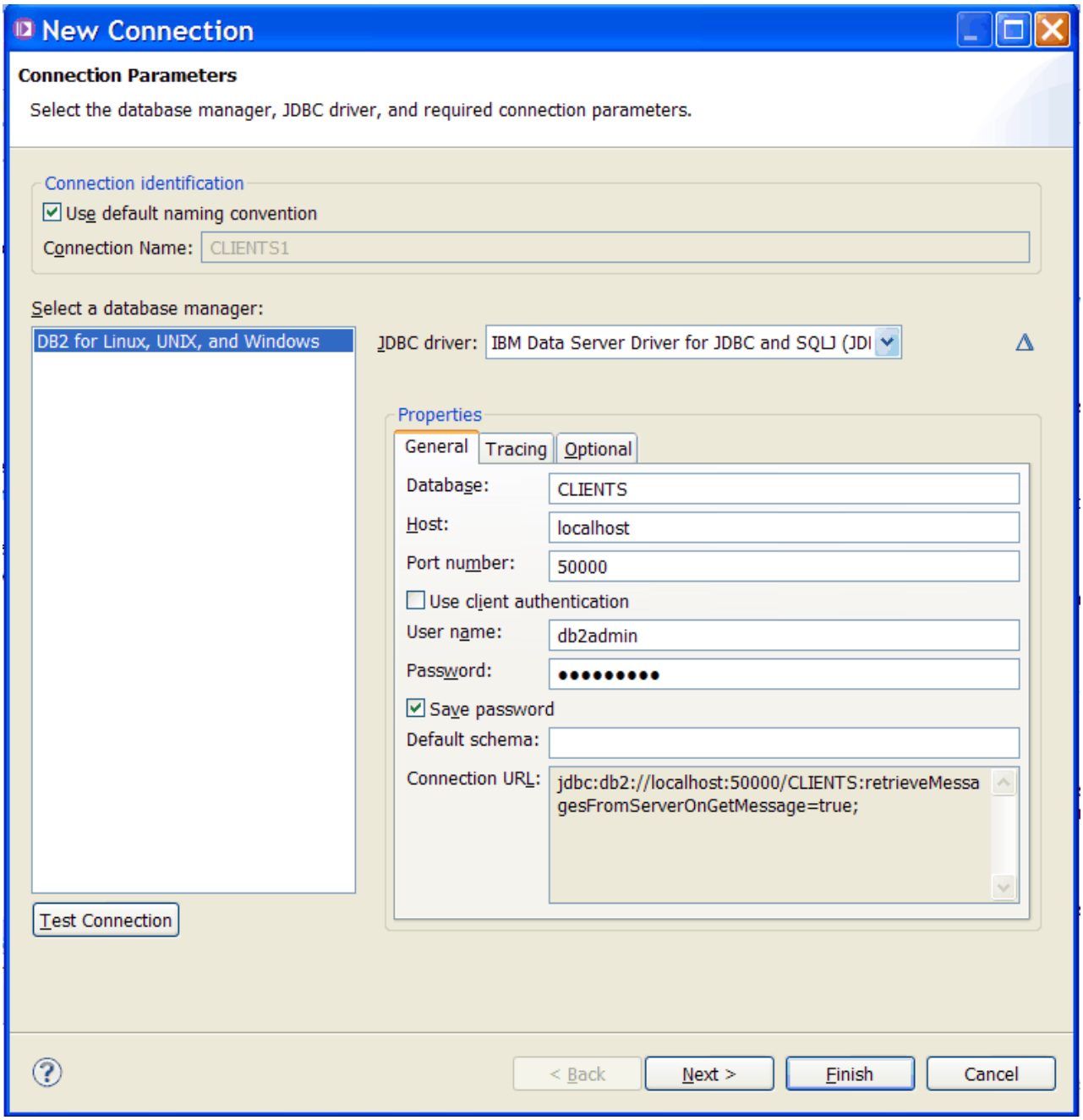
4. In the New Connection - Connection Parameters, edit the following properties to configure the **CLIENTS** database connection:
   a. Enter **CLIENTS** as the *Database* value.
   b. Enter **db2admin** as the *User name*, and enter your database administrator password in *Password*.

      Note: You can use a different user name to connect to the database. This user must have database administration permissions.
   c. Click **Test Connection** to verify the settings that you selected for your database.

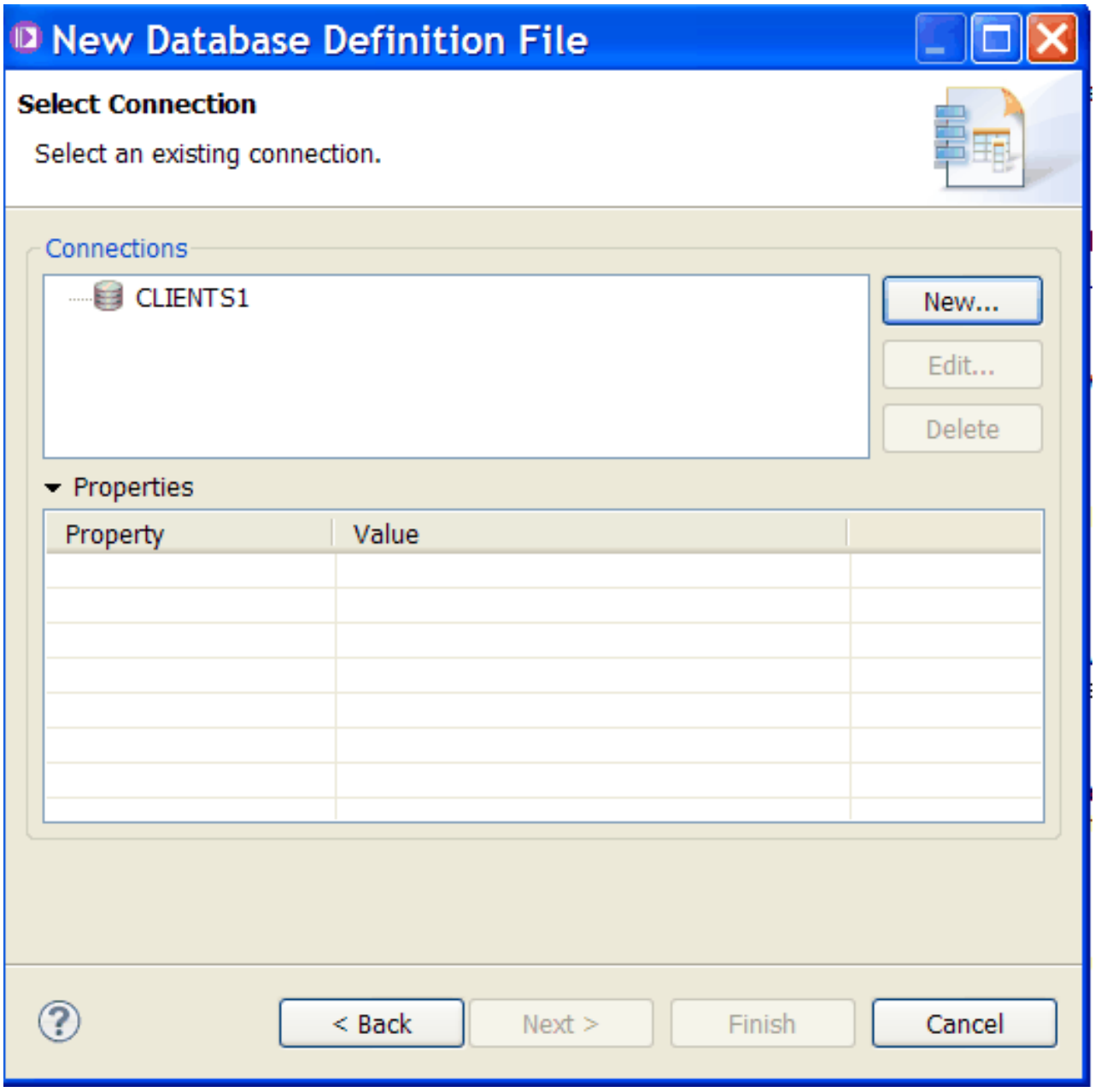
      If the connection is successful, the following window opens:



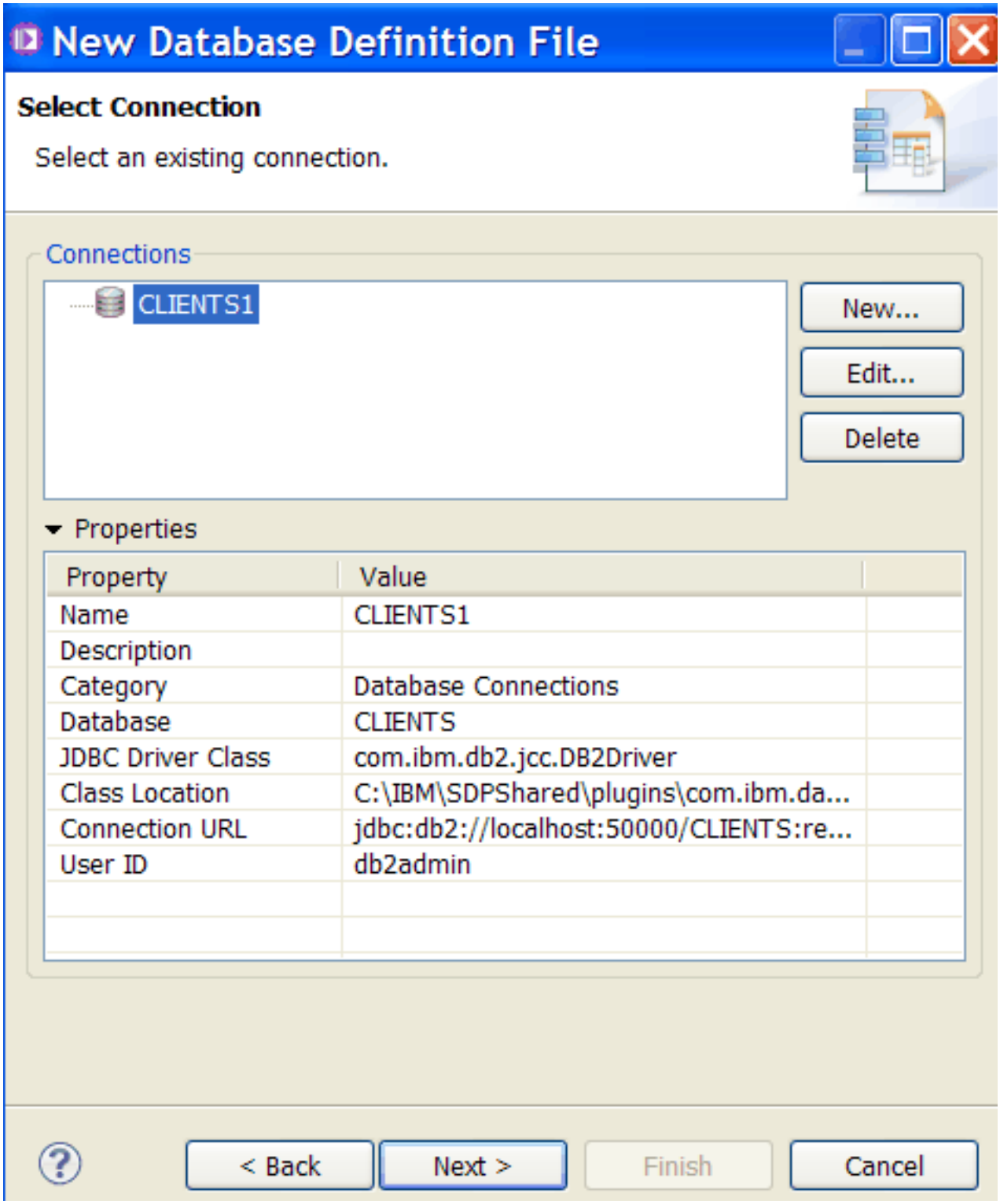
d. Select *Save Password*.



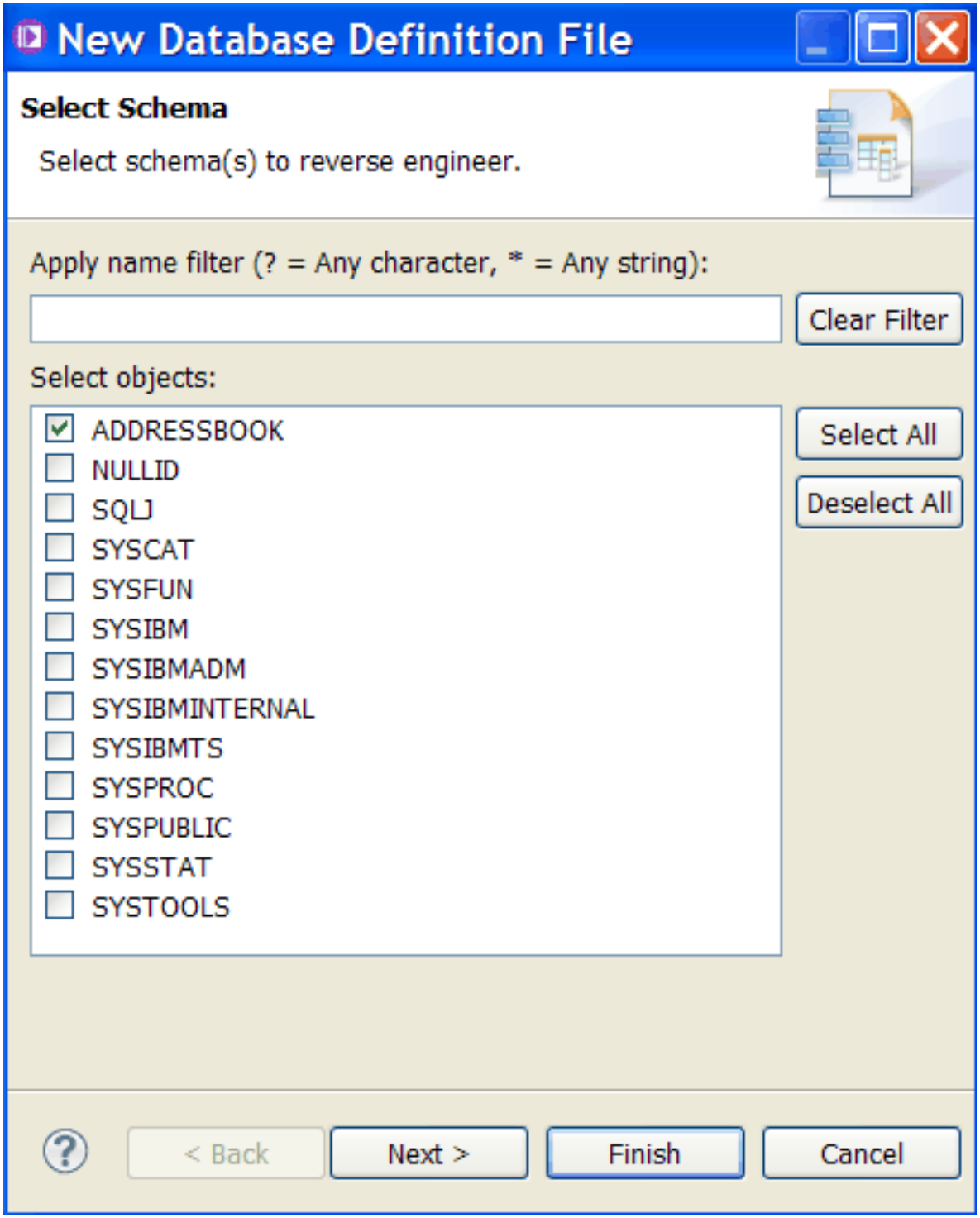
5. In the New Connection - Connection Parameters, click **Finish**.
   The **CLIENTS1** connection is created.



6. Select the **CLIENTS1** connection, and then select **Next**.



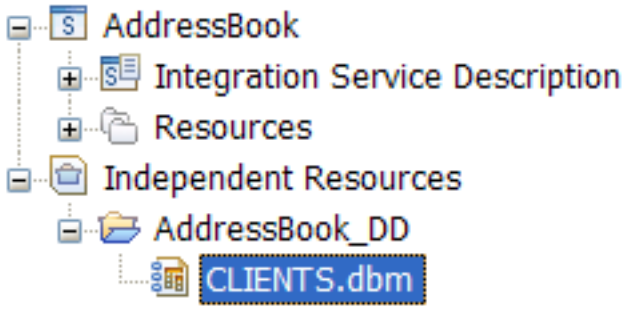
7. Select the schema **ADDRESSBOOK**, and then click **Next**.



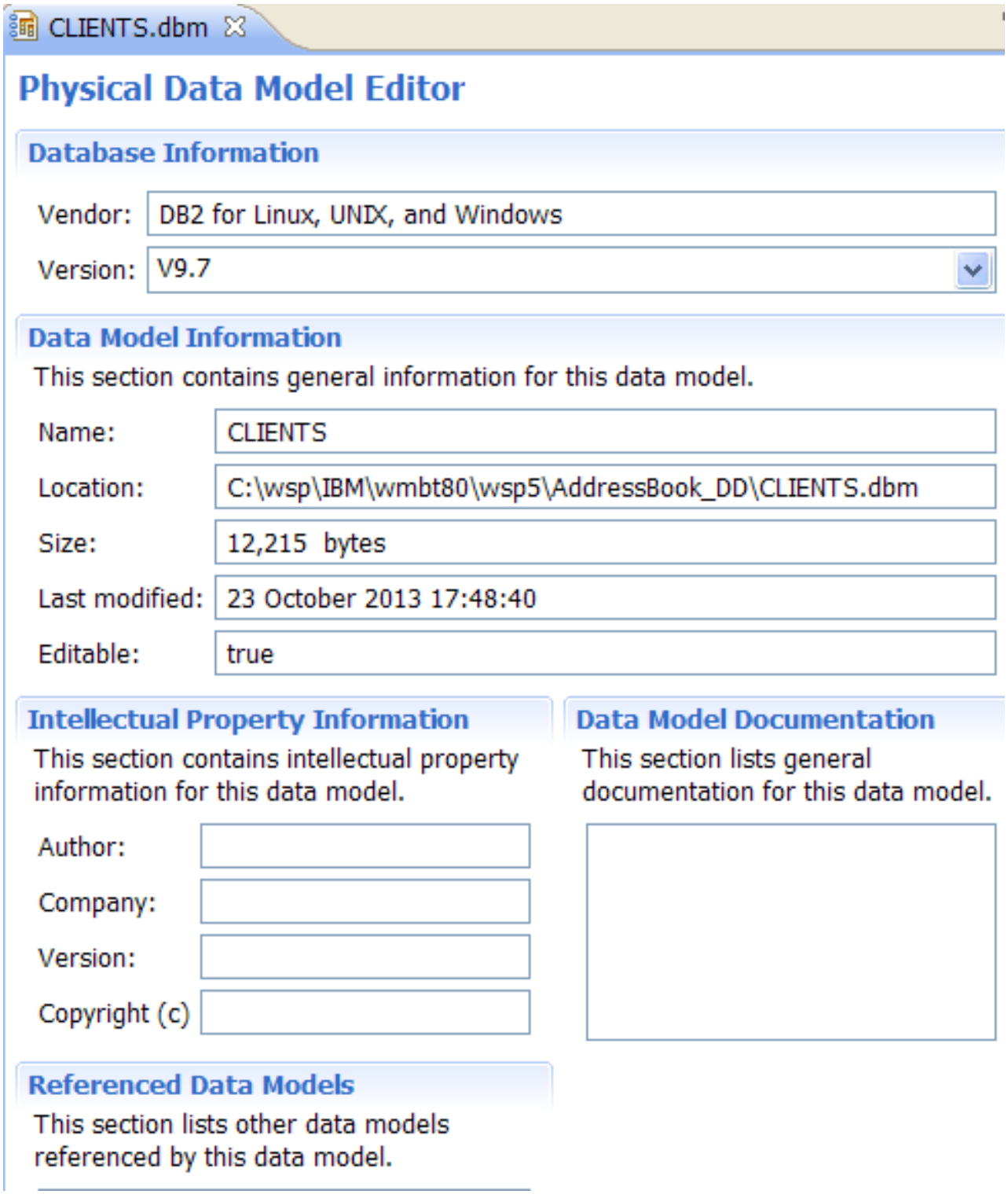
8. In the New Database Definition File - Database Elements window, select **Tables**, then click **Finish**.

Results

The CLIENTS.dbm database definition file is available in the Application Development view under **Independent resources**. The database definition file is created inside the **AddressBook_DD** data project.



The following figure shows the **CLIENTS** database definition file opened in the Physical Data Model Editor.



What to do next

After you create the database definition file **CLIENTS.dbm**, you must configure your integration solution to access specific database resources. For more information, see "Configuring an integration solution to access database resources" on page 21.

Configuring an integration solution to access database resources

To access database tables, you must associate the data design project where your database definition file is located with your integration solution.

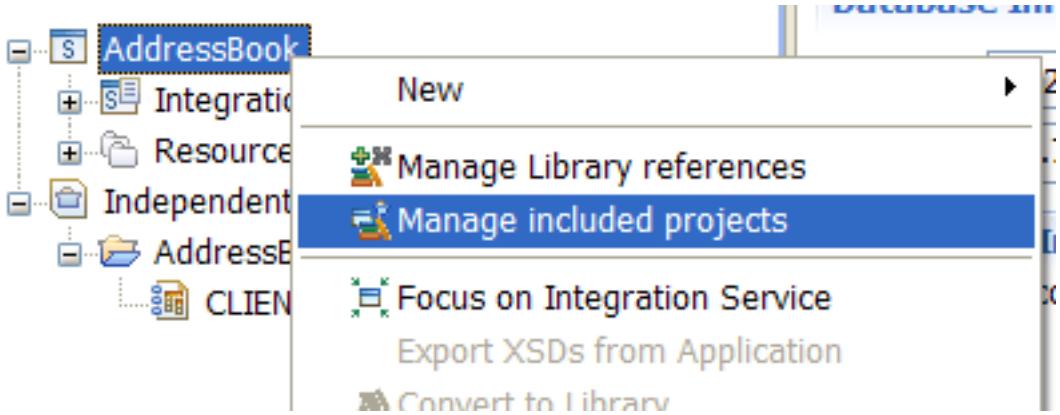
Before you begin

Create the **CLIENTS.dbm** database definition file. For more information, see "Creating the database definition file" on page 11.

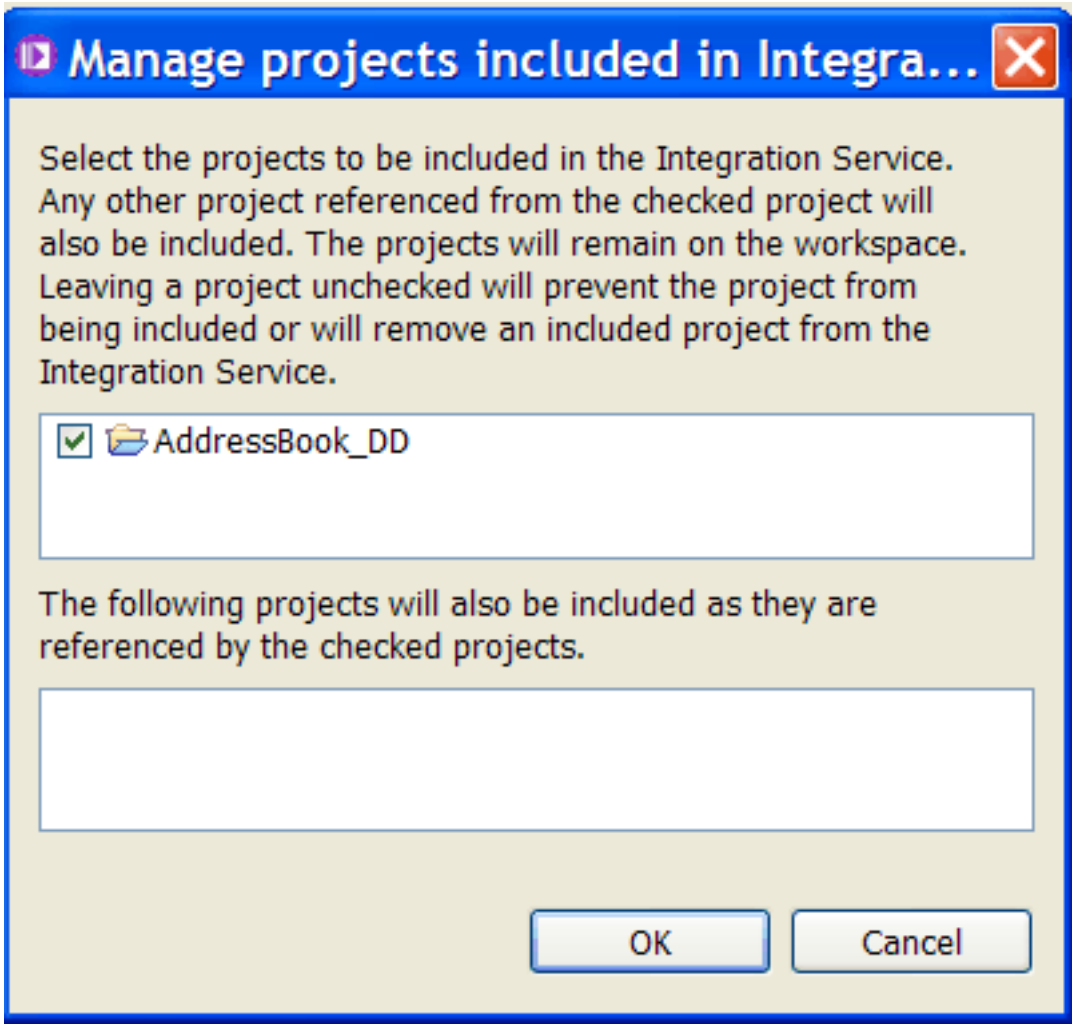
Procedure

To associate the **CLIENTS.dbm** database definition file to the **AddressBook** integration service, complete the following steps:

1. In the Application Development view, right-click **AddressBook**, and then select **Manage included projects**.



   The Manage projects included in Integration Service opens.
2. Select the **AddressBook_DD** data design project.

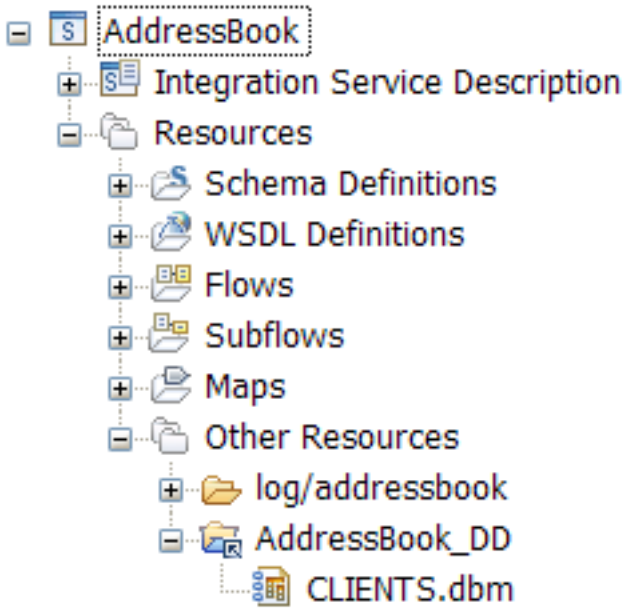


3. Select **OK**.

Results

The data design project **AddressBook_DD** is included as part of the integration service **AddressBook**.

The following figure shows how the data design project **AddressBook_DD** is located under **Other Resources** within the integration service **AddressBook**.



What to do next

After you configure the **AddressBook** integration service to include the data design project **AddressBook_DD**, you must add the database tables to your message map. For more information, see "Adding database tables your message map."

Adding database tables your message map

To retrieve data from the database, you must define which database tables the message map uses.

Before you begin

Configure the **AddressBook** service to include the data design project **AddressBook_DD** that contains the **CLIENTS.dbm** database definition file. For more information, see "Configuring an integration solution to access database resources" on page 21.

About this task

Note: When you add database tables to your message map, you should add all the tables under the same database schema together, that is, one resultset per set of tables. You reduce the number of connections that WebSphere Message Broker requires to retrieve database information from those tables.

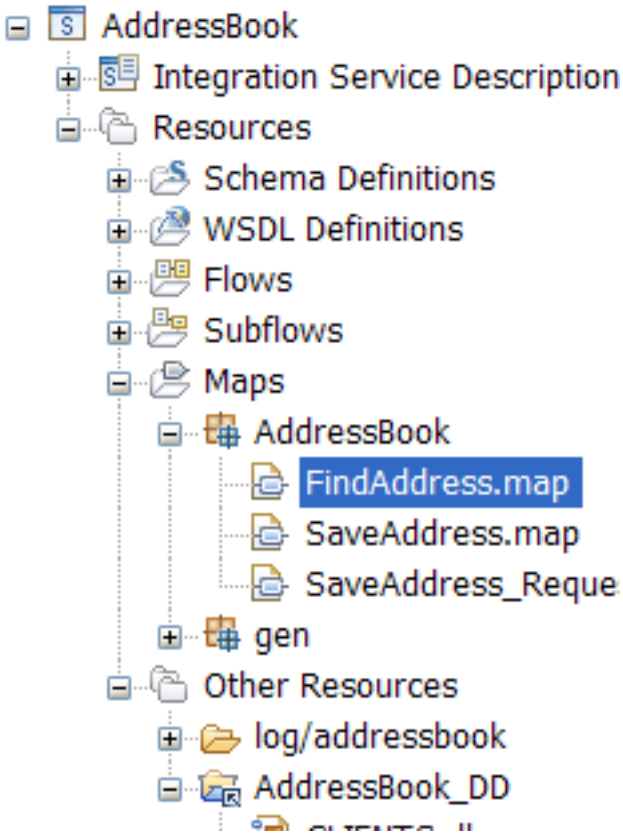
Procedure

Complete the following steps to add PERSON, ADDRESS, and PHONE database tables under the **ADDRESSBOOK** schema:

1. Open the message map **FindAddress** by completing the following steps:
   a. In the Application Development view, navigate to **AddressBook** > **Resources** > **Maps** > **AddressBook**.

b. Double-click on **FindAddress.map**.

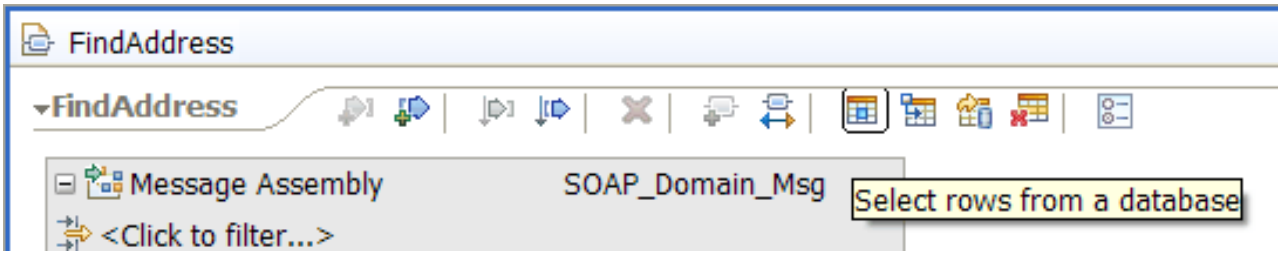
The following figure shows the navigation tree where you can find the message map **FindAddress.map**:



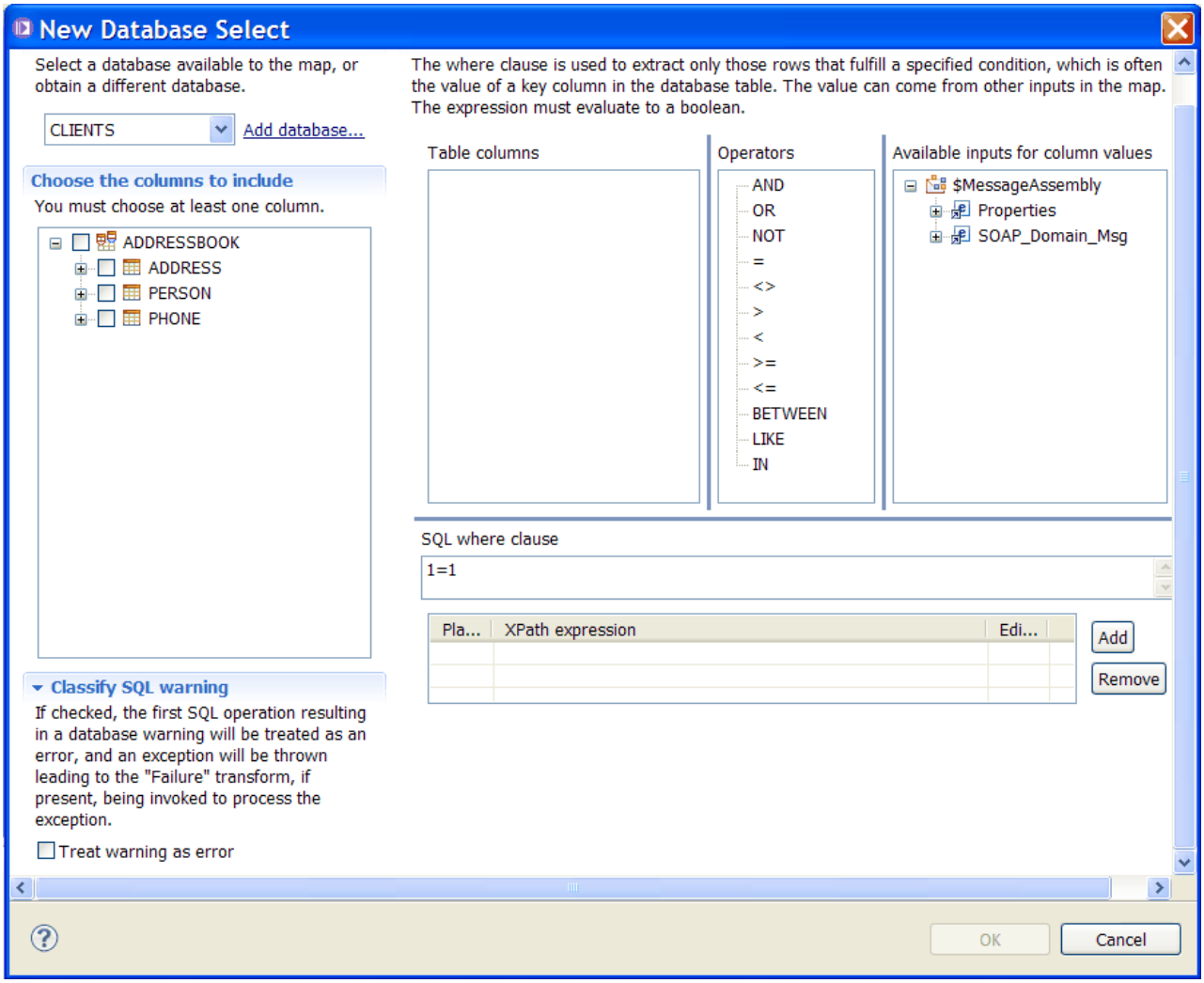
The message map **FindAddress.map** opens in a new tab.

2. Click the **Select rows from a database** icon.

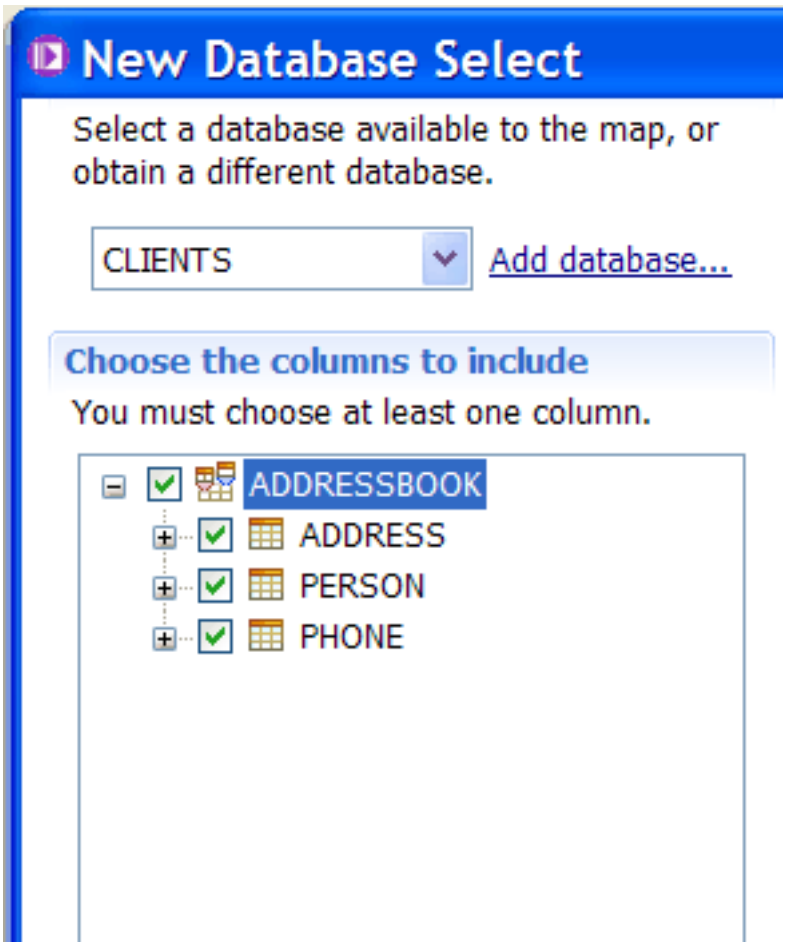
   The following figure shows the icon that you must choose to select the option **Select rows from a database**:



The New Database Select wizard opens.



3. Select the schema **ADDRESSBOOK**, and the database tables PERSON, ADDRESS, and PHONE.



4. Define the **SQL where clause** expression that you use to extract a single address record from the database.

   To define the expression, you can drop or double-click a column, an operation, or an input into the **SQL where clause** pane, use copy and paste, or use content assist (CTRL+Space).

   You can use the following **SQL where clause** expression:

   ```
   ADDRESSBOOK.PERSON.LASTNAME IN ? AND ADDRESSBOOK.PERSON.COUNTRY = ?1
   ```

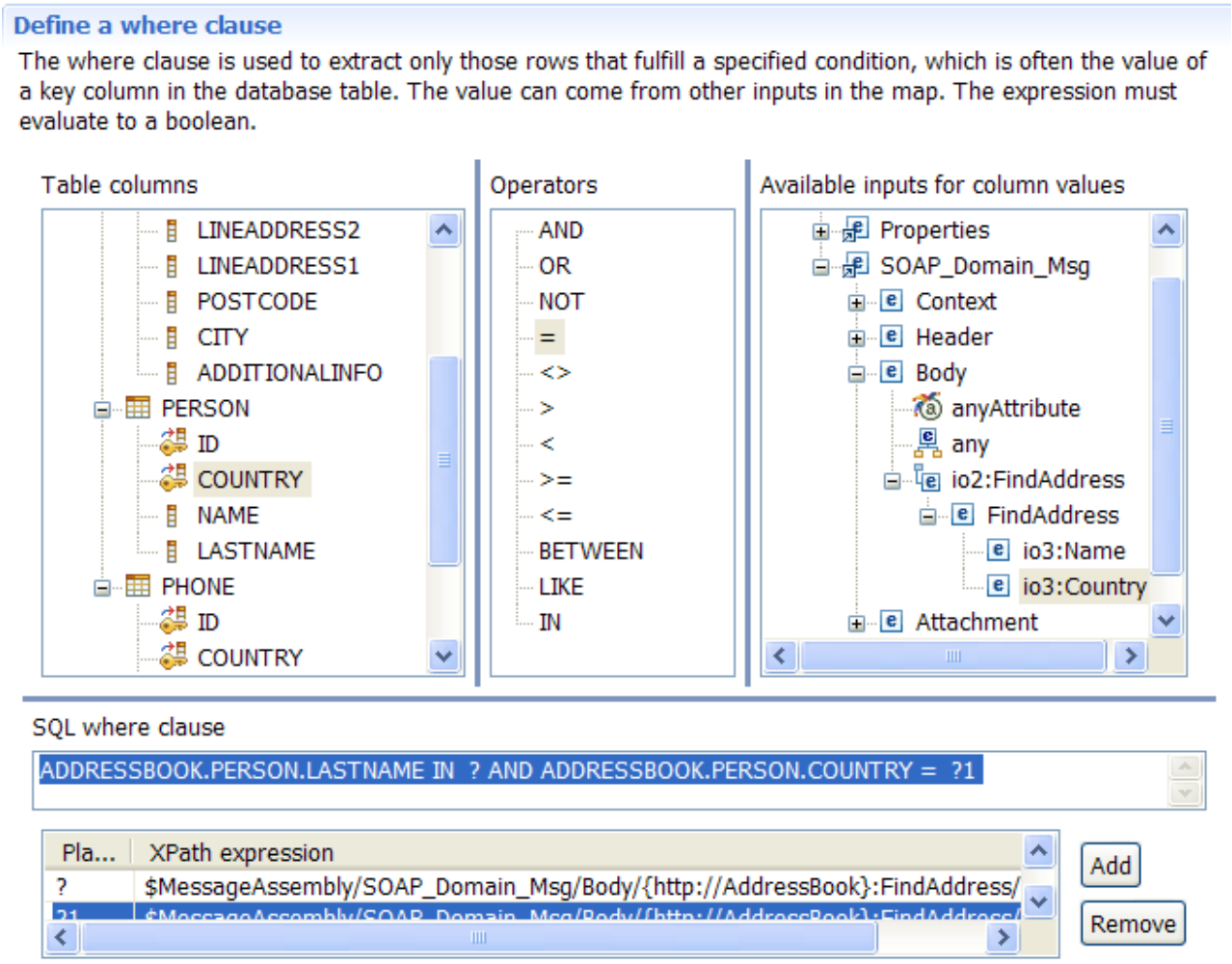
   where *?* represents the XPath expression:

   ```
   $MessageAssembly/SOAP_Domain_Msg/Body/{http://AddressBook}:FindAddress/
   FindAddress/{http://addressbook.com}:Name
   ```

   and *?1* represents the XPath expression:

   ```
   $MessageAssembly/SOAP_Domain_Msg/Body/{http://AddressBook}:FindAddress/
   FindAddress/{http://addressbook.com}:Country
   ```

   The following figure shows the expression defined within WebSphere Message Broker:



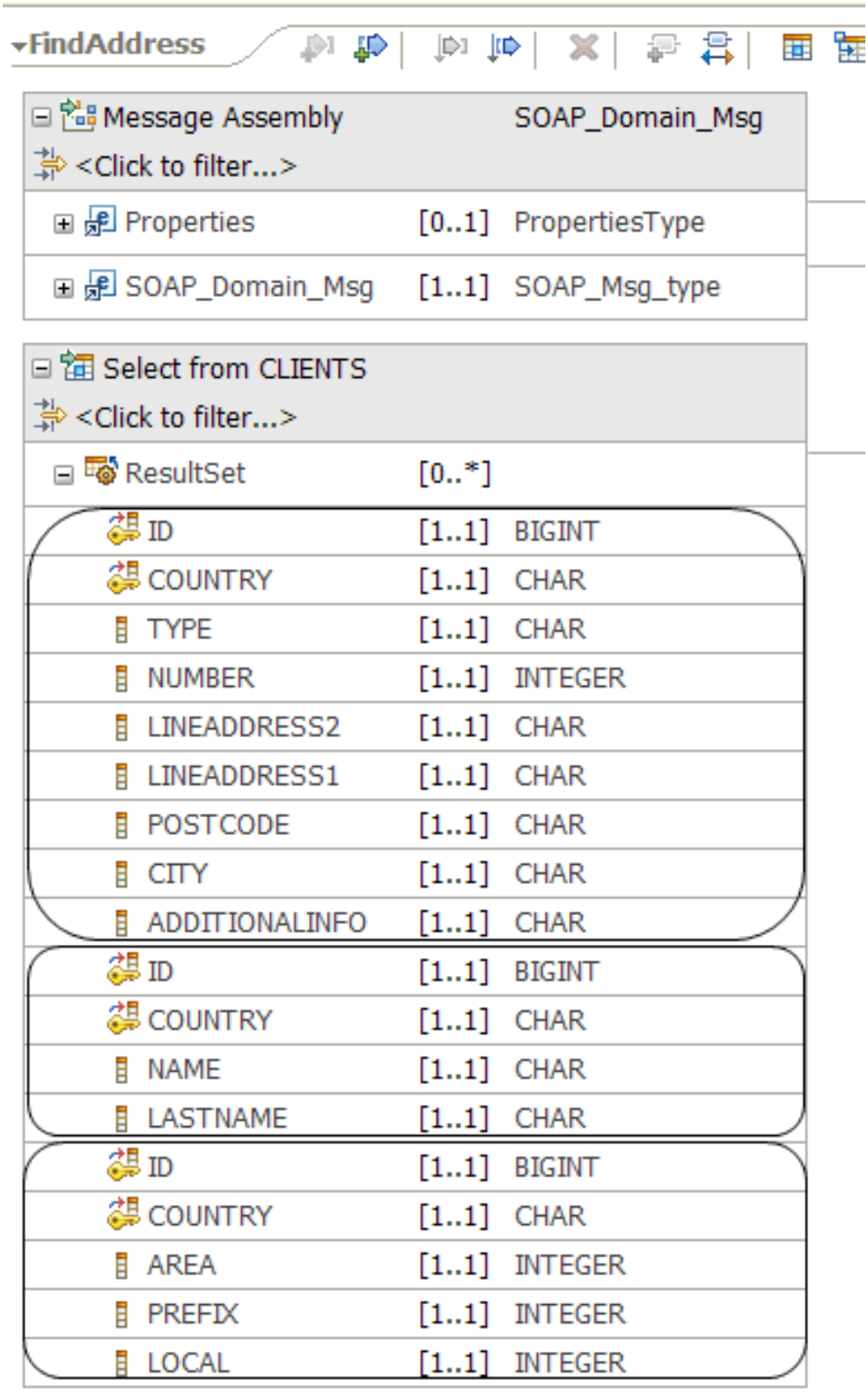
5. Select **OK**.

Results

Under **Message Assembly**, the **Select from CLIENTS** section is added. This section contains one resultset. The resultset has three tables. To see which table an element belongs to, select the element in the resultset, and then view the Properties tab.

Beware that tables are included in alphabetical order.

The following figure shows the resultset that you get when you include the tables PERSON, ADDRESS, and PHONE.



What to do next

You must configure the Select transform in your message map. For more information, see “Configuring the Select transform in a message map” on page 29.

Configuring the Select transform in a message map

You use the **Select** transform to retrieve database information and to perform transformations between the input elements and the output elements of the Message Assembly.

Before you begin

Add the database tables to the message map **FindAddress.map**. For more information, see "Adding database tables your message map" on page 23.

About this task

You can use the **Select** transform in a message map to enrich a message with database information.

The **Select** transform retrieves records from a database based on the **SQL where clause** that you define when you add tables to a message map.

A **Select** transform has a nested map. This nested map is where you transform the input and output elements of the Message Assembly.

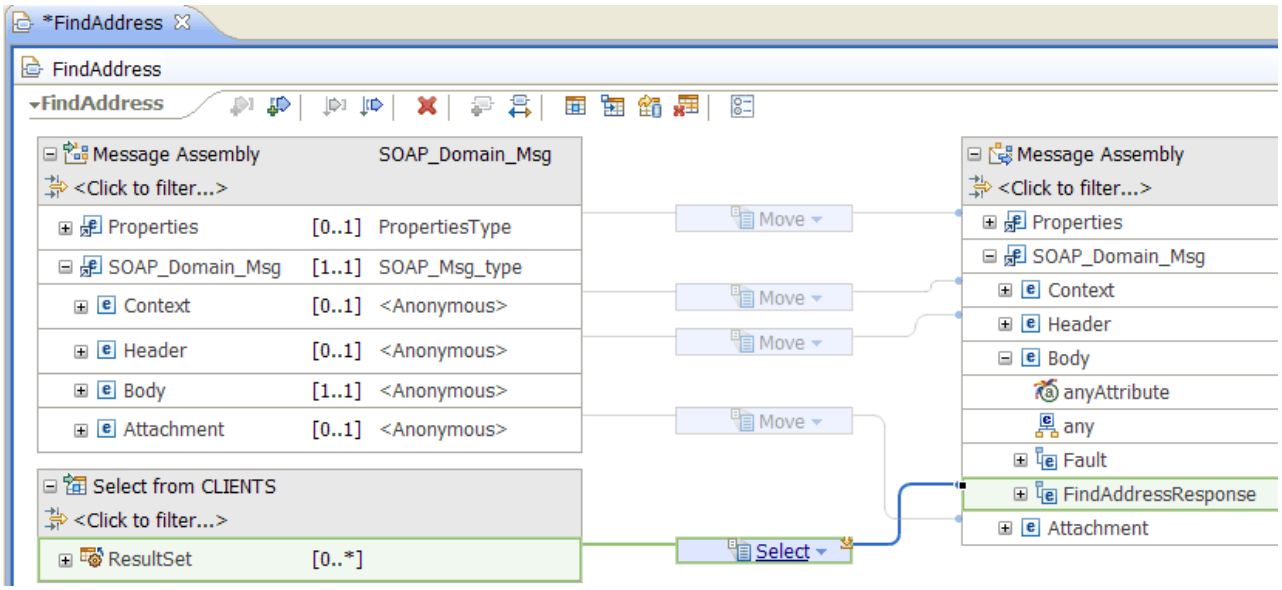
You can set the cardinality of the **Input array indexes** in the **Select** transform properties view to work with a particular row or set of rows, or you can leave this field blank to choose all rows.

This section explains how to configure the **Select** transform when the data available in three database tables is retrieved in one result set.

Procedure

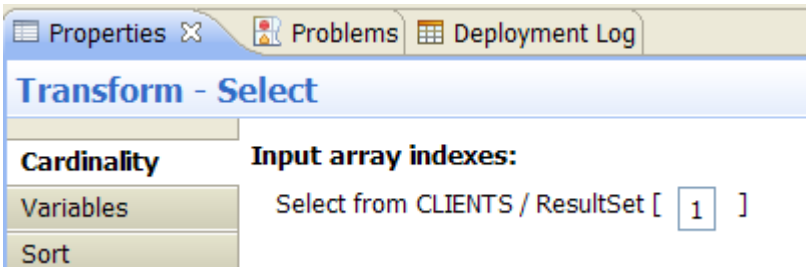
Complete the following steps to enrich a message with the address of a client from the **CLIENTS** database:

1. Open the message map **FindAddress.map** in the Graphical Data Mapping editor.
2. Connect the database section **ResultSet** to the message assembly body section **FindAddressResponse** with a **Select** transform.



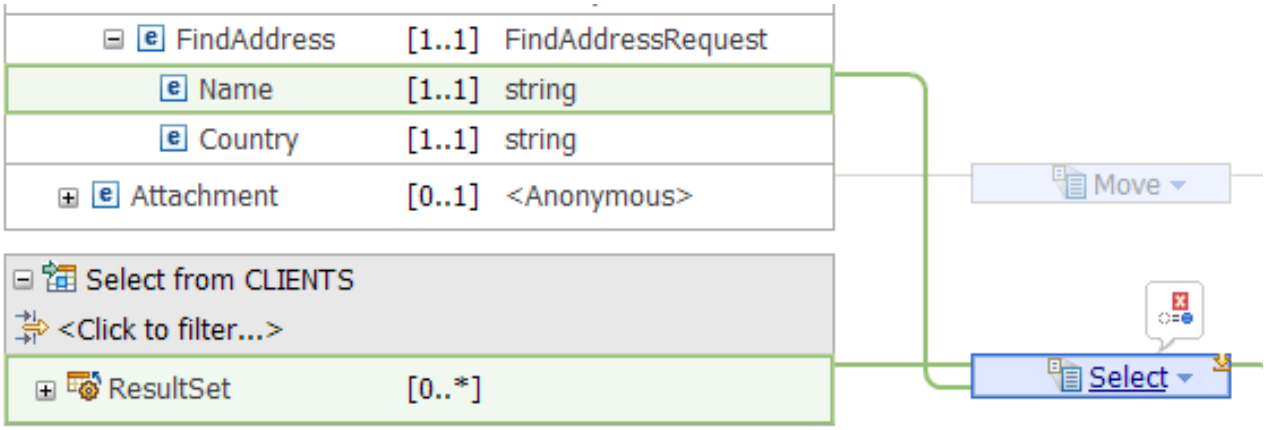
3. Set the cardinality of the **Input array indexes** to 1 in the **Select** transform properties view to indicate that you only wish to work with the first row of the result set returned by the database.

   The following figure shows the Properties tab of the **Select** transform:



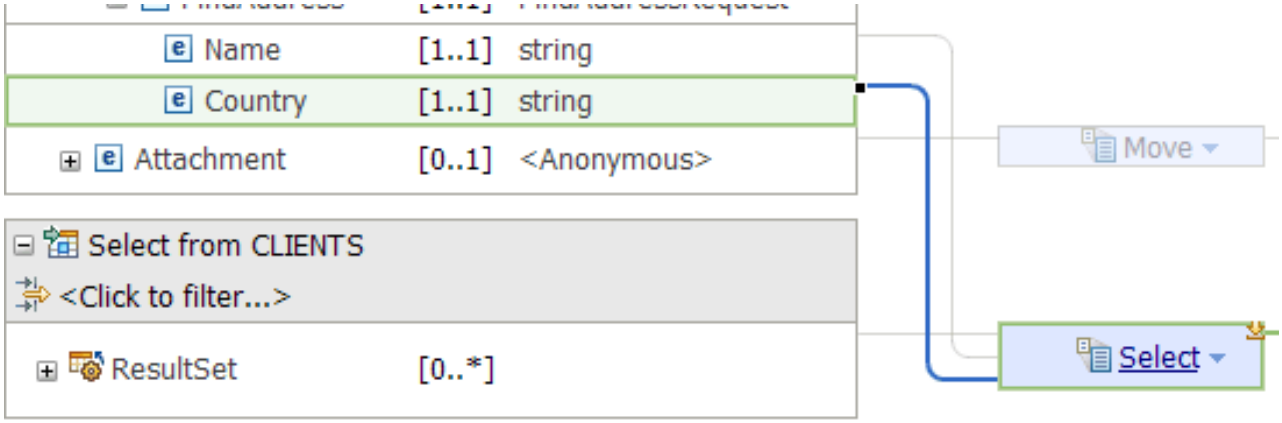
4. Connect the message assembly element **Name** to the **Select** transform.

   The following figure shows the element **Name** connected to the **Select** transform:

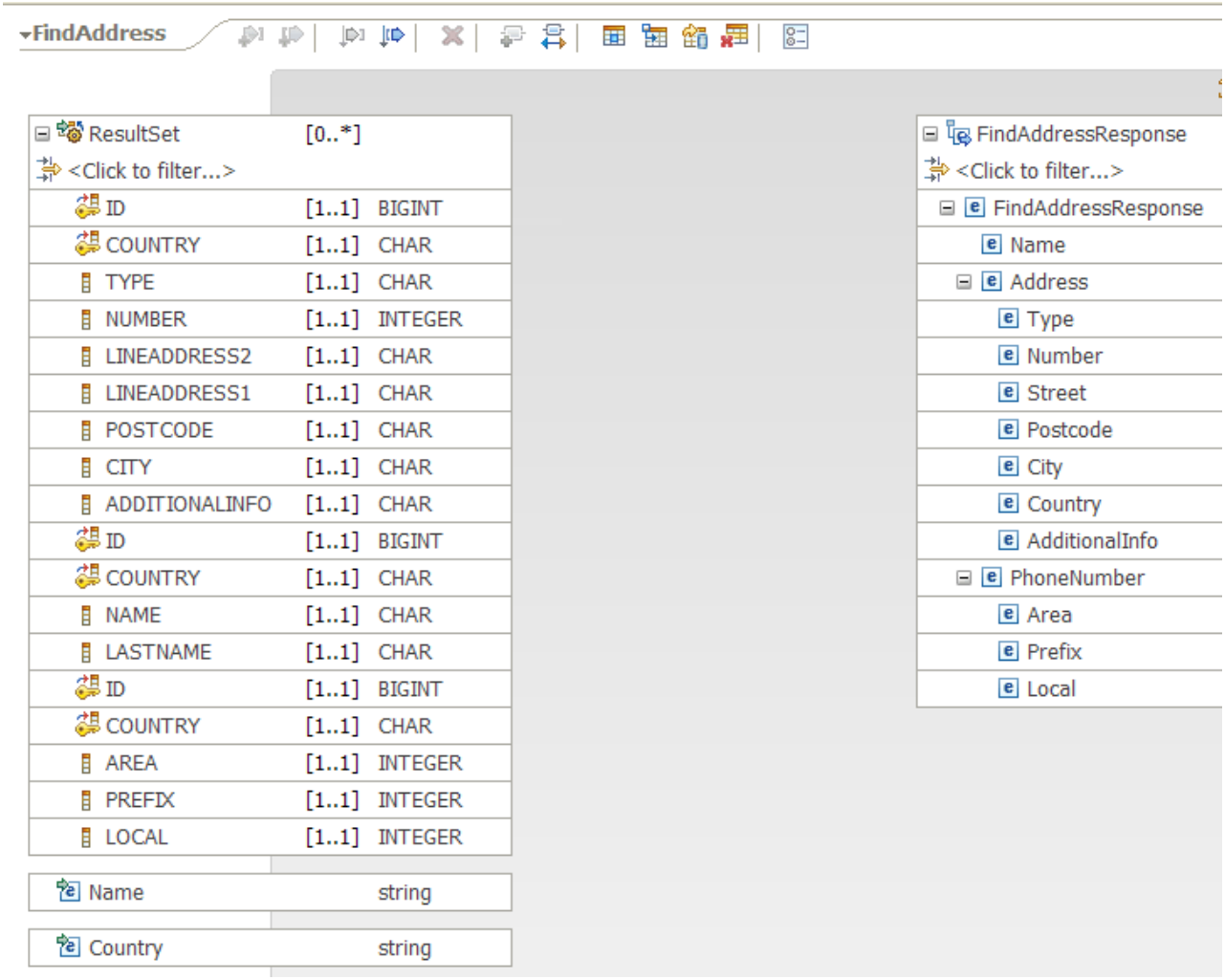


5. Connect the message assembly element **Country** to the **Select** transform.

The following figure shows the element **Country** connected to the **Select** transform:



6. Click **Select**.
   The nested map associated to the **Select** transform opens.
   The following figure shows the nested map with the input and output objects.



7. Define the transforms between the database elements and the message assembly output elements. This can be automatically completed by using the Auto Map capability. To manually define each transform complete the following steps:
   a. Connect the input element **Name** to the output element **Name** in the message assembly body section **FindAddressResponse** with a **Move** transform.
   b. Connect the input element **Country** to the output element **Country** in the message assembly body section **FindAddressResponse** with a **Move** transform.
   c. Connect the element **TYPE** to the element **Type** in the message assembly body section **FindAddressResponse** with a **Move** transform.
   d. Connect the element **NUMBER** to the element **Number** in the message assembly body section **FindAddressResponse** with a **Move** transform.
   e. Connect the elements **LINEADDRESS2** and **LINEADDRESS1** to the element **Street** in the message assembly body section **FindAddressResponse** with a **fn:concat** transform.
   f. Connect the element **POSTCODE** to the element **Postcode** in the message assembly body section **FindAddressResponse** with a **Move** transform.
   g. Connect the element **CITY** to the element **City** in the message assembly body section **FindAddressResponse** with a **Move** transform.

h. Connect the element **ADDITIONALINFO** to the element **AdditionalInfo** in the message assembly body section **FindAddressResponse** with a **Move** transform.

i. Connect the element **AREA** to the element **Area** in the message assembly body section **FindAddressResponse** with a **Move** transform.

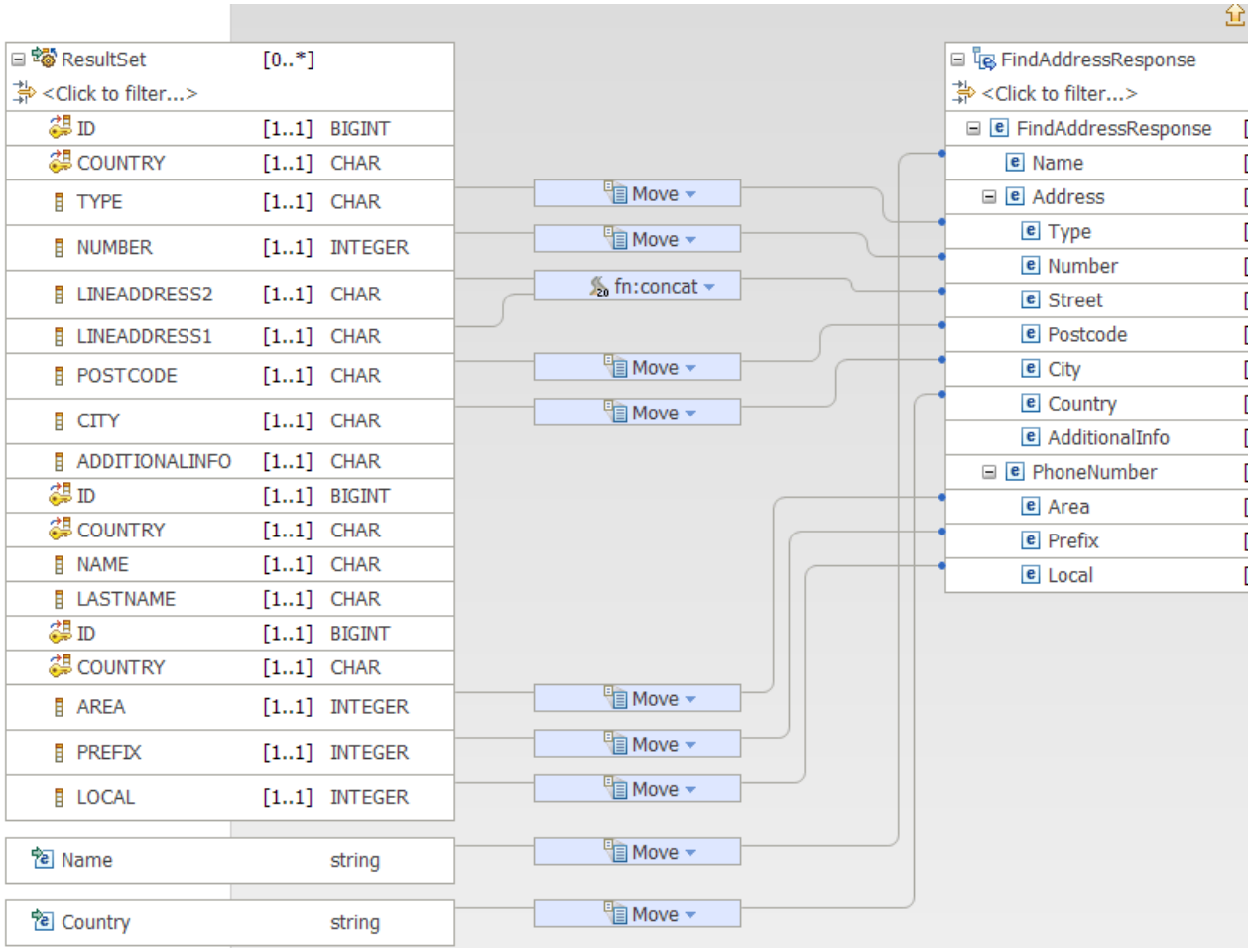
j. Connect the element **PREFIX** to the element **Prefix** in the message assembly body section **FindAddressResponse** with a **Move** transform.

k. Connect the element **LOCAL** to the element **Local** in the message assembly body section **FindAddressResponse** with a **Move** transform.

Results

You have successfully configured the nested map of the **Select** transform.

The following figure shows the nested map:



What to do next

Handle failure of the **Select** transform in a message map. For more information, see "Handling database failures in a Select transform" on page 34.

Handling database failures in a Select transform

You can configure a **Failure** transform for each **Select** transform that you define in a message map to handle explicitly SQL database exceptions. By default, the Mapping node throws database exceptions that can be handled by other nodes in the message flow.

Before you begin

Configure the **Select** transform in a message map. For more information, see "Configuring the Select transform in a message map" on page 29.

About this task

By default, the Mapping node throws database exceptions that the SOAPInput node catches and automatically uses to build a SOAP fault to return to the client.

In the scenario, you use an optional **Failure** transform to process the first SQL exception that might be thrown from the **Select** transform database transaction. You build a SOAPFault to include the database exception detail and the Name and Country elements used for the search of an address which failed.

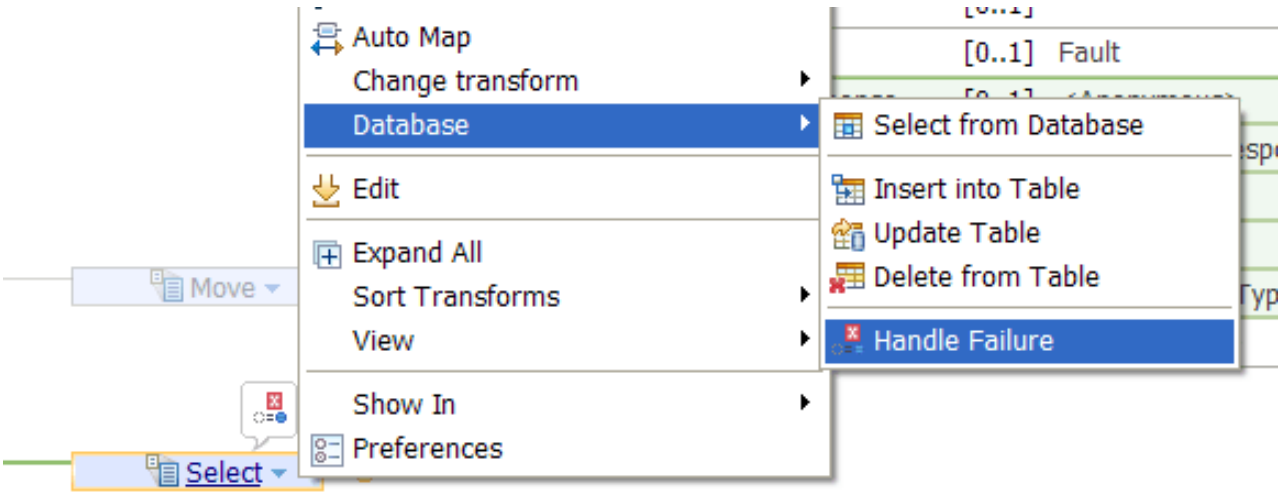
A **Failure** transform has a nested map. This nested map is where you transform the input and output elements of the Message Assembly to define how to handle failure.

Procedure

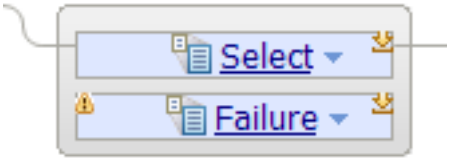
To configure the **Failure** transform in the scenario, complete the following steps:

1. Right-click **Select**, and then select **Database** > **Handle Failure**.

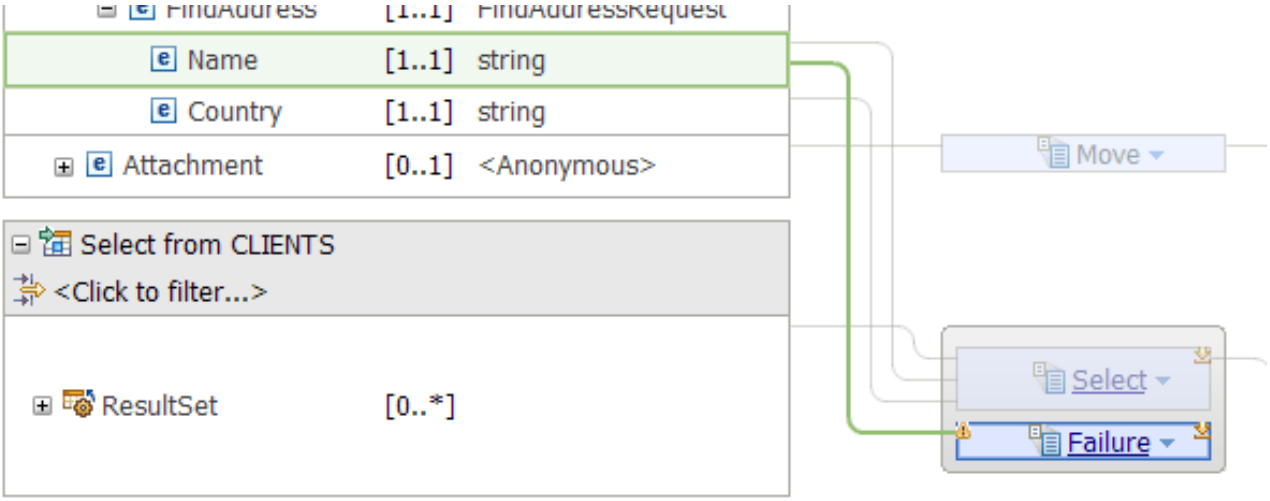
   The following figure shows the graphically the path choices to add a **Failure** transform to a **Select** transform:



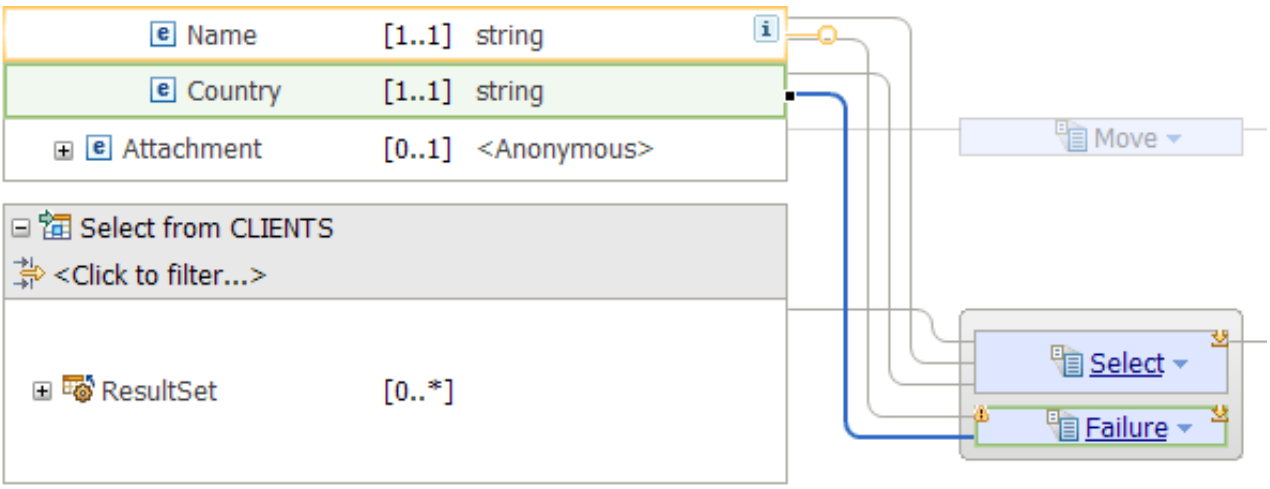
A **Failure** transform is added to the **Select** transform.



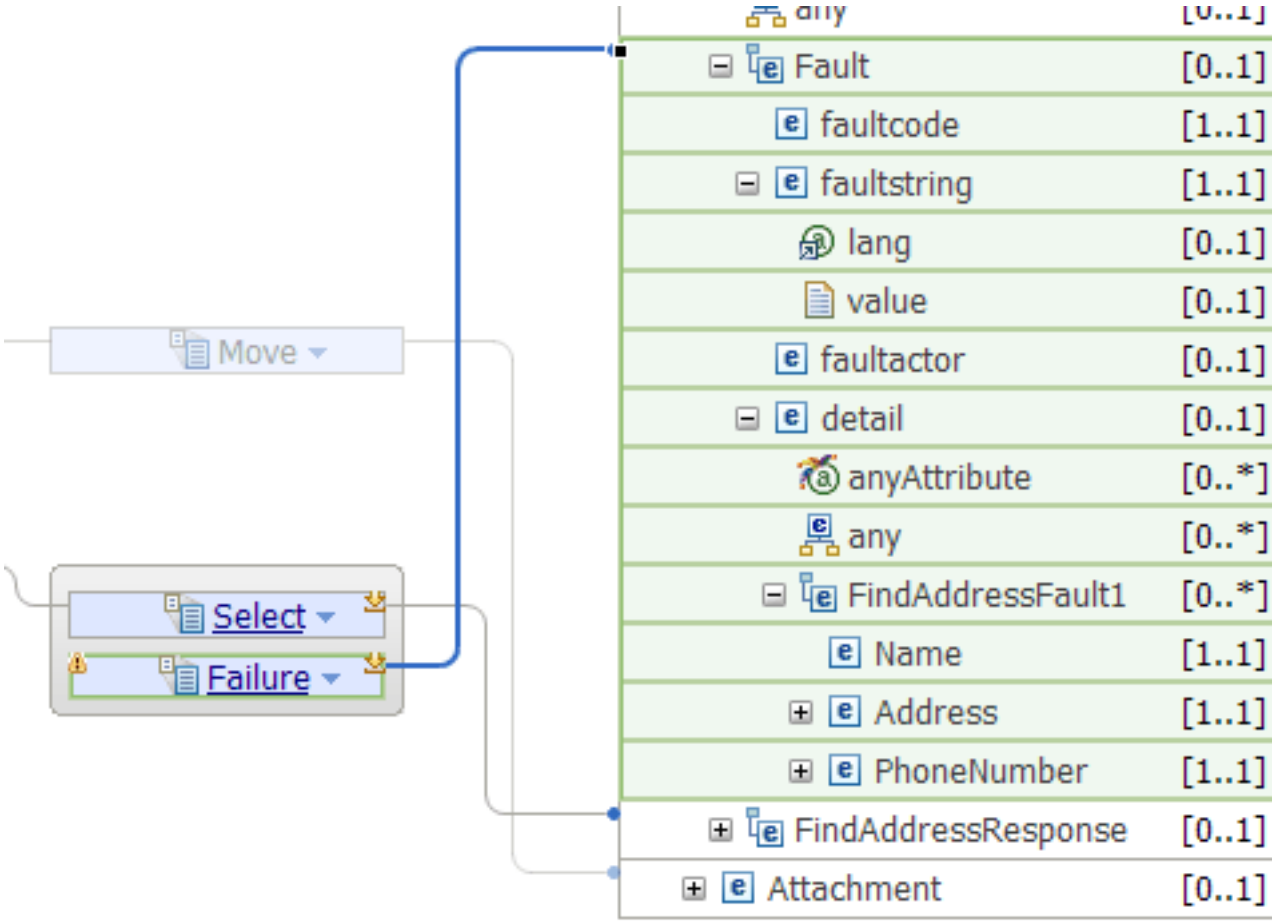
2. Connect the message assembly element **Name** to the **Failure** transform.
   The following figure shows the element **Name** connected to the **Failure** transform:



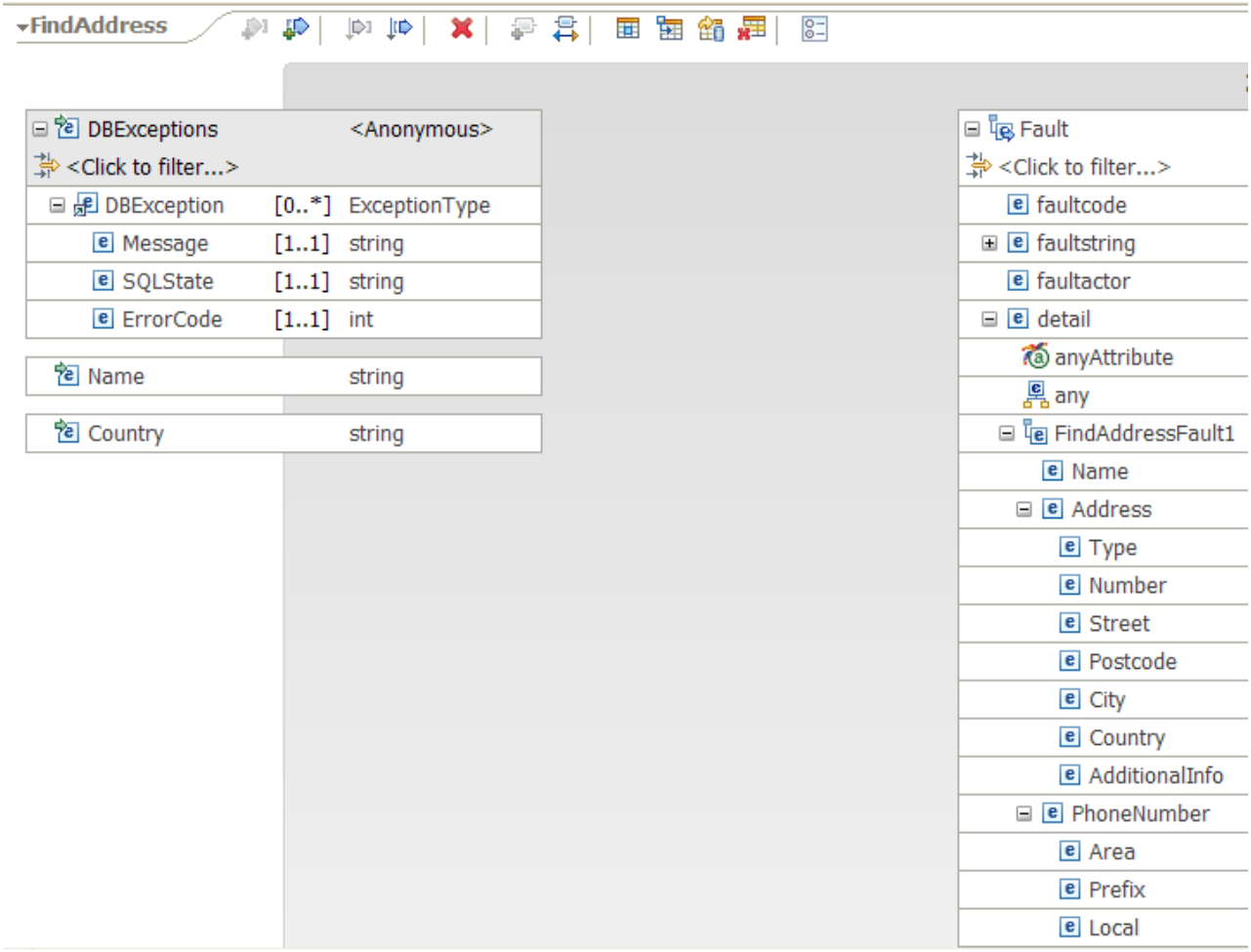
3. Connect the message assembly element **Country** to the **Failure** transform.
   The following figure shows the element **Country** connected to the **Failure** transform:



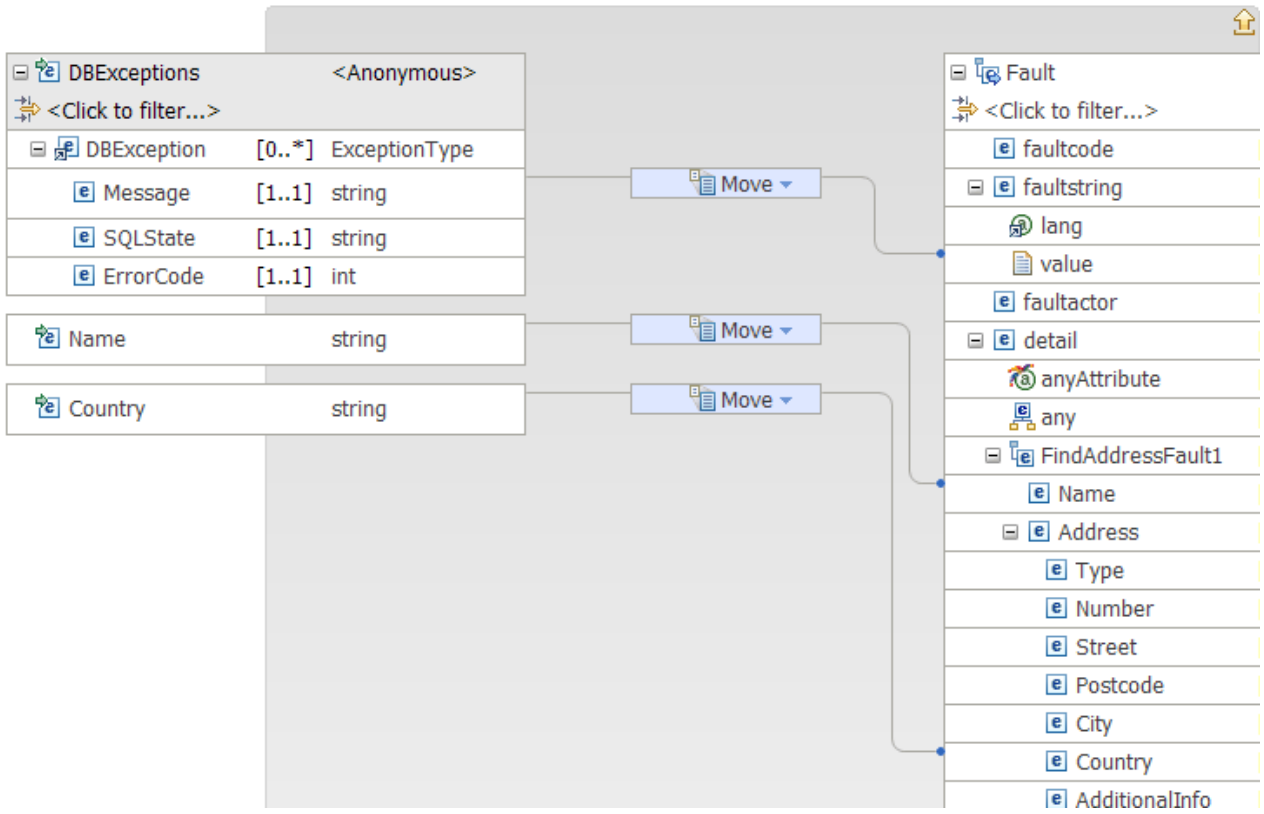
4. Connect the **Failure** transform to the output element **Fault**.



5. Select **Failure**.
   The **Failure** transform nested map opens.

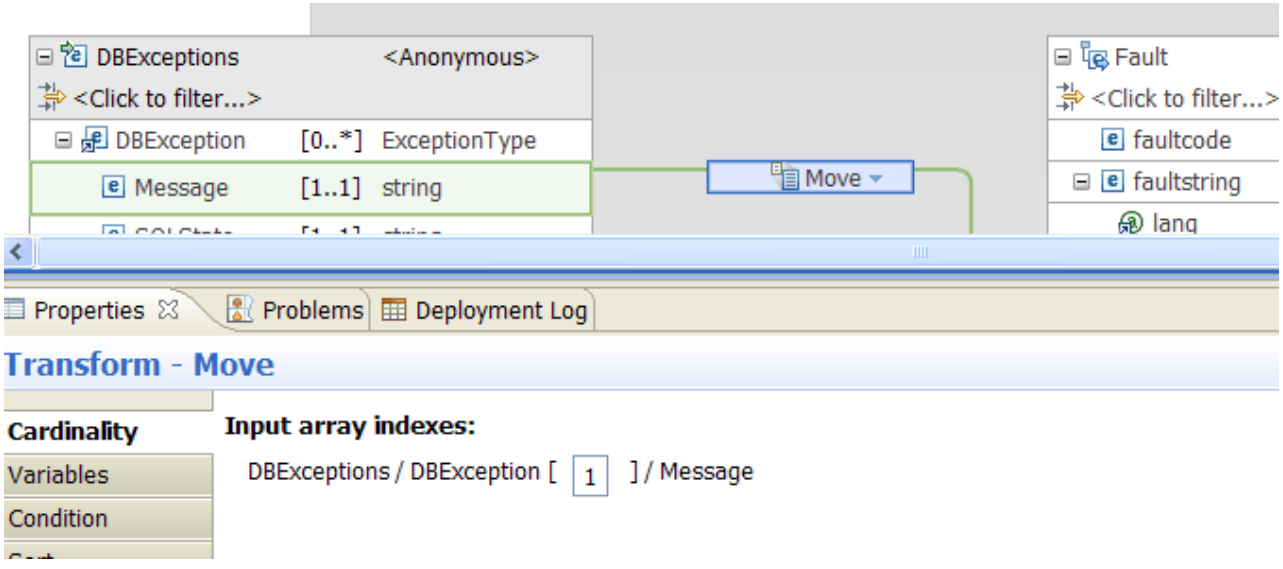


6. Define the transforms between the input elements and the message assembly output elements inside the nested map. Complete the following steps:
   a. Connect the input element **Name** to the output element **Name** in the message assembly body section **Fault** with a **Move** transform.
   b. Connect the input element **Country** to the output element **Country** in the message assembly body section **Fault** with a **Move** transform.
   c. Connect the database exception element **Message** to the output element **value** in the message assembly body section **Fault** with a **Move** transform.



7. Set the cardinality index for the database element **Message** to 1 in the Properties tab of the **Move** transform.

   The following figure shows the Properties tab of the **Move** transform for the element **CITY**:



Results

You have successfully completed the development steps of the scenario.

Configure the JDBC connection at run time. For more information, see "Configuring a database to be available at run time."

Configuring a database to be available at run time

To make database information accessible at run time, you must establish a connection with the database to fulfill the operations that are performed by the Mapping node. You must define a JDBC database connection.

About this task

The WebSphere Message Broker Toolkit connects to the database CLIENTS and runs discovery so the Graphical Data Mapping editor can use the database definition to visualize the database tables PERSON, ADDRESS, and PHONE.

At run time, the Mapping node uses a JDBC provider configurable service to obtain the configuration parameters that will enable it to make the connection to the runtime database that the message map will execute against.

Procedure

To configure the JDBC connection between the WebSphere Message Broker run time and the runtime database, you must complete the following steps:

1. Create a JDBC provider configurable service. For more information, see "Configuring a JDBC provider configurable service."

   The JDBC provider configurable service provides the WebSphere Message Broker run time with the information that it needs to complete the connection to the runtime database.
2. Set up security. For more information, see "Securing the JDBC provider configurable service" on page 41.

   Some databases require access to be associated with a known user ID and password, for other, this association is optional. A DB2® database requires a data source login name and password on all connections.

Configuring a JDBC provider configurable service

You can configure a JDBC provider configurable service by running the **mqsicreateconfigurableservice** command.

Before you begin

The broker named `Server1` is created and started.

About this task

The WebSphere Message Broker Toolkit connects to the database CLIENTS and runs discovery so the Graphical Data Mapping editor can use the database definition to visualize the database tables PERSON, ADDRESS, and PHONE.

At run time, the Mapping node uses a JDBC provider configurable service named CLIENTS to obtain the configuration parameters that will enable it to make the connection to the runtime database PCLIENTS that the message map will execute against.

You configure the runtime database resources by defining the JDBC provider configurable service properties. You must set the following properties:

- **databaseName** property: You must set its value to be the runtime database name PCLIENTS.
- **databaseSchemaNames** property: You must set its value to use at run time the database schema PADDRESSBOOK.

Note: The table names must be the same in the database development environment and in the run time database environment.

The following table lists the database resource names in the development environment, and in the runtime environment:

Table 1. Scenario database resource names

	Development database resource names	**Runtime database resource names**
Database name	CLIENTS	PCLIENTS
Schema name	ADDRESSBOOK	PADDRESSBOOK
Tables names	PERSON, ADDRESS, PHONE	PERSON, ADDRESS, PHONE

Procedure

To configure the JDBC provider configurable service **CLIENTS**, run the **mqsicreateconfigurableservice** command:

```
mqsicreateconfigurableservice Server1 -c JDBCProviders -o CLIENTS -n
databaseName,databaseSchemaNames -v PCLIENTS,PADDRESSBOOK
```

where

- `Server1` is the name of the runtime broker.
- **-o** defines the name of the JDBC configurable service.

  Set the value to the name of the development database, that is, CLIENTS. CLIENTS is the development database name that you used to configure the data definition file in the WebSphere Message Broker Toolkit.
- **-n** defines the list of properties that you must set to configure the JDBC connection.

  These properties are required by the Mapping node to access the database information at run time.

  You must define the **databaseName** property, and the **databaseSchemaNames** property.
- **-v** defines the values you set for each property defined in **-n**.
  - Set the **databaseName** property to be the name of your runtime database, that is, PCLIENTS.
  - Set the **databaseSchemaNames** property to be the name of your runtime schema, that is, PADDRESSBOOK.

Results

A JDBC provider configurable service is available at run time.

What to do next

Secure the JDBC connection. For more information, see "Securing the JDBC provider configurable service."

Securing the JDBC provider configurable service

You secure the JDBC connection to a DB2 database by running the **mqsisetdbparms** command, and the **mqsichangeproperties** command.

About this task

A DB2 database requires a data source login name and password on all connections.

Procedure

You must secure the JDBC connection to the DB2 database by completing the following steps:

1. Identify the user ID and password that you want to associate with the JDBC connection.

   In the scenario, **db2admin** is the user ID used. Request the user ID and password of your installation from your system administrator.

2. Run the **mqsisetdbparms** command to associate the user ID and password with the security identity **scenario** that is associated with the database.

   ```
   mqsisetdbparms Server1 -n jdbc::SecurityIdentity -u userID -p password
   ```

   where:

   - **-n** is the security identity that is used to authenticate the JDBC connection. Set the value to `jdbc::scenario`.

     Note: In the scenario, you create a security identity whose value is **scenario**. However, you can use any name for the security identity. The security identity name that you define in this step must be used to configure the **securityIdentity** property of the JDBC configurable service in the following step.

   - *userID* is your user ID.
   - *password* is the password of the user ID.

   Run the following command:

   ```
   mqsisetdbparms Server1 -n jdbc::scenario -u db2admin -p password
   ```

   Note: The security identity that you define in this step is also used to configure the **securityIdentity** property of the JDBC configurable service.

3. Update the **securityIdentity** property of the **CLIENTS** JDBC configurable service to associate the JDBC connection with the database security identity. Run the **mqsichangeproperties** command.

   ```
   mqsichangeproperties Server1 -c JDBCProviders -o CLIENTS -n
   securityIdentity -v scenario
   ```

   where:

   - `Server1` is the name of the runtime broker.
   - **-o** defines the name of the JDBC configurable service. Set the value to **CLIENTS**.

- **-c** defines the type of the configurable service. Set the value to **JDBCProviders**.
- **-n** defines **securityIdentity** as the name of the property that you must set.
- **-v** defines the value of the **securityIdentity** property. Set the value to **scenario**.

Results

You have secured the JDBC connection.

You have completed the scenario.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information (www.ibm.com/legal/copytrade.shtml).

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44-1962-816151
  - From within the U.K., use 01962-816151
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Printed in USA