Oracle Applications InterConnect

User's Guide

Release 4.0

September 13, 2000

Part No. A86660-01

ORACLE

Oracle Applications InterConnect User's Guide, Release 4.0

Part No. A86660-01

Send Us Your Comments

Oracle Applications InterConnect User's Guide , Release 4.0

Part No. A75325-04

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

Oracle Corporation
Oracle Server Documentation
Attention: Oracle Applications InterConnect
500 Oracle Parkway
Redwood Shores, CA 94065
USA

If you would like a reply, please give your name, address, and telephone number below.

If you have problems with the software, please contact your local Oracle Support Services.

Contents

1 Introduction to Oracle Applications InterConnect

2 Design Time Concepts and iStudio

Index

Preface

Welcome to the ***Oracle Applications InterConnect User's Guide*** **Release 4.0**.

This user's guide provides information to work effectively with Oracle Applications InterConnect, including instructions on how to operate and use the product.

This preface explains how this user's guide is organized and introduces other sources of information that can help you.

Intended Audience

The target audience for Applications InterConnect is any organization that needs to integrate multiple systems that includes Oracle Applications, non Oracle Applications, Message Oriented Middleware (MOM) both intra and inter-enterprise.

This manual contains four chapters:

Chapter 1	Introduces Oracle Applications InterConnect and presents an overview of the product and the tools.
Chapter 2	Describes the design-time components and concepts of Oracle Applications Interconnect. It also explains iStudio, the point and click wizard based tool that allows you to specify your integration logic.
Chapter 3	Describes the runtime components and concepts of Oracle Applications InterConnect.
Chapter 4	Introduces Runtime Management Console and describes how you use it to troubleshoot errors during the execution of interconnected applications and systems.

Audience

This guide is targeted at the following types of users:

- Business analysts and integration engineers, for iStudio.
- System Administrators, for the runtime component.

The audience ideally should have the following pre-requisites, which are discussed but not explained:

1. Domain knowledge of the applications that you are integrating.
2. Database concepts and working knowledge of SQL, PL/SQL, or SQL* Plus.

Related Materials

Additional Documentation

Oracle Applications InterConnect shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other user's guides when you set up and use Oracle Applications InterConnect.

- *Oracle8i PL/SQL Programming Guide*
- *Oracle8i - The Complete Reference*
- *Oracle Message Broker (OMB) User's Guide*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.

Convention	Meaning
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Applications InterConnect working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle8 server, and your hardware and software environment.

1

Introduction to Oracle Applications InterConnect

This chapter provides you an overview of Oracle Applications InterConnect, its features and components.

This chapter contains the following sections:

What is Oracle Applications InterConnect?

Oracle Applications InterConnect (OAI) is a comprehensive and flexible application integration platform that enables seamless integration of enterprise software. It is designed to integrate heterogeneous systems be it Oracle applications, non-Oracle applications or message oriented middleware (MOM). This integration could be deployed either within an enterprise or across various enterprise boundaries.

Applications InterConnect eliminates the complexities of point to point solutions. It allows de-coupled integration of heterogeneous system through the hub and spoke integration paradigm (see Chapter 2 for details). A design time tool, iStudio, allows you to model the integration data and logic. This tool is targeted towards business analysts who understand the functional aspects of the systems participating in the integration scenario. iStudio eliminates the need for "hardwired" or "hard-coded" integrations. Users define their integration using this tool which minimizes (at best, eliminates) the need to

write any code for the integration. The integration information is captured as metadata in a repository.

Applications InterConnect uses the Oracle Integration Server (OIS) for integration plumbing and provides a semantic layer on top to bring end to end integration together as an accessible whole. Oracle Integration Server provides proven, state of the art, reliable, and scalable messaging, communication, and retention services that are utilized by OAI to provide a robust integration solution.

Components

Core Components

OAI has the following core components:

iStudio

iStudio is a design time tool targeted towards business analysts that allows them to semantically map one application to another through events, messages, transformations, etc.

iStudio is designed to minimize (and at best, eliminate) the need for source code development to enable application integration. iStudio is an easy to use, wizard-based tool used to specify and configure the seamless integration of applications using Applications InterConnect.

You use iStudio to model hierarchical data that represents the business objects you are integrating into multiple applications. You also use iStudio to specify data transformations, and implement Publish/Subscribe, Request/Reply, or Point-to-Point messaging paradigms.

Repository

The repository stores all the iStudio modeled information as integration metadata in database tables to be used by adapters as runtime instructions. It provides fine grained versioning support for the integration logic.

Adapters

The adapters connect to applications to transfer data between the application and the hub after transforming it. They use the repository metadata to drive message capture, transformation, routing and delivery. An adapter is the Applications InterConnect component that sits at the spoke with the application to make it InterConnect enabled.

Internally, the adapter is written as two components for improved reuse of existing interfaces. These components are:

Bridge. A bridge is a protocol connector that is responsible for getting data in and out of applications. Two applications using the same protocol may use the same bridge. Third party adapters may be used as bridges too or new bridges could be built using the adapter SDK. The bridge does not understand the semantics of the message. The semantics are the responsibility of agents.

Agent. An agent is a generic engine that processes iStudio repository metadata as runtime instructions. The agent does not know how to talk to a particular application. It uses the bridge to "bridge" the protocol gap.

Runtime Console

The runtime console allows the user to manage and administer different OAI components at runtime from one central console.

Adapter SDK

The adapter SDK allows you to build new adapters by writing code in Java.

iStudio SDK

The iStudio SDK allows you to build new utilities to import metadata from application native repositories. e.g SAP’s Business Object Repository. This imported metadata is used for business object definitions in the integration scenario. This SDK also allows you to write custom transformations in Java.

Dependent OIS Products

OAI uses the following products under the Oracle Integration Server umbrella:

Oracle Internet Directory (OID)

OID is used by OAI as a CORBA naming/location service so that different OAI components and dependent OIS products can communicate to each other via CORBA.

Oracle Message Broker (OMB)

Oracle Message Broker is the hub component that controls all inter-application messaging at runtime. It is the message store and forward unit that utilizes database level messaging support provided by Advanced Queues (AQ). OMB conforms to the Java

Message System (JMS) standard and acts as a JMS server for messaging clients (adapters).

Oracle 8i Database

Advanced Queues in the Oracle database provide low level messaging support.

Applications InterConnect as a Comprehensive Toolkit

Applications InterConnect cleanly separates runtime connectivity (plumbing) from the integration logic (semantics). The semantics are stored as metadata in a repository which drives the runtime behavior of the plumbing.

Applications InterConnect provides a comprehensive toolkit to enable your end to end integration. This toolkit provides three levels of service to the customer:

Level 1. Pre-packaged Plumbing and Semantics

For certain integration scenarios, Oracle provides productized adapters and integration logic for out of the box integration. e.g. Oracle iProcurement 11i to SAP. These integrations are deployed as is and then customized through iStudio. No code needs to be written. Both the plumbing and integration logic is supported 24x7 through Oracle Support.

Level 2. Pre-packaged Plumbing Only

OAI provides a library of pre-built application adapters (plumbing). However, the integration logic (semantics) needs to be built specific to the integration scenario. This is done through iStudio without having to write code.

Level 3. Basic Toolkit

For a particular integration scenario, OAI might not have any pre-built adapters. Using the Adapter SDK you can build new adapters by writing Java code. After the adapter is built, you are at service level 2.

Applications InterConnect Features

Oracle Applications InterConnect's key features are:

* Out-of-the-box integration

- Distributed deployment
- Loose coupling of applications
- Tools for easy customization
- Fine grained versioning of integration logic
- Single point of contact for customer support
- Based on proven Oracle Server Technologies (Java, CORBA, Oracle 8i)
- Guaranteed, exactly once, in-order message delivery
- Content-based routing
- Supports each of the major messaging paradigms—Publish/Subscribe, Request/Reply, and Point-To-Point.
- Enabling Infrastructure

Out-of-the-box Integration

Level 1 service described above provides out of the box pre-packaged integrations which can be be rapidly deployed with little or no need for customizations.

Distributed Deployment

At the heart of OAI runtime, is an event based distributed messaging system. Applications InterConnect's infrastructure supports reliable, high-performance integration between applications installed either locally, distributed over a WAN, or across the internet. This gives you the flexibility of deploying geographically distributed applications.

Loose Coupling of Applications

Oracle Applications InterConnect is based on an asynchronous messaging architecture. This allows the applications being integrated to be loosely coupled with each other. When loosely coupled, applications can continue to function normally even when other participating applications become unavailable. Further, when an application is upgraded or modified, loose coupling minimizes the impact on the other applications.

Synchronous messaging is also supported using the request/reply paradigm.

Tools for Easy Customization

Oracle Applications InterConnect includes iStudio, a wizard GUI driven integration specification tool that allows the business analyst to visually create, review and modify

the integration logic. This integration specification is stored in a repository as metadata. Through iStudio, you get one centralized point for managing your entire integration semantics.

Fine Grained Versioning of Integration Logic

Through the iStudio toolset, you get finely grained versioning for integration logic like events, messages, business objects, transformation mappings, etc.

Single Point of Contact Support

Oracle Applications InterConnect offers customers a 24x7, single point of contact support for the OAI product. Additionally, all integration logic supported through level 1 service is also supported 24x7 through Oracle Support.

Based on Proven Oracle Server Technologies

Applications InterConnect plumbing is built upon the Oracle Integration Server messaging stack and takes full advantage of its features and functionality. The messaging backbone for InterConnect is provided by the advanced queuing (AQ) features of Oracle 8i and Oracle Message Broker (a JMS server).

Together, these technologies enable mission critical scalability and robustness required by enterprise integration platforms.

Guaranteed, Exactly Once, In-order Message Delivery

Oracle Applications Interconnect runtime synchronizes all interconnected applications, and coordinates and verifies the receipt, transformation and delivery of each message in the system to ensure inter-system integrity. These messages are guaranteed to be delivered to the target systems exactly once and in the order they were sent out.

Content-based Routing

Messages can be routed to a specific application based on specific content values contained in the message. For example, an electronic funds transaction settlement application is designed to transmit bank transactions with a specific bank code to identify the destination bank system. When the EFT application publishes each message at runtime, the Oracle Application InterConnect runtime components determine the BankCode value based on objects stored in the repository, and route the message to the appropriate recipient system. No code needs to be written to specify the rules for routing. All rules are specified through wizards in iStudio.

Major Messaging Paradigm Support

Applications InterConnect supports the three major messaging paradigms. The paradigms supported include publish/subscribe, request/reply (both synchronous and asynchronous), and point-to-point. The definitions are used at runtime to route each message appropriately.

Enabling Infrastructure

Applications InterConnect maintains certain enabling infrastructure which is key to cross application collaboration. Examples include cross-referencing (mapping native keys for the same entity across applications), domain value mapping (mapping code tables across applications), event map (mapping application internal events to integration events), etc. Please refer to Chapter 2 for more information.

A Picture Is Worth A Thousand Words

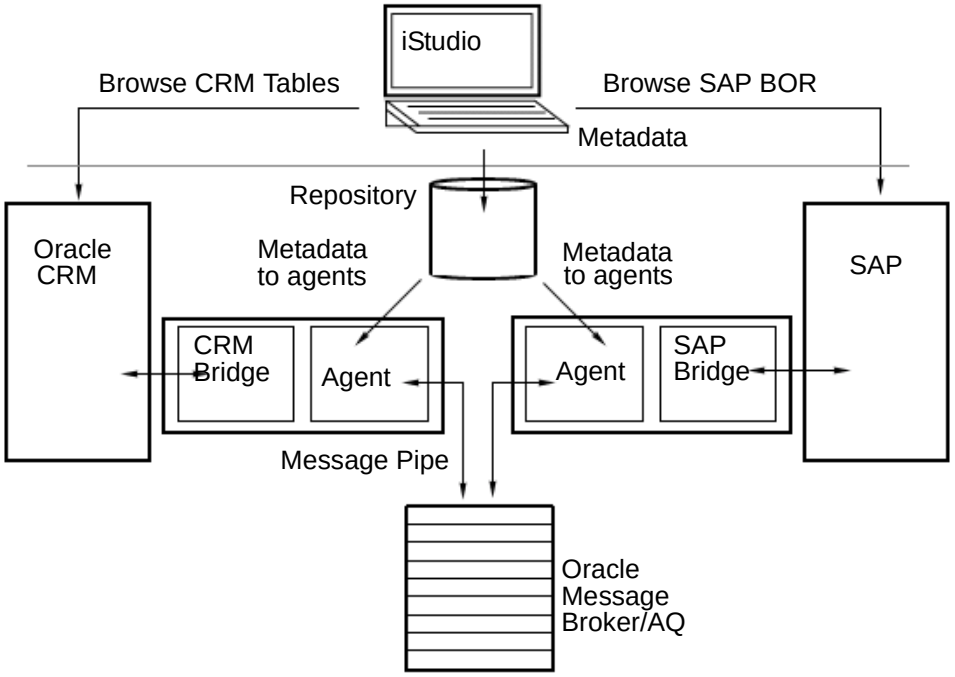


Figure 1–1 An example of Applications InterConnect architecture—Oracle CRM application integrated with SAP R/3 backend.

2

Design Time Concepts and iStudio

This chapter describes Applications InterConnect's design time concepts and iStudio, the GUI-based application for creating metadata that describes events, objects and other types of messages that interconnected applications use during runtime. It also explains how to perform important tasks including publishing and subscribing to an event using iStudio.

This chapter contains the following sections:

Modeling Paradigm

Applications InterConnect modeling is primarily based on the definition of a common format for data to be exchanged between interconnected applications. The model supports a hub-and-spoke architecture at the design level.

Each application possesses a specific format for each data structure used in the application. In iStudio, this "application view" of the data is mapped to a common view and stored as metadata in the Applications InterConnect Repository. At runtime, the metadata is retrieved from the Repository by the Adapter and used to map the application view of a specific data object to the common view and sent out as a message. Incoming messages likewise are converted from the common view to the application view and sent in to the application for processing.

This loose coupled paradigm ensures minimal modifications to integration when an application is modified or a new application is added.

Hub and Spoke Methodology

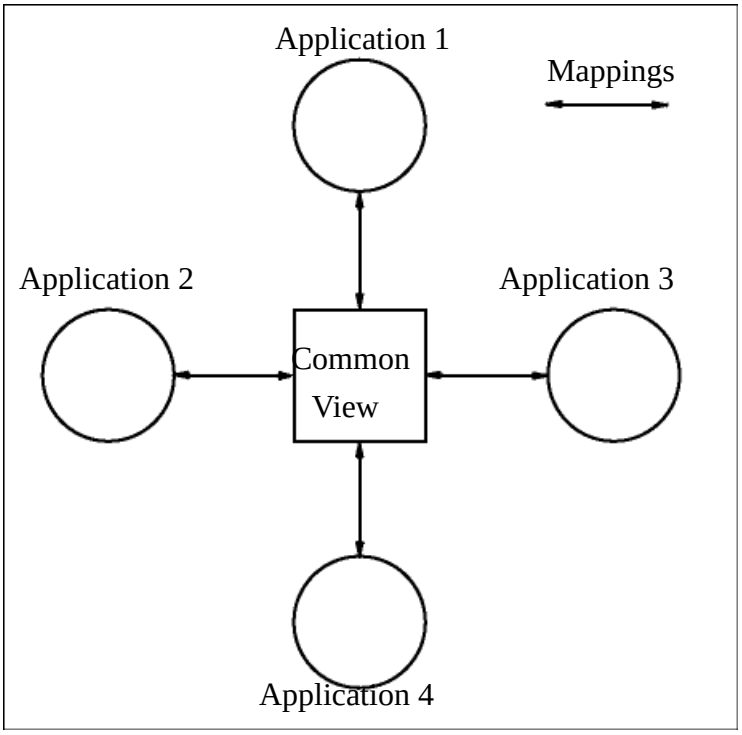


Figure 2–1 Hub and spoke integration methodology used in Applications InterConnect

Applications InterConnect supports three messaging paradigms. These paradigms are defined in iStudio at design time. The definitions are used at runtime to route the messages appropriately.

The messaging paradigms are:

- Publish/Subscribe
- Request/Reply
- Point-to-Point

Publish/Subscribe

Publish/Subscribe messaging is used when an application sends—or publishes—a message to each subscribing application, and does not wait for a reply. This is often referred to as a non-blocking call. Using iStudio, you define which application is the publisher and the subscriber for a particular message.

Request/Reply

Request/reply messaging is used when an application sends a message and waits for a reply. In this paradigm, the application that sends the request blocks—or waits—for a reply. Even though many applications might be listening for this message, only one of them receives the message and sends a reply. The reply is sent using the point-to-point paradigm described below.

Point-to-Point

Point-to-Point messaging is used when the sending application specifies a destination and the message is delivered only to that destination.

Note: If this paradigm is used, a message can be delivered only to one destination.

Integration Process Overview

Application integration using Oracle Applications InterConnect involves two phases. They are:

- Design Time
- Runtime

Design Time

During the design phase, a developer uses iStudio to define the integration objects, applications which participate in the integration, and the specifications of the data exchanged between applications. All the specifications are stored as metadata in the Applications InterConnect Repository.

Runtime

For each application participating in a specific integration, Applications InterConnect attaches an Adapter to it. At runtime, the Adapter retrieves the metadata from the Repository

to determine the format of messages, and perform transformations between the various data formats, and route the messages to the appropriate queues under OMB.

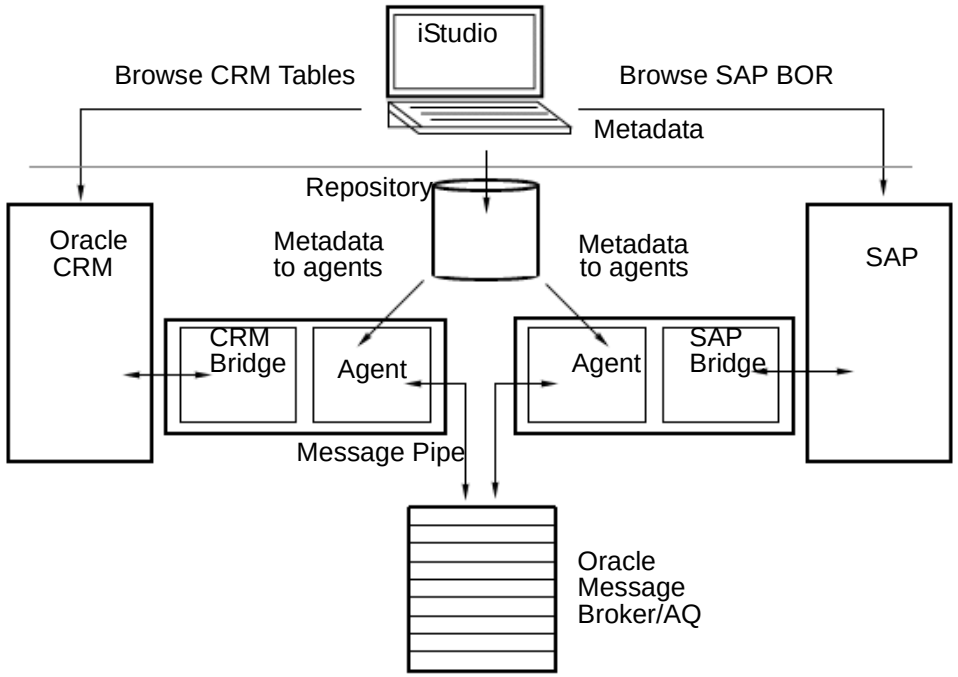


Figure 2–2 A graphical overview of design time and runtime phases in integration

iStudio Concepts

Applications InterConnect modeling is accomplished using iStudio, an easy-to-use graphical tool. These sections describe the concepts implemented by iStudio.

Projects

In iStudio, a project defines a Repository connection. To create a project, you have to specify the Repository connection information including the Repository name and host. Several people may work on the same project.

Workspaces

Workspaces store user settings and preferences (for example, SAP and DB login credentials) in iStudio. In contrast, each iStudio project stores information about the current Repository. When a project is opened in a workspace, it is automatically added to the list of recently opened projects. These are accessed by clicking **File** - **Reload**. To reload the Repository, click on a project in the **Reload** menu.

Applications

Each component that is integrated with Applications InterConnect is referred to as an application. Each application can specify the events and procedures it is interested in. The user can define the application views and transformations between these common views and their application views.

Common View

Applications InterConnect de-couples applications by introducing the concept of common view. Procedures, events, and common data types (described below) together make the common view. The developer only needs to define interactions of an application with the common view, and not with other applications directly. In other words, a point-to-point architecture is replaced by a hub and spoke architecture, with common view serving as the representation of data in the hub.

Business Objects

For convenience and grouping, procedures and events are organized into business objects. A business object roughly corresponds to an object-oriented class. Procedures in the business object resemble methods in the class. For example:

Customer with a `getAddress()` procedure and a `newCustomer` event.

Events are used to model the publish/subscribe paradigm. Procedures are used to model the request/reply paradigm.

Procedures

Procedures are part of the common view that encapsulate functionality. Each procedure has a name and IN/OUT arguments. iStudio allows a developer to define procedures that different applications can invoke, or implement. For example:

```
getAddress(IN string Name, OUT string Address) where,
getAddress - name of the procedure
Name - IN argument of type string denoting person’s name
Address - OUT argument of type string denoting person’s address
```

In the case of two applications being integrated (for example, SAP and CRM), the CRM application can invoke the procedure `getAddress`. It does not need to know or specify which application implements the procedure. The procedure may be implemented by any ERP application such as the SAP application. Applications InterConnect does the necessary mappings at run-time and invokes the appropriate ERP function using the vendor-specific bridge.

There are two styles of the request/reply paradigm:

1. Asynchronous. The procedure invoking application does not block waiting for a reply. It makes an invocation and then continues normal processing. The OUT arguments are returned to the caller using a mechanism similar to callbacks.

2. Synchronous. The procedure invoking application blocks until it gets a reply. The OUT arguments are returned to the caller as part of the invocation. The caller then unblocks and continues with normal processing.

Events

Applications may be interested in events that originate from another application. For example, when a CRM application when adds a new customer, the ERP application must be notified of the new customer. iStudio allows developers to define events that encapsulate this kind of information. Applications may publish or subscribe to events.

An Event is defined by a name and the data attributes it contains. For example:

```
EVENT newCustomer CONTAINS
{
String Name,
String StreetName,
String City
}
where,
newCustomer - Name of the Event, and
Name, StreetName, City - Event data attributes
```

NOTE: Events and procedures are part of the common view. So, the data they contain is normally a superset of the data that being integrated across applications. (See Common Data Type for more information).

Application View

Each application that uses Applications InterConnect for integration defines its own definition or view of the common view which is called the application view. This may be different or the same as that of the common view it mirrors.

For example, an application view that corresponds to the `getAddress()` procedure may be represented in this way:

```
getAddress(IN string LastName, IN string FirstName, OUT string Street, OUT
string City) where,
getAddress - name
LastName - IN argument of type string denoting person's last name
FirstName - IN argument of type string denoting person's first name
Street - OUT argument of type string denoting person's street address
City - OUT argument of type string denoting the city
```

Application views are mapped to or from common views depending on whether the application is invoking/publishing or implementing/subscribing a function module.

Event Map

Event Maps are for published events and invoked procedures. They allow you to associate your application's data with OAI messages. For example, you may have two events - CreateCustomer and UpdateCustomer, but your application may have only one view for these two events - a Customer XML message. Your Customer XML message might have an Action field in it. If the Action is equal to CREATE at runtime, then you want to the CreateCustomer event to be published. However, if the Action is equal to UPDATE at runtime, then you want the UpdateCustomer event to be published.

When you are defining the event maps for a particular message, you are saying that if these fields at runtime are equal to the supplied values, then this message should be published (or invoked). For example, if you are creating (or editing) the CreateCustomer event, you would set the event map condition that Action should be equal to CREATE. If you were creating (or editing) the UpdateCustomer event, you would set the event map condition that Action should be equal to UPDATE.

You may include as many event map fields as you want. If you have multiple event map fields, all of them must be equal to their supplied values at runtime in order for the message to be published. For example, if you are creating (or editing) the CreateCustomer event and say that Customer.Action should be equal to CREATE and Customer.Country should be equal to FR, then at runtime, the CreateCustomer event will only be published if your application produces an XML that has Action equal to CREATE *and* Country equal to FR.

Note: event maps are currently only supported for the XML message type.

Mapping and Transformations

Each application's view of data may be different than the common view. In these situations the developer should specify how fields in the application view map to fields in the common view. This mapping may also involve simple transformations on the fields being mapped.

For example, the `LastName` and `FirstName` fields of the application view may need to concatenated to the `Name` field in the common view. Applications InterConnect provides a set of standard transformations. However, the developer may define custom transformations in Java which can be imported into iStudio and plugged into the runtime system.

Data Types

Data types enable users to model complex hierarchical data that is exchanged between applications. For example, a purchase order contains a header object and one or more line item objects. Both the header and line item can be defined as data types that are then used to define a PurchaseOrder data type.

Data types are also useful when you want to reuse data. For example, you can create a common data type (described below) called `customer` if you have multiple events that utilize customer information. This is a superior alternative to importing or typing the data that is transmitted with these events repeatedly.

Application Data Types

Data types defined in the application view of each application are referred to as application data types.

Common Data Types

Data types defined in the common view are referred to as common data types.

Metadata Versioning

iStudio supports versioning for Common Data Types, Application Data Type, Events, Procedures, and Messages.

Comprehensive versioning support is achieved by using the concept of an owner of these objects in addition to versions. An owner is the creator of the object. Only the creator of an object can modify the object. However, other users can create new versions or copy the original object under a new name. The owner is specified at the time of Repository installation.

In the following example, the metadata is being created at Oracle, and at the time of Repository installation, "OAI" was specified as the owner of the metadata. The following functionality is available for versioning:

Automatic Versioning First, an event called "NewCustomerEvent" is created. When you create this object for the first time, the assigned owner is OAI and the version is V1. This even is `NewCustomerEvent/OAI/V1`.

Modify Object If you are the owner, you can change the contents of the event (the data associated with it) by clicking **Edit**, but you cannot change the version number or the name of the event. The event is `NewCustomerEvent/OAI/V1`.

Create New Version If instead, you want to keep the original `NewCustomerEvent` but want to create a new version of the information with modified data, click **Create New Version**. When you save this version, you now have two objects — `NewCustomerEvent/OAI/V1` and `NewCustomerEvent/OAI/V2`.

Load Version Not all versions of objects are loaded into iStudio. To work with a specific version of an object, use the **Load Version** capability. In the scenario above, when you created a new version it became your current version. Now to load `NewCustomerEvent/OAI/`V1, you should use **Load Version**.

Copy Object To create a `NewBigCustomerEvent` which has a lot of common elements with `NewCustomerEvent/OAI/V1`, first load `NewCustomerEvent/OAI/V1` and then click **Copy Object**. Using **Copy Object** allows you to not only modify the data, but also modify the name of the event. When you have modified the name of the event, `NewBigCustomerEvent/OAI/V1` and `NewCustomerEvent/OAI/V1` will both coexist in the Repository.

NOTE: You cannot type in a name that already exists.

In our scenario above, all the metadata was built at Oracle. Now we can transmit this metadata to a customer, NewCorp. When NewCorp installs the Repository and specifies the owner as `NewCorp`, the metadata is in a read-only state. Now, if they want to customize `NewBigCustomerEvent/OAI/V1`, they cannot modify the existing version since the owners are different. They can however, use the other features described above.

To customize the metadata, they must create a new version, so that `NewBigCustomerEvent/OAI/`V1 and `NewBigCustomerEvent/NewCorp/V2` coexist in the Repository. The client can use both events in defining messages if required, and NewCorp can now modify the event it owns.

Content-Based Routing

Content-based routing increases message delivery efficiency by routing each message directly to the intended application by examining specific attributes in the data object.

Cross Reference Tables and Domain Value Mapping

Cross Reference Tables and Domain Value Mappings are features supported by the Applications InterConnect platform to facilitate easy creation of mapping tables at design time, and their population, lookup, and deletion during runtime.

For example, consider two heterogeneous applications A and B being integrated through Applications InterConnect. To achieve integration, there is a requirement to maintain mapping tables that track the associations between the various identifiers (ids) created by the different systems. A *customer* created on application A has an id 100, while the same *customer* created on application B, as part of the integration, has an id 10000. The association between 100 and 10000 needs to maintained for subsequent information interchanges between the two applications. For example, if the address of the *customer* with id 100 is being updated by application A, and the change is propagated to application B, the id 100 needs to be automatically transformed to 10000 to ensure the correct record is updated at B. The Cross Reference Tables address this functionality.

In case of Domain Value Mappings the population of data occurs at design time. This feature is useful when the associations between the various ids (or names) is known at design time.

For example, the order status might be represented by *BOOKED* on application A, while its equivalent on B might be *ORDERED*. The association between *BOOKED* and *ORDERED* is maintained in a domain value map for order status.

Event-Based Routing

General routing of messages is based on the events applications publish and subscribe and the procedures applications invoke and implement.

Message Capability Matrix

On a per application basis, for every published message, you can specify a hub queue in which the message should be stored. Conversely, for each subscribing application, you can specify a hub queue from which the message should be retrieved. This pairing of publish and subscribe queues constitutes the Message Capability Matrix for each application. Using this matrix, the integrator can determine which queues need to be created in the hub. This matrix is stored in the Repository as metadata and is used by the adapters (see runtime section) to route messages to the appropriate queue on behalf of publishing applications, and to listen for messages on the appropriate queues on behalf of subscribing applications.

Message Partitioning

For performance reasons, you can partition the messages in the Message Capability Matrix. At runtime, adapters can be assigned a partition. This way they process only a subset of the messages in the Message Capability Matrix.

Application Queues

Adapters processing XML type messages need to be assigned the spoke application AQs from which they will dequeue messages they send and enqueue messages they receive.

Message Type

Message Type specifies the mode of communication between an Adapter and an application.

Using iStudio

iStudio is the GUI-based development tool that implements the concepts described in the previous section.

This section describes how to perform the various tasks required by Applications InterConnect during the design phase using iStudio. You can perform the tasks by referring to the steps described in the following sections.

* Start the iStudio tool. The following window is displayed: (Figure 2–3)

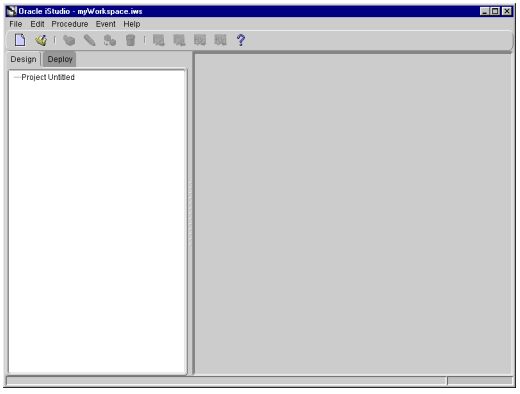


Figure 2–3 iStudio window

Toolbar

Figure 2–4 iStudio tool bar options

- You can also select the tasks by clicking the icons in the toolbar.

Creating a New Project

To define a new project, click **File,** and select **New Project** in the iStudio main menu panel. iStudio displays the New Project Dialog box: (Figure 2–5)

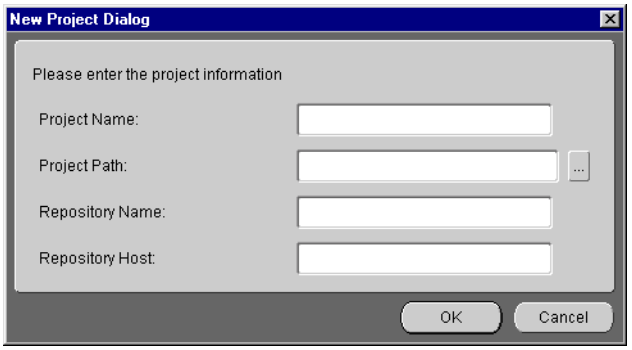


Figure 2–5 New Project dialog

Enter the project details as follows:

- **Project Name**—the project name for the integration.
- **Project Path**—the directory path where the project will be stored.
- **Repository Name** —the Repository where the project will be stored.
- **Repository Host**—the Repository host name.

Click **OK** to save the record, or **Cancel** to abort and exit.

Creating Workspaces

To create a new workspace, click **File** and select **New Workspace**. The New Workspace Dialog is displayed:

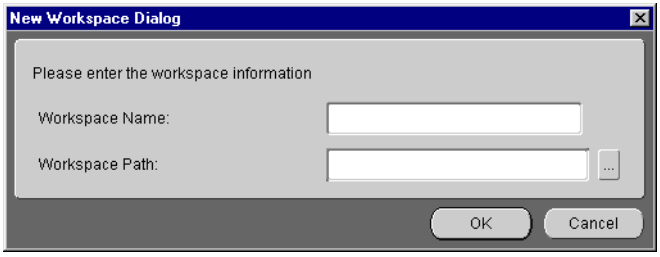


Figure 2–6 New Workspace Dialog

Enter the workspace details as follows:

- **Workspace Name**—the Workspace name for the project.
- **Workspace Path**—the Workspace directory location.

Click **OK** to save the record, or **Cancel** to exit without saving the workspace.

When you start iStudio, the default workspace **myWorkspace.iws** and the last opened project is automatically loaded. To save the SAP and DB login credits, check the box **Save settings as default** in the SAP and DB login dialogs. The user settings are automatically saved to the workspace.

Opening an Existing Workspace

To open a workspace that has already been created, click **File** and select **Open Workspace**. The file system dialog is displayed. Enter the workspace name and path to open the workspace.

Creating an Application

To create an application, click **File** in the iStudio main menu panel, click **New** in the pull-down menu, and select **Application**. The Create Application window is displayed: (Figure 2–7)

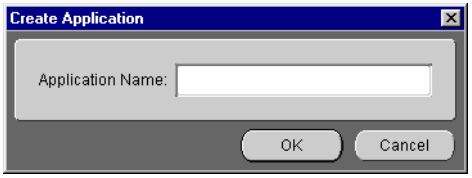


Figure 2–7 Create Application window

Enter event details as follows:

- **Application Name**—the name of the application.

- Click **OK** to save and exit, or **Cancel** to exit without saving.

Creating a Business Object

To define a new business object, click **File** / **New**, and select **Business Object**. The Create Business Object window is displayed: (Figure 2–8)

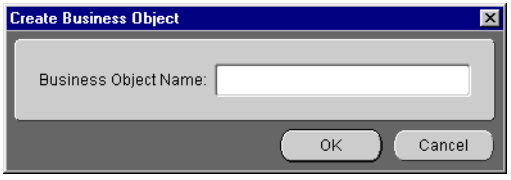


Figure 2–8 Create Business Object window

Enter the business object name:

- **Business Object Name** —the name of the business object being created. (Only alphanumeric characters are allowed.)

Click **OK** to save the new business object or **Cancel** to exit.

Creating Common Data Types

To define a Common Data Type, click **File**/**New** and select **Common Data Type.** The Create Data Type window is displayed: (Figure 2–9)

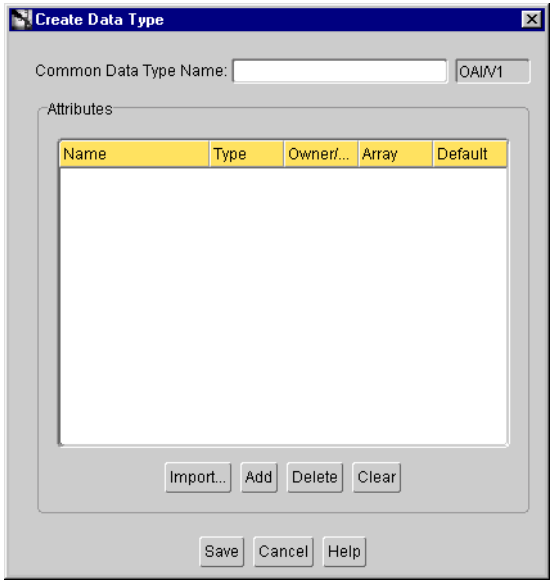


Figure 2–9 Create Data Type window

Enter common data type details as follows:

- **Common Data Type Name—**Data type name
- **OAI/V1**—the owner and version number of the Common Data Type. (It cannot be edited.)

NOTE: The first version is always V1. For subsequent versions, choose **Save As** to save it as different version number—V2, V3, V4. etc.

You can import attribute definitions from various sources. For example, you can import them from a pre-existing database table or an API Repository.

Refer to the section on Importing Attribute Information on page 16 to **Import**, **Add**, **Delete**, **Clear** the attributes.

Attributes

- **Name** —the attribute name.
- **Type** —may be integer, string, binary, float, double, or user-defined data type.

- **Array**—check this box if it is an array field. (Only user-defined data types can be of type array.)
- **Default** —the default value for the field is NULL.

Click **Save** to save and exit, **Save As** to save the Common Data Type as a different version, **Cancel** to exit without saving, or **Help** to display the help window.

Importing Attribute Information

Attributes can be imported from a database, SAP BAPI, SAP IDOC, and XML DTD. To import attributes from a database, perform these steps:

1. Click **Import** button to import the data types from a pre-existing database table.
2. Click **Database**. The database Login Info window is displayed: (Figure 2–10**)**

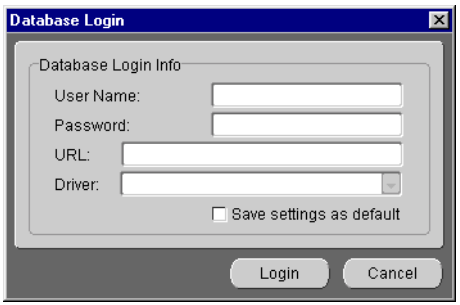


Figure 2–10 Database login info window for importing the common data types attributes

3. Enter the database log in information as follows:
   - **User Name** —the log in name
   - **Password** —the log in password
   - **URL** —in this form - **machine name: port number: database SID**
   - **Driver—**the JDBC driver being used to connect to the database.
   - **Save settings as default** —check this box to save the settings for the workspace.
4. After login, the database tables and arguments are displayed in the Database Browser Window. (Figure 2–11) Select **All** to import all the fields, or check the individual fields.
5. Click **Done** to import the attributes into the common data type.

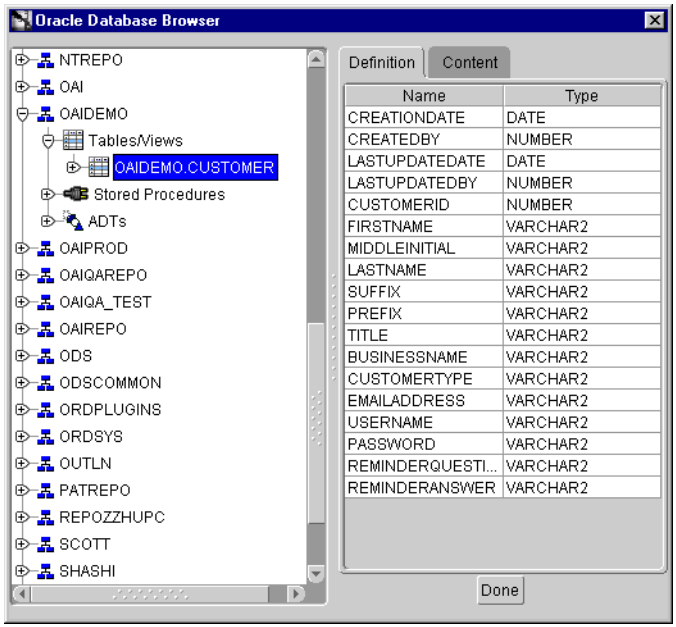


Figure 2–11 The Database Browser table to import attributes for common data type

6. If you select common view, all the arguments in the common view are copied over. After successful import, the arguments for the common data type is imported and populates the table. (Figure 2–12)

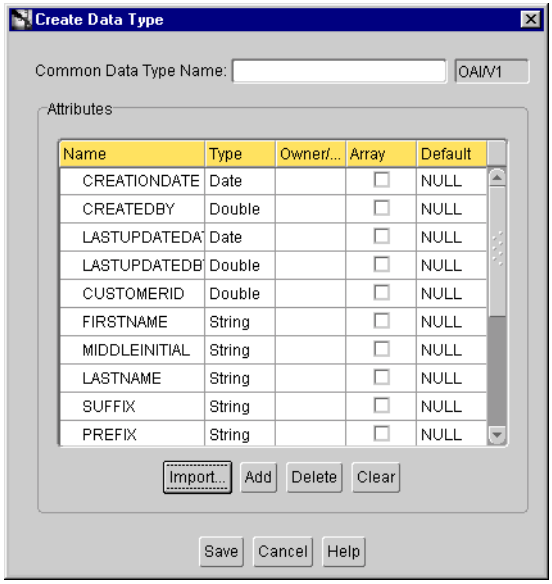


Figure 2–12 Common Data Type window with attributes

Click **Add** to add a new row of attributes, **Delete** to delete the selected attribute, or **Clear** to delete all attributes.

Editing Content Based Routing

To edit content based routing for an event or procedure, ^**right-click** the event or procedure under the **Content Based Routing** node in the iStudio design tree, then click **Edit**. The Content Based Routing Rules are displayed: (Figure 2–13)

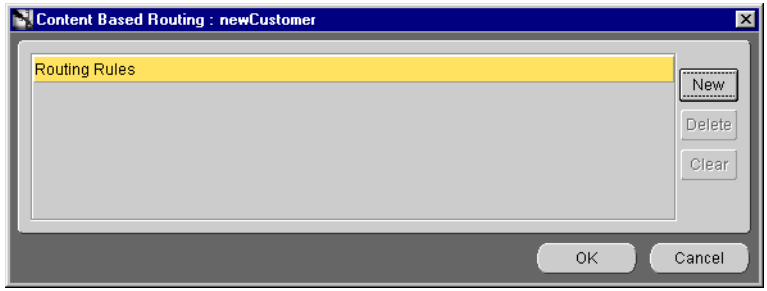


Figure 2–13 Content Based Routing Rules

Click **New** to display the Content Based Routing Wizard. Content Based Routing Wizard navigates you through these steps:

1. Choose a source event attribute.
2. Choose the operator.
3. Choose the attribute value.
4. Specify additional conditions.
5. Specify the destination applications.
6. Finish the Content Based Routing Wizard.

The steps are described in detail in the following sections.

Choose Source

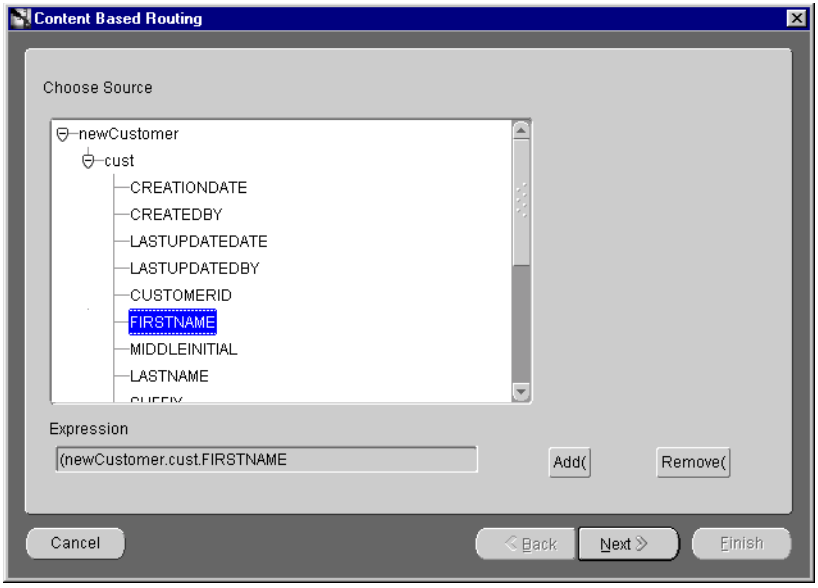


Figure 2–14 window 1 Content Based Routing Wizard window - Choose Source

- Select the source event attribute
- Click **Next** in the Wizard. The choose **Choose Operator** window is displayed (Figure 2–15)

Choose Operator

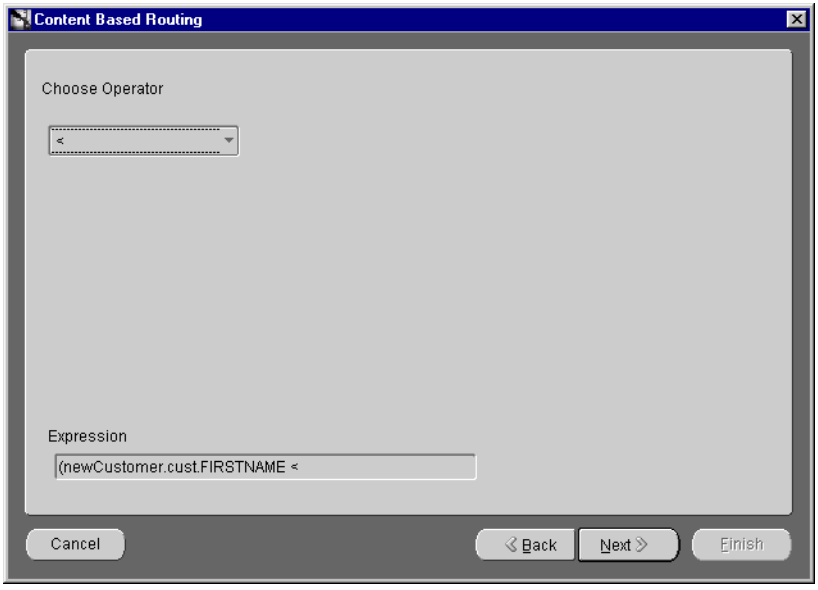


Figure 2–15 window 2 Content Based Routing Wizard window - Choose Operator

- Select the operator
- Click **Next** in the Wizard. The choose **Choose Value** window is displayed (Figure 2–16)

Choose Value

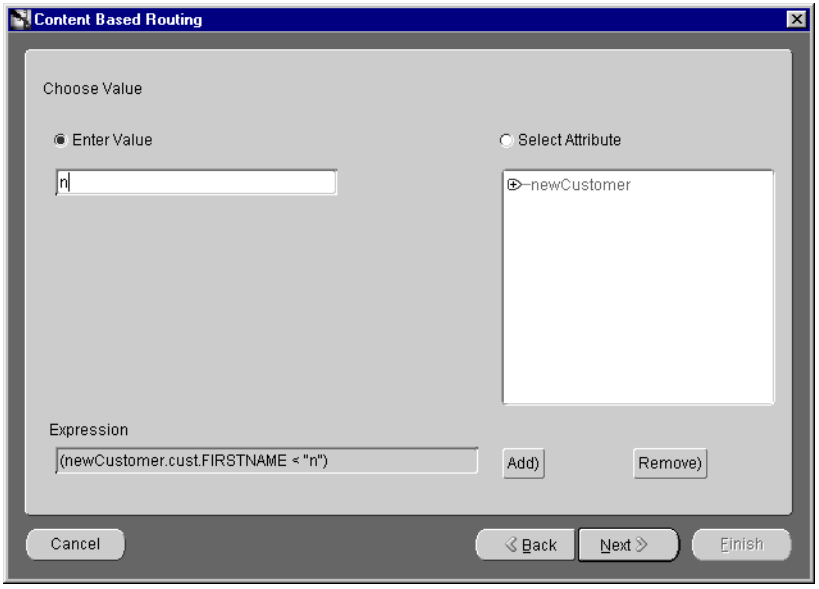


Figure 2–16 window 3 Content Based Routing Wizard window - Choose Value

- Enter the value in the text field **or** click **Select Attribute** radio button and select an attribute from the tree.
- Click **Next** in the Wizard. The **Additional Condition** window is displayed (Figure 2–17).

Additional Condition

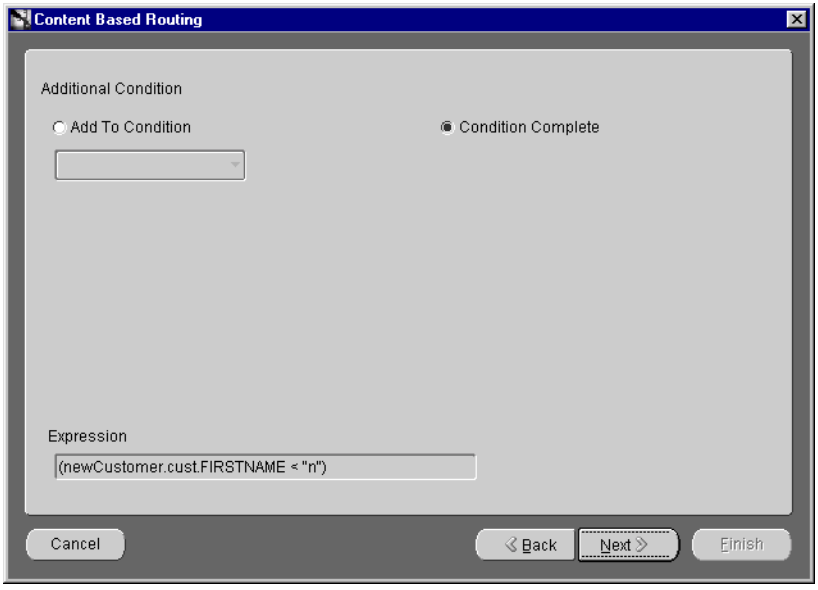


Figure 2–17 window 4 Content Based Routing Wizard window - Additional Condition

- Click **Add To Condition** radio button and select a boolean operator from combo box to add additional condition.
- Click **Next** in the Wizard. Depending on which radio button was selected either the **Choose Source** window or **Select Destination Application** window is displayed (Figure 2–18)

Select Destination Application

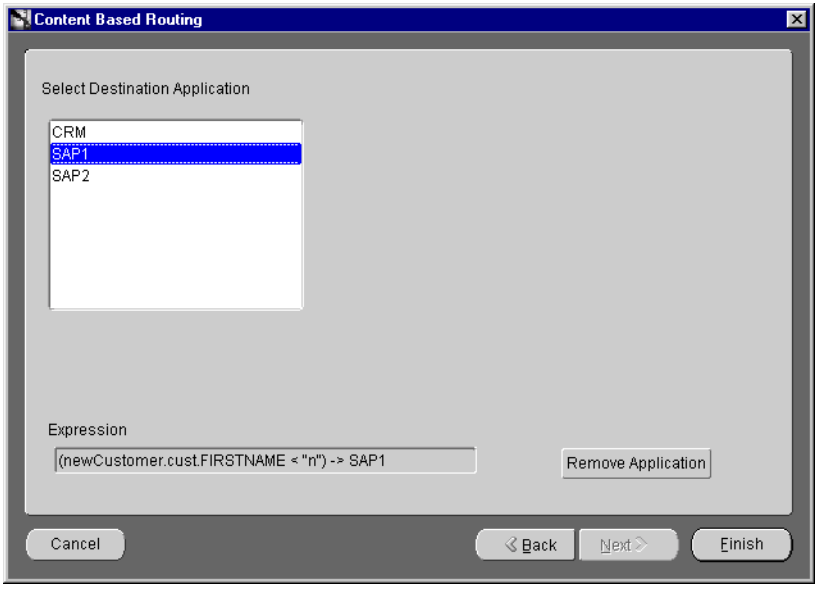


Figure 2–18 window 5 Content Based Routing Wizard window - Select Destination Application

- Select an application.
- Click **Finish** in the Wizard.

The Content Based Routing Rule is created (Figure 2–19).

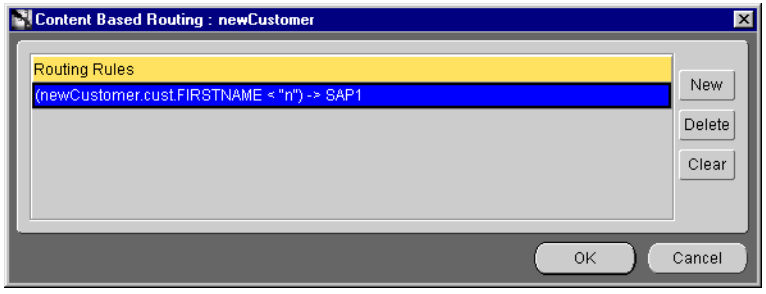


Figure 2–19 Content Based Routing Rules

Click **OK** to save Content Based Routing Rules and exit, or **Cancel** to exit without saving.

Creating Cross Reference Tables

To create a Cross Reference Table, click **File** /**New** in the pull-down menu, and select **Cross Reference Tables.** The Create X_Ref Table Dialog is displayed. (Figure 2–20)

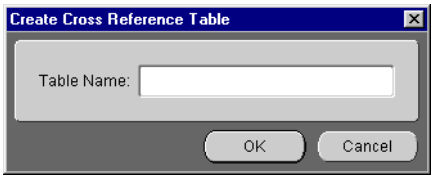


Figure 2–20 Create X_Ref Table Dialog

1. Enter the **Table Name**.
2. Click **OK** to save and exit, or **Cancel** to exit without saving.

Adding Applications to Cross Reference Tables

To add applications to the Cross Reference Table, select a Cross Reference Table in the project tree, right-click the Cross Reference Table, and click **Add App**. The Add Application to XRef Table window is displayed: **(**Figure 2–21**).**

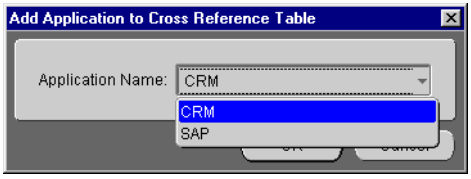


Figure 2–21 Add Application to XRef Table

1. Choose an Application Name from the list.
2. Click **OK** to add the application and exit, or **Cancel** to exit without saving.

Removing Applications From Cross Reference Tables

To remove applications from the Cross Reference Table, select a Cross Reference Table in the project tree, right-click the Cross Reference Table, and click **Remove App**. The Remove Application from XRef Table window is displayed: **(**Figure 2–22)

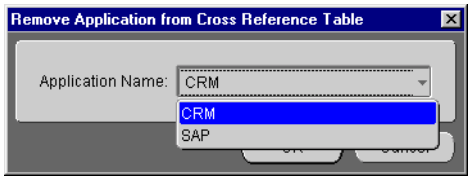


Figure 2–22 Remove Application from XRef Table

1. Select the application name to remove from the list.
2. Click **OK** to remove the application and exit, or **Cancel** to exit without saving.

Creating Domain Value Mappings

To create a Domain Value Mappings table, click **File** in the iStudio main menu panel, click **New** in the pull-down menu, and select **Domain Value Mapping**. The Create DVM Table Dialog is displayed. (Figure 2–23)

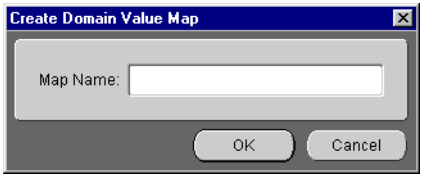


Figure 2–23 Create DVM Table

Adding Applications to Domain Value Mappings

To add applications to Domain Value Mappings, select a Domain Value Mapping in the project tree, right-click the Domain Value Mapping, click **Add App**. The Add Application to DVM Table dialog is displayed**. (**Figure 2–24**)**

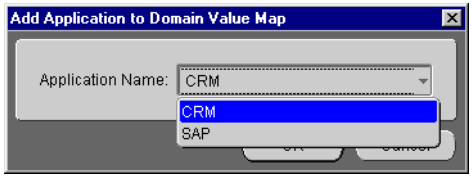


Figure 2–24 Add Application to DVM Table

1. Choose an Application Name from the list.
2. Click **OK** to add the application and exit, or **Cancel** to exit without saving.

Removing Applications From Domain Value Mappings

To remove applications from the Domain Value Mappings, select a Domain Value Mapping in the project tree, right-click **Domain Value Mapping**, and click **Remove App**. The Remove Application from Domain Value Mapping dialog is displayed**.**

In the Domain Value Mapping dialog, perform these two steps:

1. Choose the Application Name to remove from the list.
2. Click **OK** to remove the application and exit, or **Cancel** to exit without saving.

Editing Data in Domain Value Mappings

To add data to Domain Value Mappings, select a Domain Value Mapping in the project tree, right-click Domain Value Mapping, and click **Edit Domain Value Map.** The Edit Domain Value Map Dialog is displayed: **(**Figure 2–25**)**

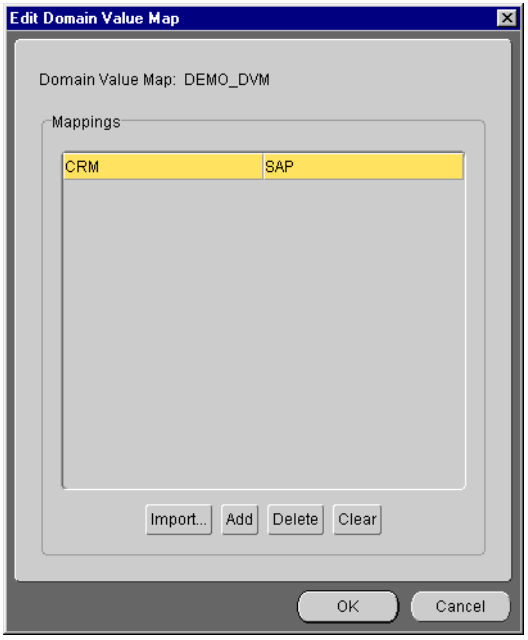


Figure 2–25 Edit DVM Dialog

Enter the mapping values and click **Add**.

To delete or clear the values in the mappings table, select a Domain Value Mapping, and click **Delete.**

To delete the DVM table, right-click the Domain Value Mapping, and click **Delete.** The DVM table is deleted**.**

Creating an Event

To create an event, click **Project (your project name)** in the iStudio main menu panel, click **New** in the pull-down menu, and select **Event**. The Create Event window is displayed: (Figure 2–26)

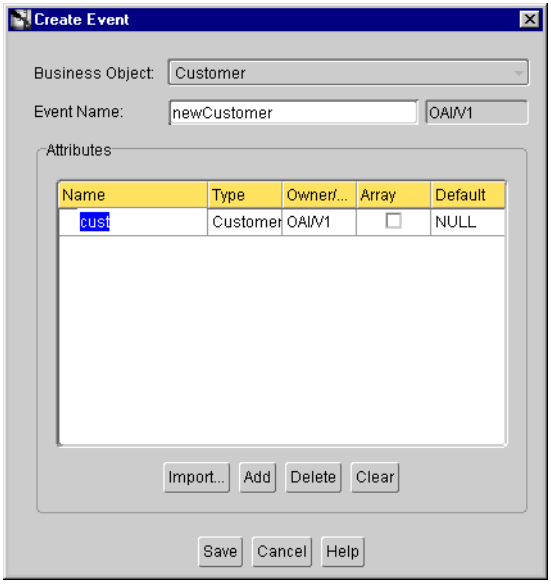


Figure 2–26 Create Event window and Common Data type selection for an attribute

Enter event details as follows:

- **Business Object Name**—the name of the category to which the event belongs to.
- **Event Name**—the event name. Only Alphanumeric characters are allowed.
- **OAI/V1**—the owner and version number of the Business Object.

Attributes

- **Name**—the attribute name.
- **Type** —Can be an integer, string, binary, float, double, data type. If you choose data type, you can select the common data type which have been defined already.
- **Array**—check this box if it is an array field. Only user-defined data types can be of type array.
- **Default** —default value for the field is NULL.

- Refer to the section on Importing Attribute Information on page 16 on how to **Import**, **Add**, **Delete**, **Clear** the attributes.

- Click **Save** to save and exit, **Save As** to save the Business object as a different version, or **Cancel** to exit without saving, or **Help** for help window.

Creating a Procedure

To create a procedure, click **File** in the iStudio main menu panel, click **New** in the pull-down menu, and select **Procedure**. The Create Procedure window is displayed: (Figure 2–27)

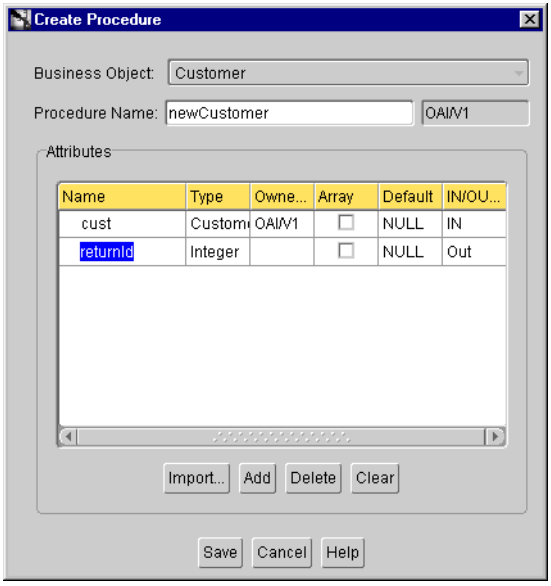


Figure 2–27 Create Procedure window with the arguments fields imported

Enter the procedure details as follows:

- **Business Object Name**—the name of the category to which the procedure belongs to.
- **Procedure Name**—the procedure name. Only Alphanumeric characters are allowed.
- **OAI/V1**—the owner and version number of the procedure.

Arguments

- **Name**—the argument name.
- **Type** —integer, string, binary, float, or double data type. If you choose data type, you can select the common data type which has been defined already.

- **Array**—check this box if it is an array field. Only user-defined data types can be of type array.
- **Default** —default value for the field is NULL.
- **IN/OUT/INOUT**—IN - input parameter, OUT- output parameter, INOUT - input and return parameter.

- Refer to the section on Importing Attribute Information on page 16 on how to **Import**, **Add**, **Delete**, **Clear** the attributes.
- Click **Save** to save and exit, **Save As** to save the procedure as a different version, **Cancel** to exit without saving, or **Help** for help window.

Creating a Publish Event

To create a publish event in an application, click **Event** in the iStudio main menu panel, then click **Publish** in the pull-down menu. The Publish Wizard window is displayed: (Figure 2–28)

The Event Publish Wizard navigates you through these four steps:

1. Select an event.
2. Specify the application view.
3. Map and transform the common objects.
4. Finish the event publish.

The steps are described in detail in the following sections.

Select an Event

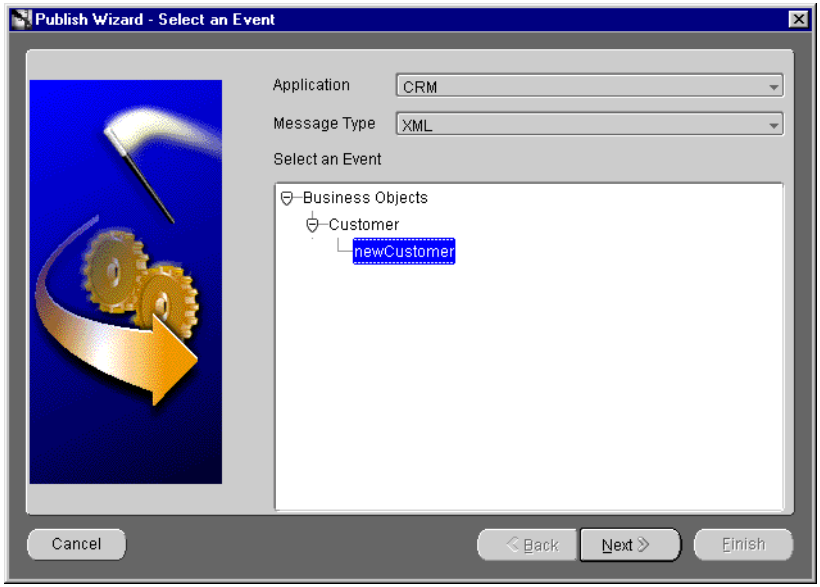


Figure 2–28 window 1 Publish Wizard window - Select an Event

Enter the event details as follows:

- **Application**—the name of the application which is publishing the event.
- **Message Type (e.g. XML)**—this specifies the mode of communication between the Adapter and the application.

  You can select from the following Message Types:

  Database—the Adapter picks the message data from the database.

  SAP BAPI—the Adapter communicates with the application using BAPI.

  SAP IBP—the Adapter communicates with the application using IBP.

  SAP IDOC—the Adapter communicates with SAP using IDOC.

  XML—the Adapter communicates with the application using XML.

  Generic—the Adapter communicates with application using a user defined bridge.
- Select the event name.
- Click **Next** in the Publish Wizard. The **Define Application View** window is displayed: (Figure 2–29)

Specify the Application View

After selecting the event to publish, you define the application view. The application view window is initially an empty table. You may define the attributes using the **add** button, or importing the definitions from a database or an API Repository.

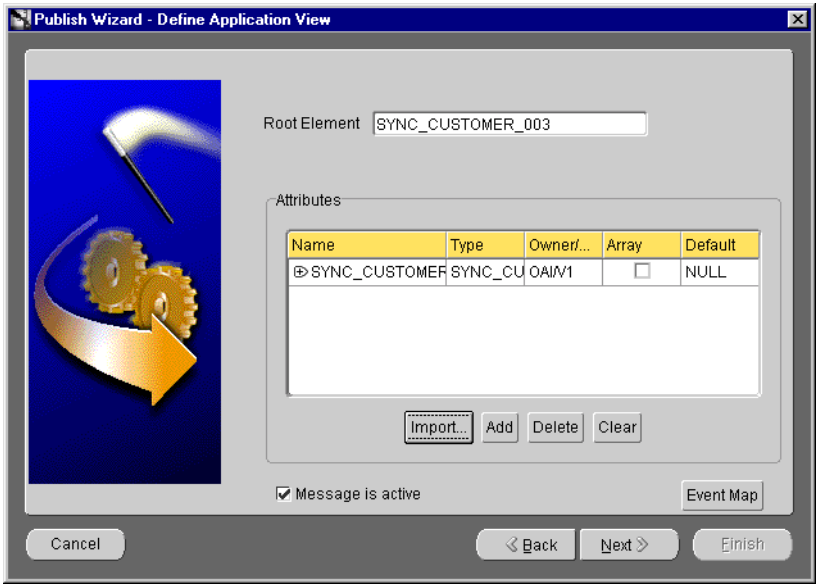


Figure 2–29 window 2 Publish Wizard - Define Application View

Enter these attribute details:

Attributes

- **Name**—the attribute name.
- **Type**—may be an integer, string, binary, float, or double data type.
- **Owner/Version**—the owner and version of this application view.
- **Array**—check this box if it is an array field. Only user-defined data types can be of type array.
- **Default**—default value for the field is NULL.

Refer to the section on Importing Attribute Information on page 16 to **Import**, **Add**, **Delete**, **Clear** attributes. If you are importing a XML DTD, a file dialog is displayed: (Figure 2–30)

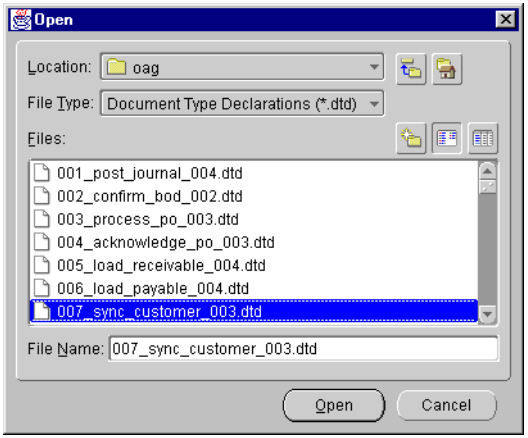


Figure 2–30 DTD file dialog

- Select the DTD file and click **Open** to confirm, or **Cancel** to exit.
- The **Choose Root Element** dialog is displayed (Figure 2–31).

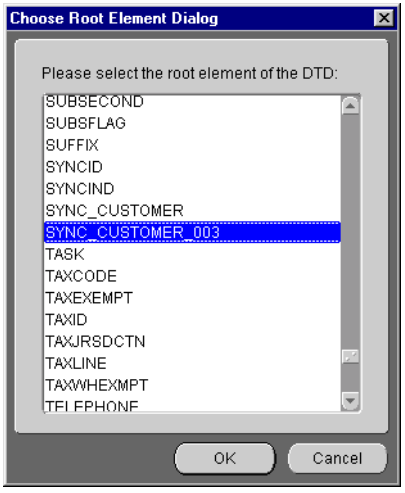


Figure 2–31 Choose Root Element dialog

- Select the root DTD element and click **OK** to confirm, or **Cancel** to exit.

If this is a XML type message, the **Event Map** button is visible. Click it to define the event map. The **Event Map** dialog is displayed (Figure 2–32).

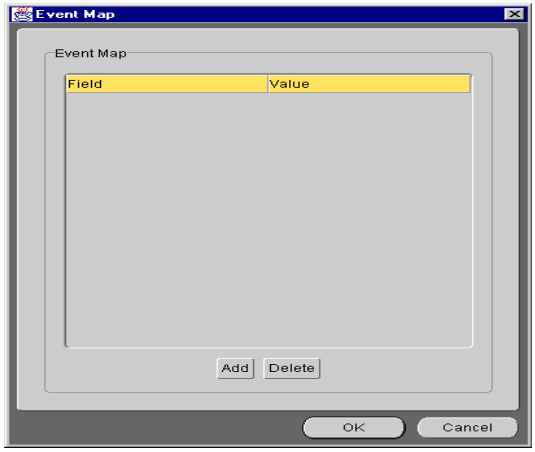


Figure 2–32 Event Map dialog

1. Click **Add** to add an event map attribute. The **New Event Map** dialog is displayed (Figure 2–33).

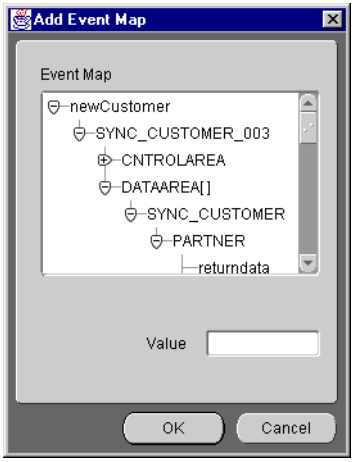


Figure 2–33 New Event Map dialog

- Choose an attribute from the tree.

- Enter a value for it in the text field.
- Click **OK** to confirm and exit the **New Event Map** dialog or **Cancel** to exit.

2. To delete an event map item, select it and click **Delete**.
3. Click **OK** to confirm and exit the **Event Map** dialog or **Cancel** to exit.

Click **Next** in the Publish Wizard. The **Define Mapping** window is displayed (Figure 2–34).

Map and Transform

Mapping can either involve copying the individual fields, or simple shape-change transformations.

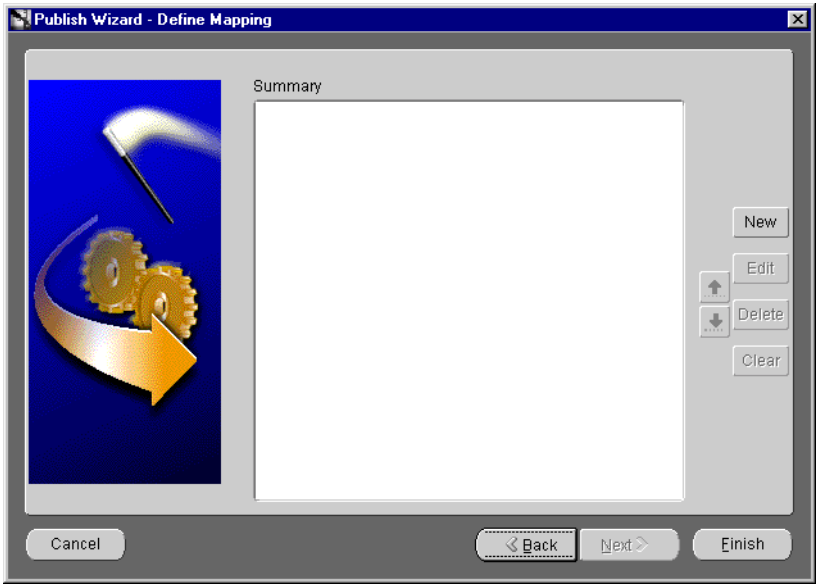


Figure 2–34 window 3 Publish Wizard - Define Mapping of Application View to Common View

- Click **New** to define mappings. See New Mapping on page 2-37 to map attributes.
- Click **Finish**.

The publish event is now created.

New Mapping

Click the **New** button to define a mapping. The **Mapping** dialog is displayed (Figure 2–35).

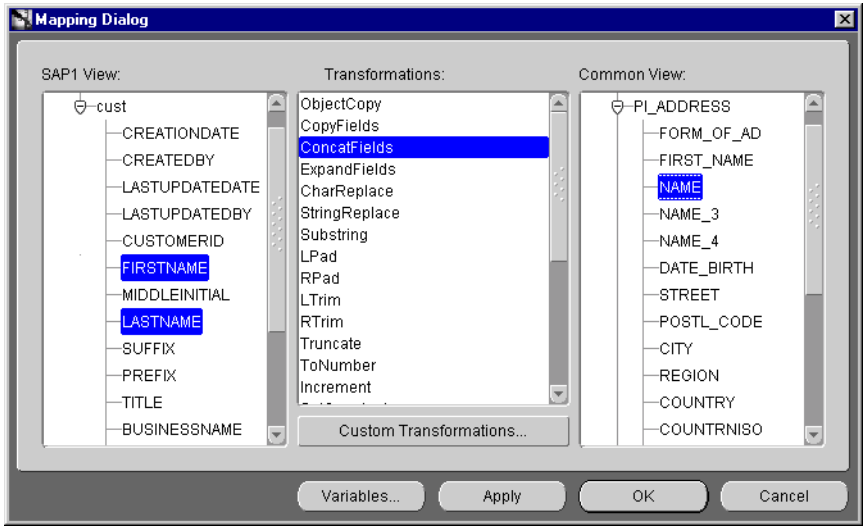


Figure 2–35 Mapping dialog

For example, to map fields `FirstName` and `LastName` in the application view to `Name` in the common view, use the `concat` transform.

The following steps illustrate this example:

1. Select fields to map from in the application view. **^left-click** the mouse to select multiple fields in a view.
2. Select the transformation to perform. (For example, ConcatFields). See Custom Transformations on page 2-38 to add or delete custom transformations.
3. Select fields to map to in the common view. **^left-click** the mouse to select multiple fields in a view.
4. Click **Apply** to confirm selection and continue specifying additional mapping, **OK** to confirm selection and exit, or **Cancel** to exit.
5. On clicking **Apply** or **OK**, if the transformation has parameters then the **Mapping Parameters** dialog is displayed (Figure 2–36). Enter the following information:
   - **Parameters**—Enter values for the transformation parameters. (For example, (blank) value for separator parameter.)

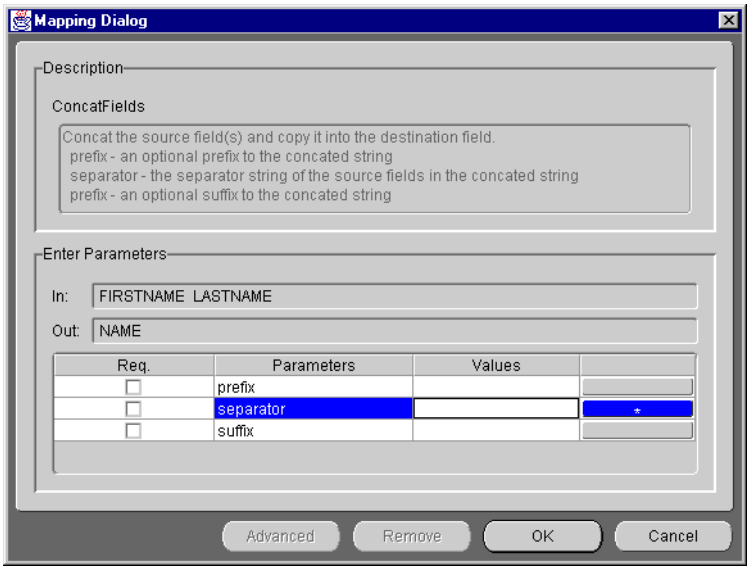


Figure 2–36 Mapping Parameters dialog

6. Click **OK** to confirm and exit or **Cancel** to exit.

Delete Attribute Mappings

1. Click **Delete** to delete a mapping, or **Clear** to clear the all mappings.

Edit Attribute Mapping

1. Select a mapping.
2. Click **Edit** to modify the mapping.

Custom Transformations

Click **Custom Transformations** in the **Mapping** dialog. The **Transformations** dialog is displayed (Figure 2–37).

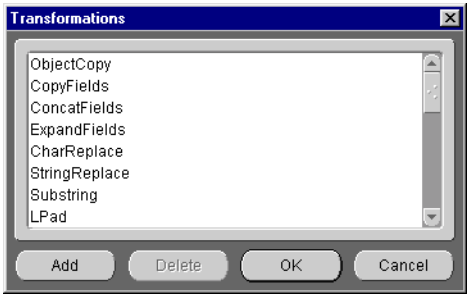


Figure 2–37 Transformations dialog

- Click **Add** to add user defined transformation or select a user defined transformation and click **Delete** to delete it.
- Click **OK** to confirm and exit or **Cancel** to exit.

Variables

Click **Variables** in the **Mapping** dialog. The **Variables** dialog is displayed (Figure 2–38).

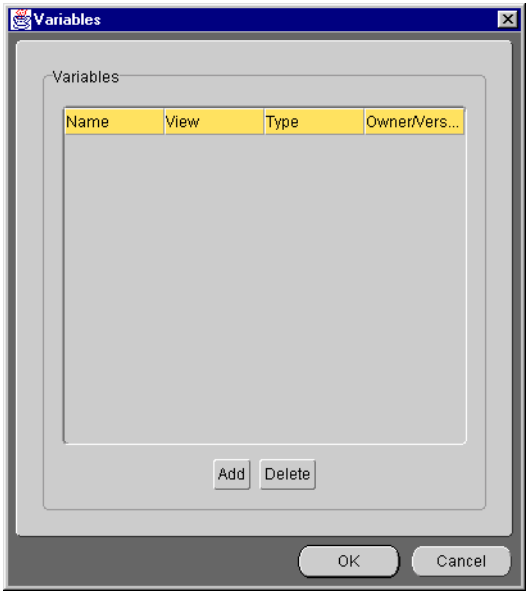


Figure 2–38 Variables dialog

- Click **Add** to add a variable. The **New Variable** dialog is displayed (Figure 2–39).

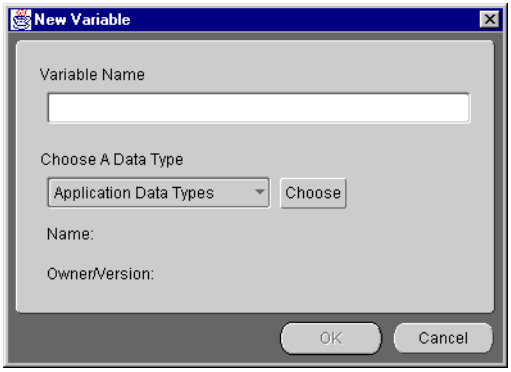


Figure 2–39 New Variable dialog

  - Choose a data type from the combo box. The data type for the variable must be previously defined.
  - Click **Choose**, select the datatype from the displayed list, and click **OK** to confirm and exit the **Data Types** dialog or **Cancel** to exit.
  - Click **OK** to confirm and exit the **New Variable** dialog or **Cancel** to exit.
- To delete a variable, select it and click **Delete**.
- Click **OK** to confirm and exit the **Variables** dialog or **Cancel** to exit.

Creating a Subscribe Event

To create a subscribe event in an application, click **Event** in the iStudio main menu panel, then click **Subscribe** in the pull-down menu. The Subscribe Wizard is displayed: (Figure 2–40)

The Event Subscribe Wizard navigates you through four steps:

1. Select an event.
2. Specify the application view.
3. Map and transform the common objects.
4. Finish the event subscription.

The steps are described in detail in the following sections.

Select an Event

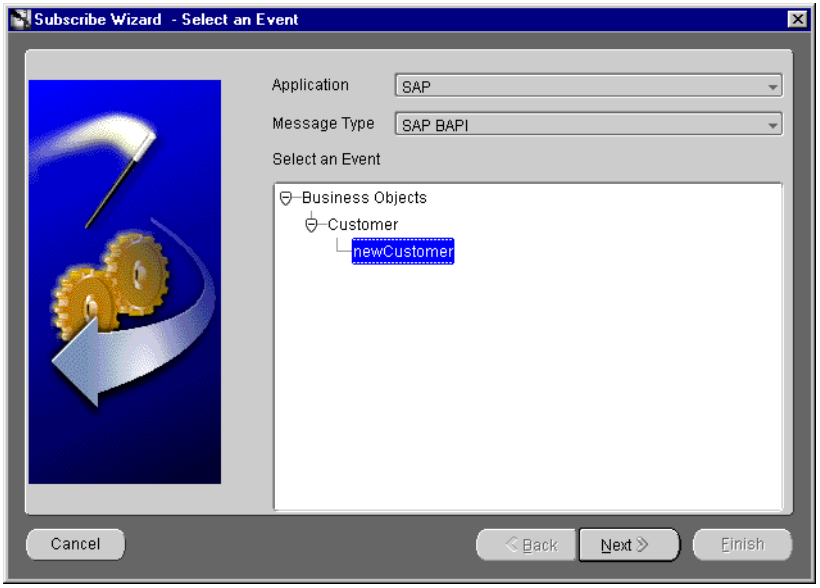


Figure 2–40 window 1 Subscribe Wizard - Select an Event

Enter the event details as follows:

- **Application**—the application name which is subscribing to the event.
- **Message Type (e.g.SAP BAPI)**—this specifies the mode of communication between the Adapter and the application.

  You can select from the following Message Types:

  Database—the Adapter picks the message data from the database.

  SAP BAPI—the Adapter communicates with the application using BAPI.

  SAP IBP—the Adapter communicates with the application using IBP.

  SAP IDOC—the Adapter communicates with SAP using IDOC.

  XML—the Adapter communicates with the application using XML. (See Publish Message, figure 2-28).

  Generic—the Adapter communicates with application using a user defined bridge.
- Select the event to subscribe.

- Click **Next** in the Subscribe Wizard to display the Define Application View window: (Figure 2–41)

Specify the Application View

After selecting the event to publish, you define the application view. The application view window is initially an empty table. You may define the attributes by using the **add** button, or importing the definitions from a database or an API Repository.

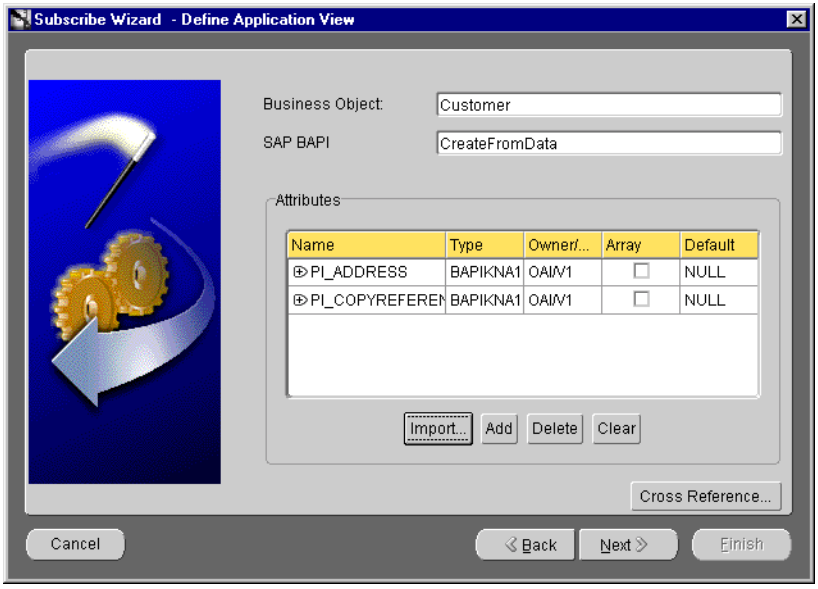


Figure 2–41 window 2 of Subscribe Wizard Event - Define Application View

Enter the application view details:

- **Business Object**—name of the business object.
- **SAP BAPI**—the SAP BAPI name which is imported.

Attributes

- **Name**—the attribute name.
- **Type**—may be an integer, string, binary, float, or double data type.
- **Owner/Version**—the owner and version number of this application view.

- **Array**—check this box if it is an array field. Only user-defined data types can be of type array.
- **Default**—default value for the field is NULL.

Refer to the section Importing Attribute Information on page 16 to **Import**, **Add**, **Delete**, **Clear** the attributes.

Click **Cross Reference...**, and the **Cross Reference** dialog is displayed. To populate and look up Cross Reference Table, refer to the section Populating Cross Reference Tables on page 2-44.

Click **Next** in the Subscribe Wizard to display the Mapping and Transformation window (Figure 2–42).

Map and Transform

Mapping can either involve copying the individual fields or simple shape change transformations.

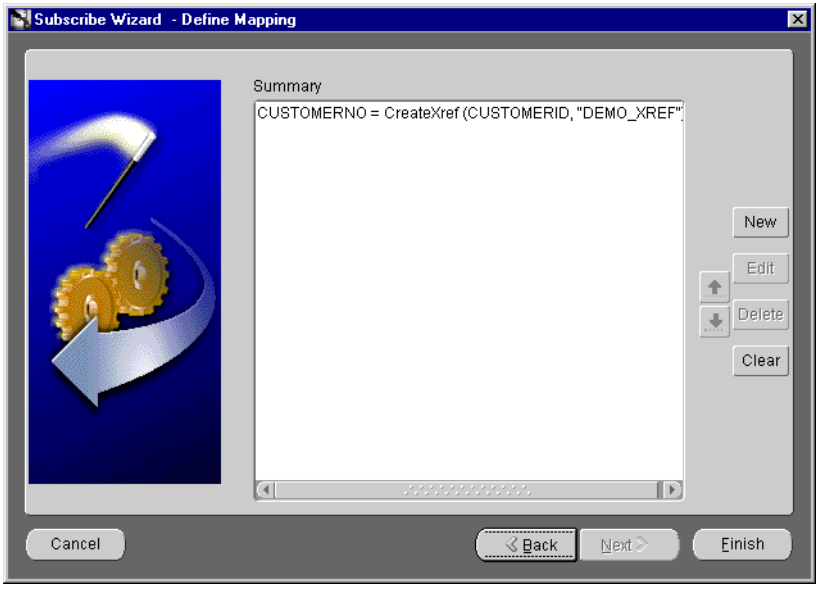


Figure 2–42 window 3 Subscribe Wizard - Define Mapping of Common View to Application View

- Click **New** to define mappings. See New Mapping on page 2-37 to map attributes.

- Click **Finish**.

The subscribe event has now been created.

Populating Cross Reference Tables

To populate the Cross Reference Tables, you define the **Returned Arguments** which are the arguments returned by subscribe/implement code for populating the Cross Reference Table.

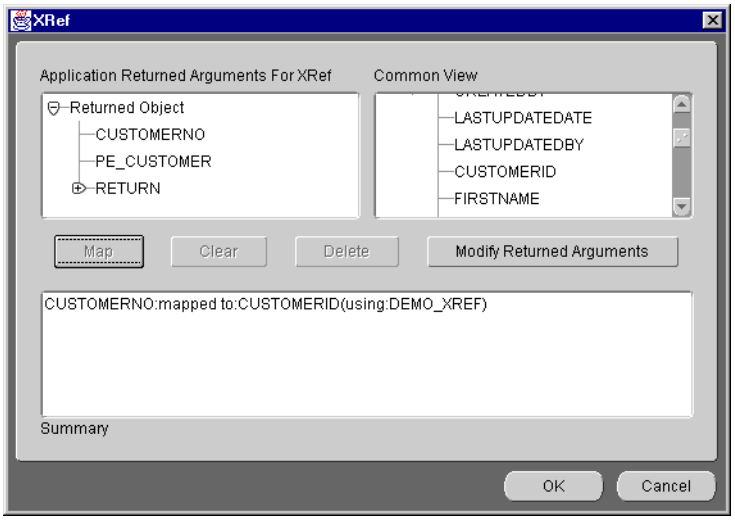


Figure 2–43 Cross Reference dialog

- The Application Returned Arguments window displays the returned arguments. It is initially populated with any OUT arguments from the application view.
- You can modify the returned arguments list by clicking **Modify Returned Arguments**.
- Select the corresponding attributes in the Application Returned Arguments and Common View windows and click **Map**.
- Specify the Cross Reference Table name to be populated using these attributes' values.

Creating an Invoking Procedure

To create an invoking procedure in an application, click **Procedure** in the iStudio main menu panel, then click **Invoke** in the pull-down menu. The Invoke Wizard is displayed: (Figure 2–44)

The Invoke Procedure Wizard navigates you through these steps:

1. Select a procedure.
2. Specify the application view.
3. Map and transform the application view and the common view.
4. Map and transform the common view and the application view.
5. Specify the stored procedure for database type messages.
6. Finish the procedure invocation.

The steps are described in detail in the following sections.

Select a Procedure

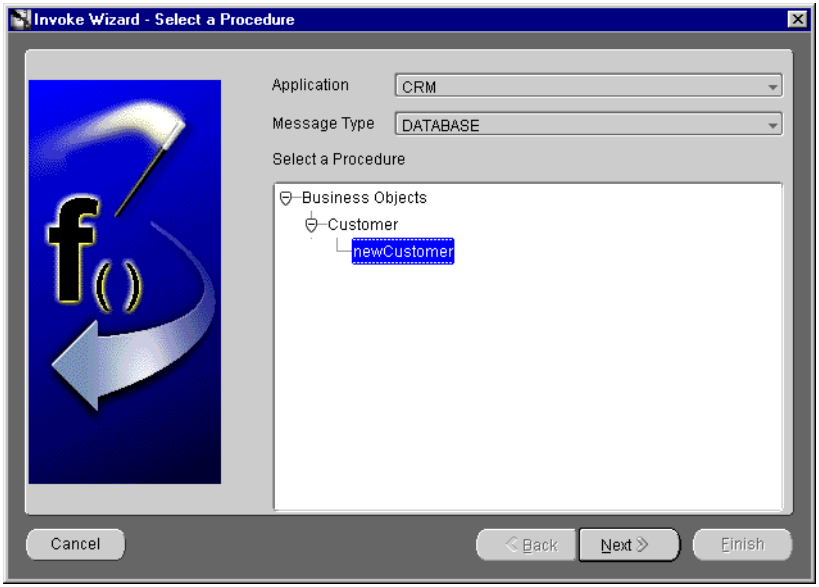


Figure 2–44 window 1 Invoke Wizard - Select a Procedure

Enter these procedure details:

- **Application (e.g.iStore)**—application name that is invoking the procedure.
- **Message Type**—specifies the mode of communication between the Adapter and the application.

You can select from the following Message Types:

Database—the Adapter picks the message data from the database.

SAP BAPI—the Adapter communicates with the application using BAPI.

SAP IBP—the Adapter communicates with the application using IBP.

SAP IDOC—the Adapter communicates with SAP using IDOC.

XML—the Adapter communicates with the application using XML.

Generic—the Adapter communicates with application using a user defined bridge.

Note: In this release of Applications InterConnect, to invoke a procedure, the message type can be a database, XML, or Generic only.

- Select the procedure to invoke.
- Click **Next** in the Invoke Wizard to display the Define Application View window: (Figure 2–45)

Specifying the Application View

After selecting the procedure to invoke, you define the application view. The application view window is initially an empty table. You may define the attributes by using the add button, or importing the definitions from a database or an API Repository.

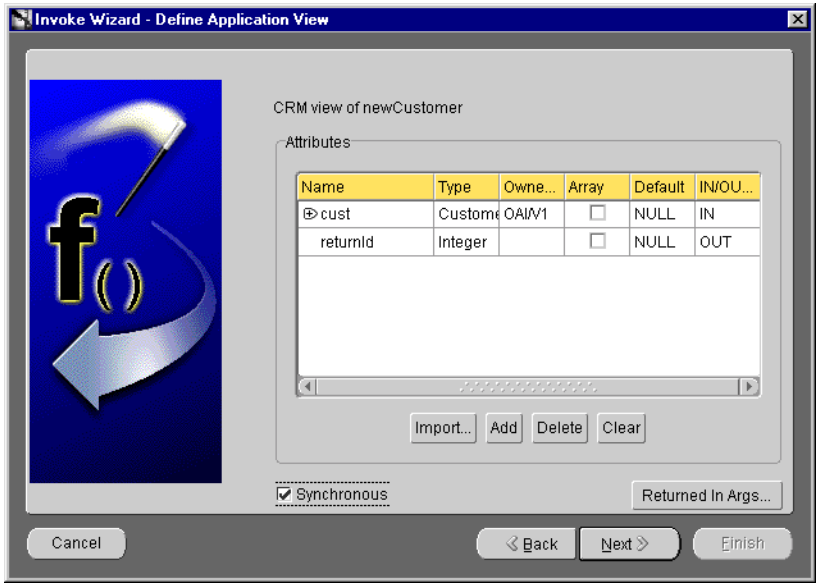


Figure 2–45 window 2 Invoke Wizard - Define Application View

Enter these application view details:

Attributes

- **Name**—the attribute name.
- **Type**—integer, string, binary, float, or double data type.
- **Owner/Version**—the owner and version of this application view.
- **Array**—check this box if it is an array field. Only user-defined data types can be of type array.
- **Default**—default value for the field is NULL.

Refer to the section Importing Attribute Information on page 16 to **Import**, **Add**, **Delete**, **Clear** the attributes.

- **Synchronous** - check this box if this is a synchronous invoke i.e. the request will block till it gets a reply.
- If you need to specify the IN arguments which are to be returned, click **Returned In Args**. The **Select Returned In Arguments** window is displayed: (Figure 2–46)

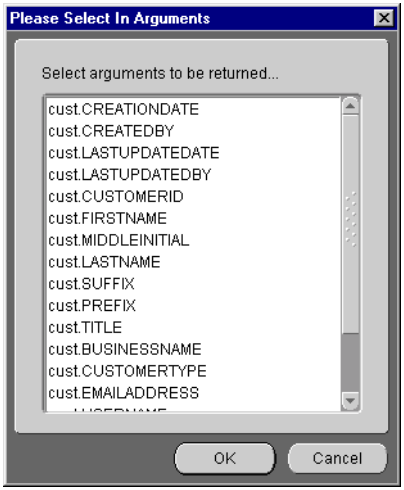


Figure 2–46 Add Returned IN arguments

- Select the input arguments to be returned. ^**left-click** the mouse to select multiple arguments. Only non user-defined input arguments are shown for selection. Click **OK** the **Select Returned Out Arguments** window is displayed, or **Cancel** to exit.
- Select the output arguments to be returned. ^**left-click** the mouse to select multiple arguments. Only non user-defined output arguments are shown for selection. Click **OK** or **Cancel** to exit.

- Click **Next** in the Invoke Wizard to display the Mapping and Transformation window: (Figure 2–47)

Map and Transform (Application View to Common View)

Mapping can either involve copying the individual fields, or simple shape-change transformations.

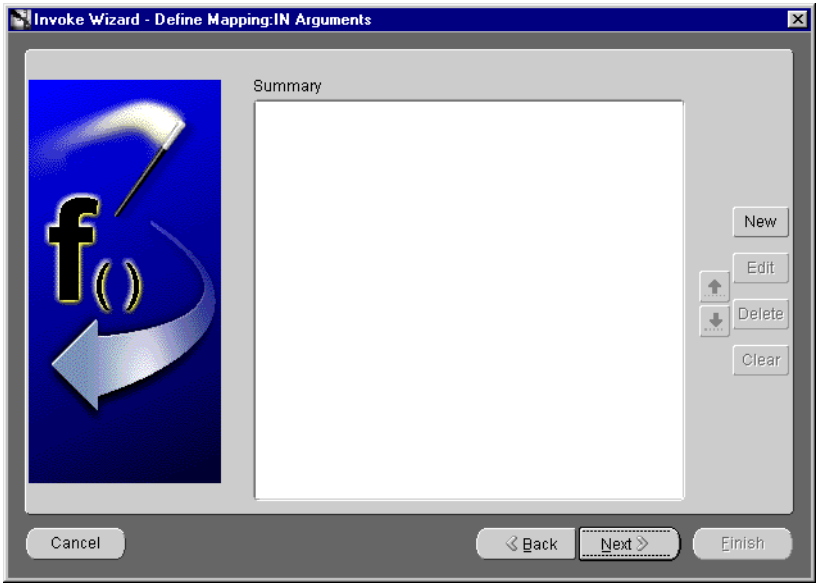


Figure 2–47 window 3 Invoke Wizard - Define Mapping of Application View to Common View

- Click **New** to define mappings. See New Mapping on page 2-37 to map attributes.
- Click **Next** in the Invoke Wizard to display the Mapping and Transformation (Common View to Application View) window: (Figure 2–48)

Map and Transform (Common View to Application View)

Mapping can either involve copying the individual fields, or simple shape-change transformations. You map the common view return arguments to the application view return arguments in this step.

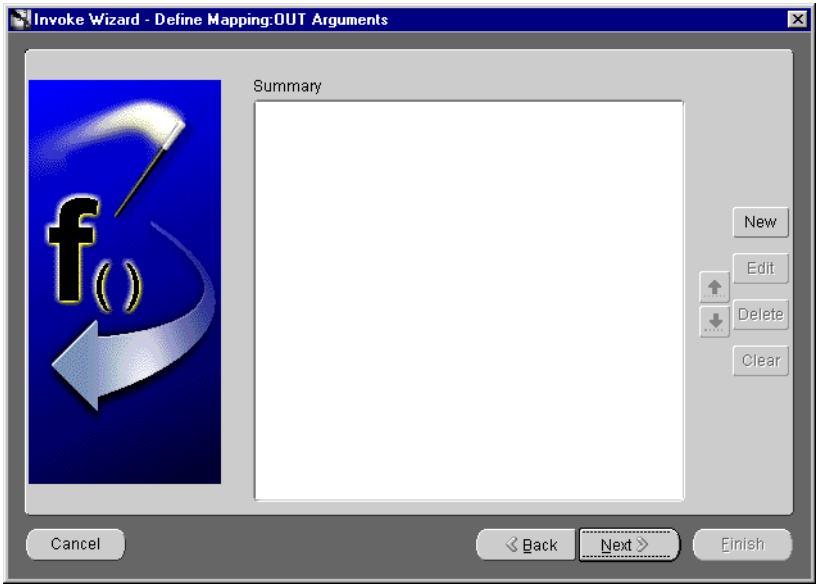


Figure 2–48 window 4 Invoke Wizard - Define Mapping of Common View to Application View

- Click **New** to define mappings. See New Mapping on page 2-37 to map attributes.
- Click **Next** to display the Specify Stored Procedure window: (Figure 2–49)

Specifying Stored Procedures for Database Message Type

If you selected the Message Type to be a database, the data will be received by a stored procedure. In this stored procedure, you can specify the action to be performed when the values are returned to the application. The Adapter will invoke the stored procedure at runtime with the appropriate data.

The following arguments will be returned:

- all the **OUT** arguments.
- all the **IN** arguments which were specified to be returned as part of the reply.

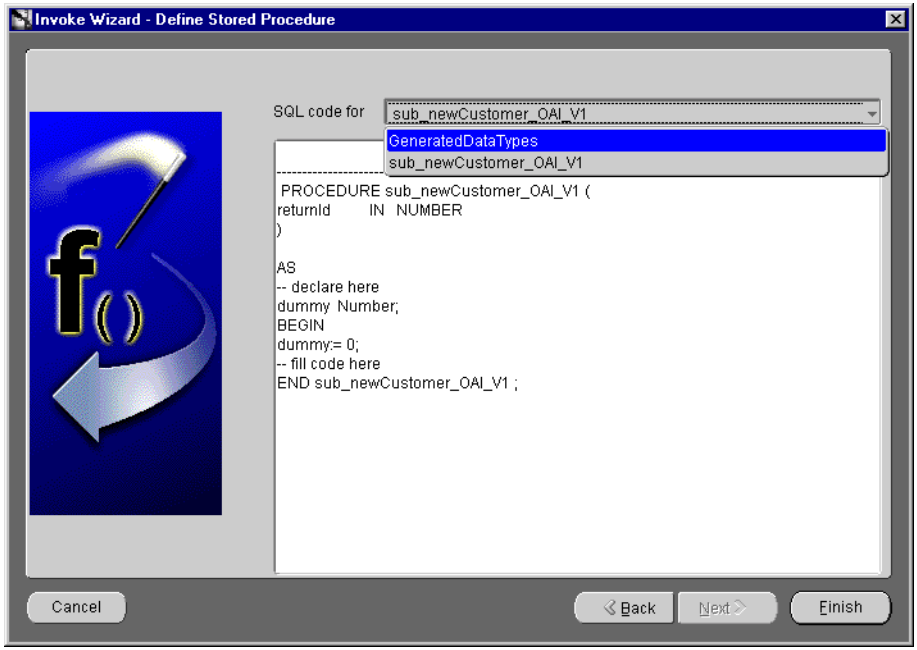


Figure 2–49 window 5 Invoke Wizard - Define Stored Procedure

Enter the stored procedure details as follows:

- **SQL Code—**select from a list of system generated procedures.
- Add new procedure code and click **Finish**.

The procedure invoke message is created.

Creating an Implementing Procedure

To implement a procedure, select **Procedure** in the iStudio main menu panel, and click **Implement** in the pull-down menu. The Implement Wizard is displayed: (Figure 2–50)

The Implement Procedure Wizard navigates you through these steps:

1. Select a procedure.
2. Specify the application view.
3. Map and transform the application view and the common view.
4. Map and transform the common view and the application view.

5. Finish the procedure implementation.

The steps are described in detail in the following sections.

Select a Procedure

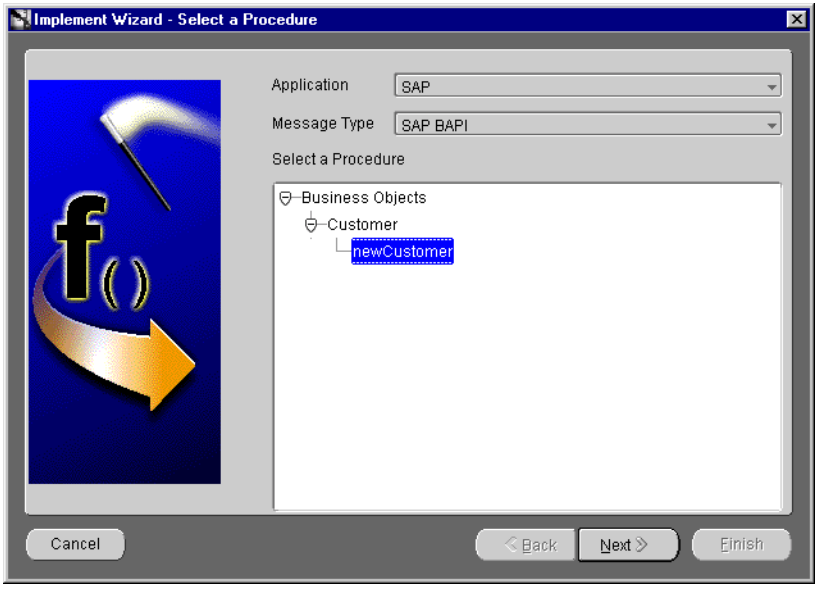


Figure 2–50 window 1 Implement Wizard - Select a Procedure

Enter these procedure details:

- **Application (e.g.SAP)**—the application name which is implementing the procedure. The example shown here is SAP.
- **Message Type**—specifies the mode of communication between the Adapter and the application.

  You can select from the following Message Types:

  Database—the Adapter retrieves the message data from the database.

  SAP BAPI—the Adapter communicates with the application using BAPI.

  SAP IBP—the Adapter communicates with the application using IBP.

  SAP IDOC—the Adapter communicates with SAP using IDOC.

  XML—the Adapter communicates with the application using XML.

Generic—the Adapter communicates with application using a user defined bridge.

- Select the procedure to invoke.
- Click **Next** in the Implement Wizard to display the Define Application View window: (Figure 2–51)

Specifying the Application View

After selecting the procedure to implement, you define the application view. The window is initially an empty table. You may define the attributes by using the add button, or importing the definitions from a database or an API Repository.

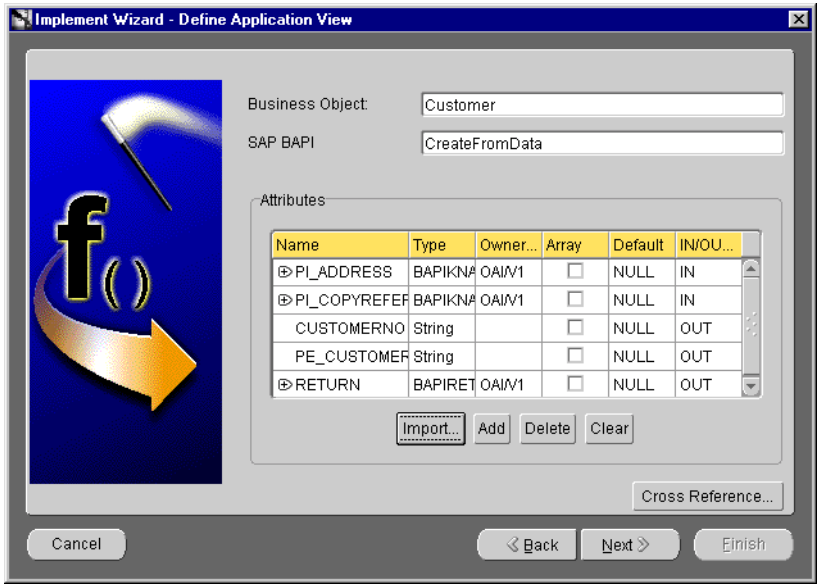


Figure 2–51 window 2 Implement Wizard - Define Application View

Enter these application view details:

- **Business Object**—name of the business object.
- **SAP BAPI**—the SAP BAPI name which is imported.

Attributes

- **Name**—the attribute name.

- **Type**—integer, string, binary, float, or double data type.
- **Owner/Version**—the owner and version of this application view.
- **Array**—check this box if it is an array field. Only user-defined data types can be of type array.
- **Default**—default value for the field is NULL.

Refer to the section Importing Attribute Information on page 16 to **Import**, **Add**, **Delete**, **Clear** the attributes. If you are importing SAP BAPI information the SAP log in window is displayed: (Figure 2–52)

Click **Cross Reference...**, and the **Cross Reference** dialog is displayed. To populate and look up Cross Reference Table, refer to the section Populating Cross Reference Tables on page 2-44

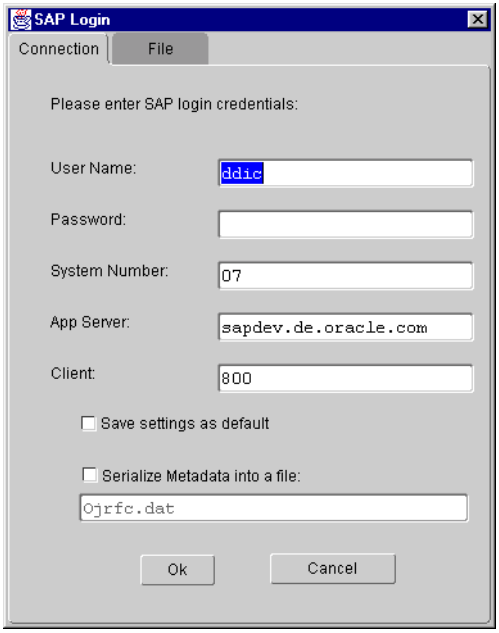


Figure 2–52 SAP login window

Enter the SAP log in information as follows:

- **User Name**—the SAP log in name
- **Password**—the password for this username

- **System Number**—the system you are logging in to
- **Application Server**—the server name
- **Client**—the client host name
- **Save settings as default**—check to save workspace settings.
- **Serialize Metadata**—check to save SAP metadata in a file.
- **Filename**—the name of the metadata file, as a fully-qualified path.
- Click **OK**. The SAP application browser window is displayed: (Figure 2–53)

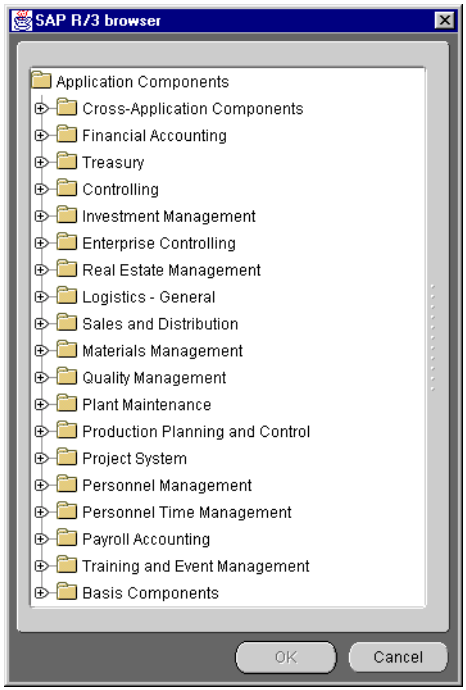


Figure 2–53 SAP R/3 browser window

- Select the SAP BAPI and click **OK.** The SAP BAPI attributes will be imported into the application view.
- Click **Next** to display the Mapping and Transformation window: (Figure 2–54)

Map and Transform

Mapping may involve copying individual fields, or simple shape-change transformations.

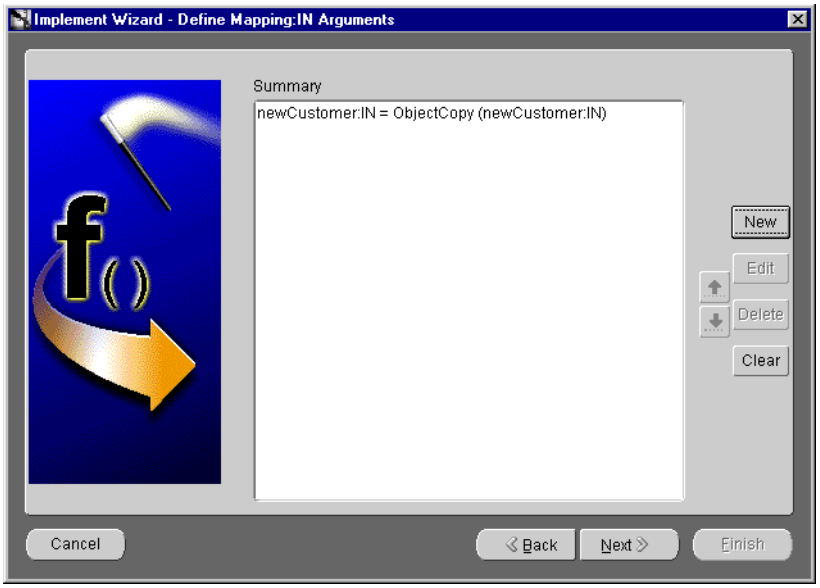


Figure 2–54 window 3 Implement Wizard - Define Mapping of Common View to Application View

- Click **New** to define mappings. See New Mapping on page 2-37 to map attributes.
- Click **Next** in the Implement Wizard to display the second Mapping and Transformation window: (Figure 2–55)

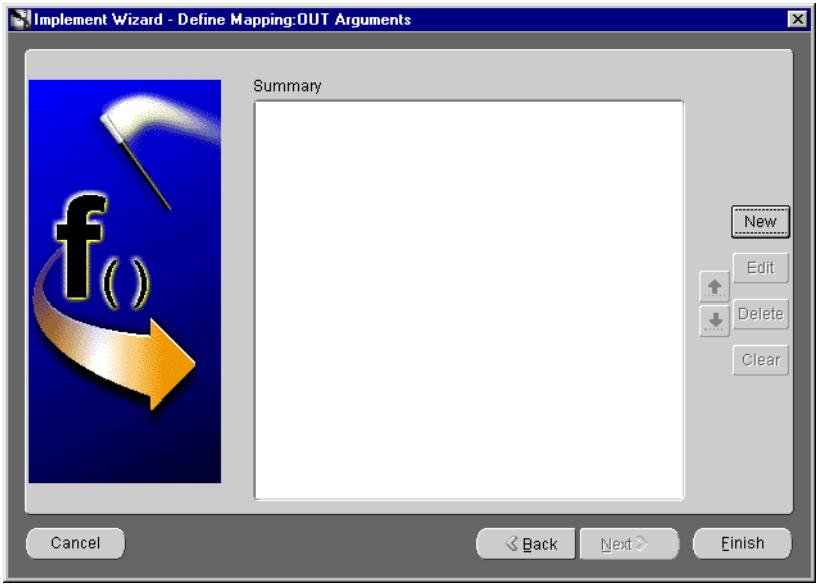


Figure 2–55 window 4 Implement Wizard - Define Mapping of Application View to Common View

- Click **New** to define mappings. See New Mapping on page 2-37 to map attributes.
- Click **Finish**.

The procedure implement message is created.

Exporting Stored Procedures

iStudio generates stored-procedure stubs to enable an application to interface with the Applications InterConnect run-time easily. These stubs are exported to a file using the export functionality.

To export stored-procedures, select **File** in the main menu, then click **Export**. The Export Application dialog is displayed: Figure 2–56

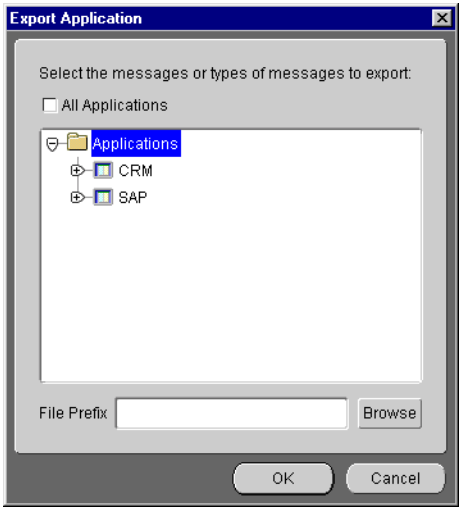


Figure 2–56 Export Application dialog

Filtering Messages

In the Export Application dialog, you can choose which messages to export stored procedures. Messages may be filtered in any of the following ways:

- to export all messages select **Applications**.
- to export all messages of a certain type (i.e. Published Events) for all applications, check the **All Applications** check box. Then select one or more types of messages to export.
- to export all messages for a specific application, select the **Application Name**.
- To export all messages of a certain type (i.e. Published Events) for a specific application, select that **type** under the **application name**.
- To export specific messages, select the messages by **name**. To select more than one message or class of messages **^click** the application.
- **File Name**—choose the name of the file to contain the exported stored procedures. The name you give generates multiple files. Click **Browse** to view the directory path.
- Click **OK** to export the stored-procedure, or **Cancel** to exit without storing.

The stored-procedure is now exported.

Editing Message Capability Matrix

For an application, the Message Capability Matrix displays queue information only for all receive messages i.e. subscribe, request receive part of implemented procedures, and reply receive part of invoked procedures. Based on this, the Runtime will automatically determine queues for the corresponding send messages i.e. publish, send request part of invoked procedures, and send reply part of implemented procedures.

To edit the message capability matrix for an application, ^**right-click** the **Message Capability Matrix** node under the **Routing** node of the application in the **Deploy** tree, then click **Edit**. The Edit Message Capability Matrix dialog is displayed: (Figure 2–57)

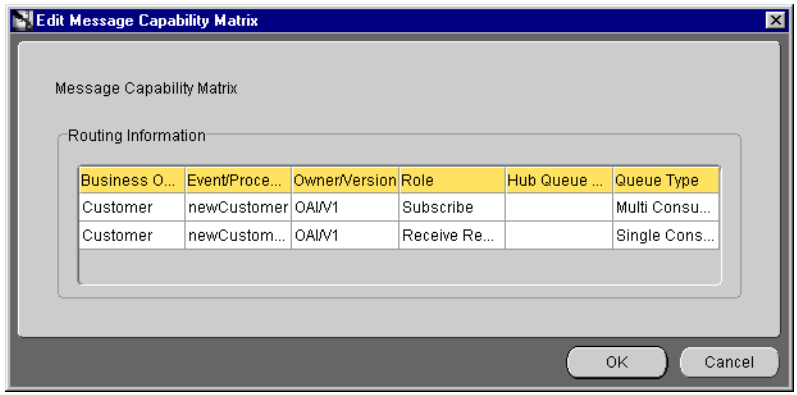


Figure 2–57 Edit Message Capability Matrix dialog

- Enter the queue names in the **Hub Queues** column.
- Click **OK** to save Message Capability Matrix and exit, or **Cancel** to exit without saving.

Creating a Partition

To create a partition, ^**right-click** the **Message Capability Matrix** node under the **Routing** node of the application in the **Deploy** tree, then click **Create Partition**. The Create Partition dialog is displayed: (Figure 2–58)

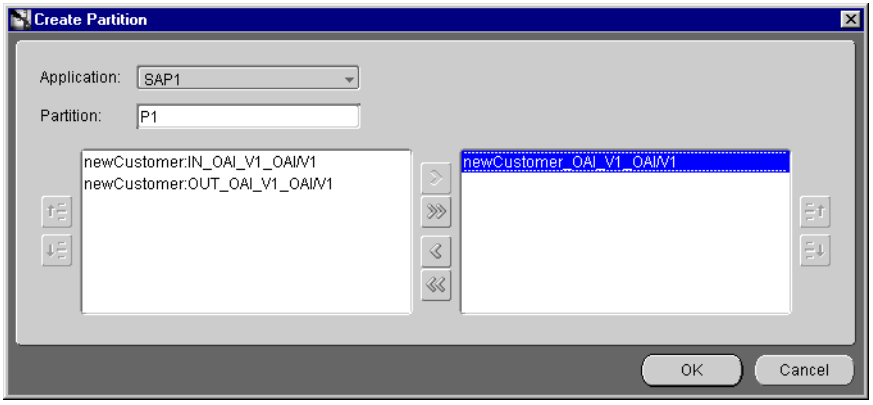


Figure 2–58 Create Partition dialog

- Enter a partition name in the text field.
- Assign messages to the partition by moving them from the left list to the right list.
- Click **OK** to save Partition and exit, or **Cancel** to exit without saving.

Editing Application Queues

For an application that processes XML messages, the Application Queues displays spoke application queue information.

To edit the application queues for an application, **^right-click** the **Application Queues** node under the **Routing** node of the application in the **Deploy** tree, then click **Edit**. The Edit Application Queues dialog is displayed: (Figure 2–59)

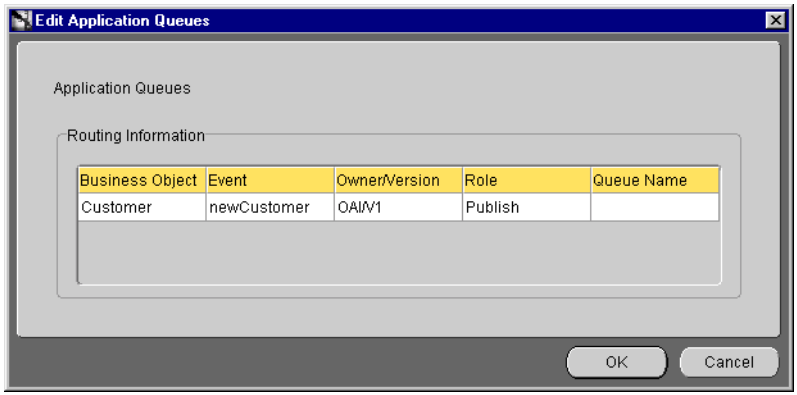


Figure 2–59 Edit Application Queues dialog

- Enter the queue names in the **Queues** column.
- Click **OK** to save Application Queues and exit, or **Cancel** to exit without saving.

3

Runtime Concepts and Components

This chapter describes the runtime concepts, components and processes of Applications InterConnect.

This chapter contains the following sections:

Introduction to the Runtime Component

The runtime component consists of an event-based distributed messaging system. An event is any action that triggers a message. The messaging system can be distributed with different components of the system communicating over a WAN or across the internet.

Features

The Applications InterConnect runtime features are categorized as follows:

- Integration architecture
- Message delivery
- Messaging paradigms
- Persistence
- Content-based routing
- Load balancing through message partitions

- Logging and tracing
- Fault tolerance

Integration Architecture

The runtime enables inter-application communication through hub and spoke integration. This methodology keeps the applications decoupled from each other by integrating them to a central hub only. The applications are at the spokes of this arrangement and are unaware of the applications they are integrating with. To them, the target is the hub. Since each application integrates with the hub, translation of data between the application and hub (in either direction) is sufficient to integrate two or more applications.

Refer to Chapter 2, "Design Time Concepts and iStudio", on page 2-1 for a graphical representation of the hub and spoke architecture.

Message Delivery

Guaranteed delivery

All messages when handed over to runtime, to be sent to another application have guaranteed delivery.

Exactly once delivery

The messages are neither lost nor duplicated. The destination application will receive each sent message exactly once.

In order delivery

The messages are delivered in the exact same order as they were sent.

Messaging Paradigms

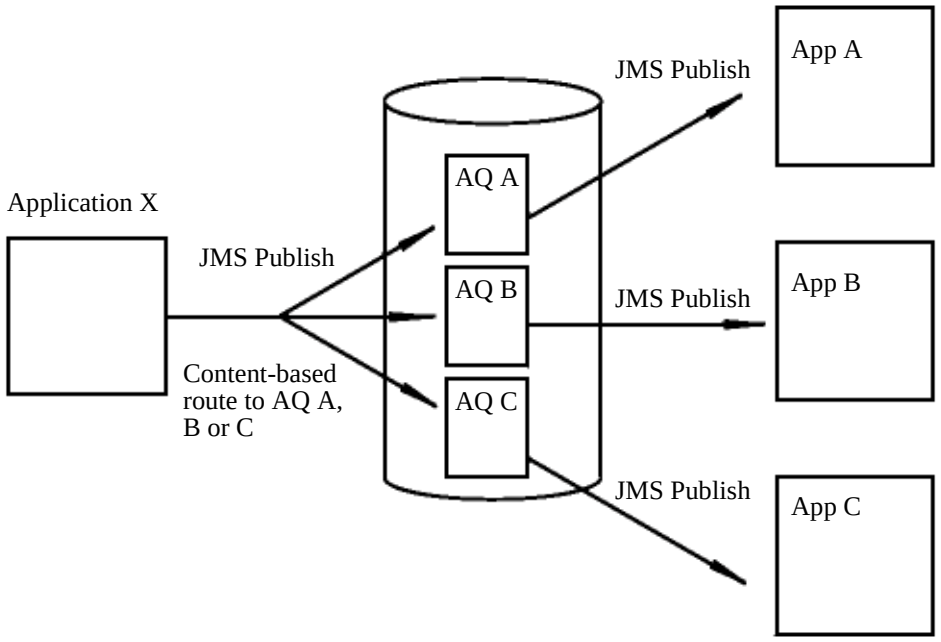
Refer to Chapter 2, "Design Time Concepts and iStudio", on page 2-1 for an explanation of the messaging paradigms supported in Applications InterConnect.

Persistence

Messages remain in the runtime system until they are delivered. The message is deleted when each application that is scheduled to receive a specific message has done so. For auditing purposes, you can configure the system to retain all messages even after they have been delivered successfully to each application.

Content-based Routing

Content-based routing increases message delivery efficiency by routing each message directly to the intended application by examining specific elements in the data object. The diagram below illustrates content-based routing in action.



The database in the middle is the one that resides in the hub under the Oracle Message Broker (OMB). OMB has been purposely omitted from the diagram to better illustrate the functionality. Also omitted are adapters (agents and bridges) that are attached to each application.

Messages can be routed to a specific application based on specific content values contained in the message. For example, an electronic funds transaction settlement application is designed to transmit bank transactions with a specific bank code to identify the destination bank system. When the EFT application publishes each message at runtime, the Oracle Application InterConnect runtime component determines the BankCode value based on objects stored in the repository, and routes the message to the appropriate recipient system.

To implement content-based routing in this scenario, the condition set captured in iStudio may be coded in this fashion:

```
if (BankCode EQUAL A) then route message to AQ A
if (BankCode EQUAL B) then route message to AQ B
if (BankCode EQUAL C) then route message to AQ C
```

iStudio allows you to specify SQL-like routing conditions based on the message content from the publishing application. This information is stored as metadata in the repository. At runtime, the publishing agent (connected to EFT application in the example above), utilizes this information to route the message to the specific recipient application.

Default Routing Support

On a per application basis, for every published message, you can specify a hub queue in which the message should be stored. Conversely, for each subscribing application, you can specify a hub queue from which the message should be retrieved. This pairing of publish and subscribe queues constitutes the Message Capability Matrix for each application. Using this matrix, the integrator can determine which queues need to be created in the hub. This matrix is stored in the repository as metadata and is used by the agents (see runtime section) to route messages to the appropriate queue on behalf of publishing applications, and to listen for messages on the appropriate queues on behalf of subscribing applications.

Load Balancing Through Message Partitions

At runtime, for performance reasons, you may need more than one adapter attached to a specific application. For example, Application A publishes three different kinds of events—EventA, EventB, and EventC. Three potential scenarios should be examined to determine whether (and how) one or more adapters should be attached to the application to meet performance objectives:

Scenario 1 The order in which the messages are sent by Application A must be adhered to strictly for the life of all messages. For example, if Application A publishes messages in a specific order, they must be received by the subscribing applications in the exact same order (even if they correspond to different event types). In this case, you cannot add more than one adapter to Application A for load balancing.

Scenario 2 The order in which messages are sent by Application A must be adhered to but not across different event types. For example, Application A publishes the following messages in order: M1_EventA, M2_EventB, M3_EventA. M1 and M3 must be ordered with respect to each other because they correspond to the same event. However, M2 has no ordering restrictions with respect to M1 and M3.

In addition, EventA messages are transformation/size/computation heavy and EventB and EventC messages are very light. In this case, you can create message partitions from the Message Capability Matrix. Partition1 can process EventA messages, and Partition2 can process EventB and EventC messages. When you install the adapters, you specify not only the application it is attach to but also the partition it uses. These message partitions can be used to effectively load balance integrated applications.

Scenario 3 There is no message order dependency, even within the same event type. Since there are no ordering restrictions, two approaches for load balancing can be employed:

A. No message partitions are created. One or more adapters are added utilizing the entire Message Capability Matrix. This means that at runtime any one of the adapters would be available to receive any message, though only one of them would actually receive the message.

B. Message Partitions can be created based on projections of the number of messages for a particular event type. For example, if there will be three times as many EventA messages than EventB or EventC messages, you could create two partitions—one for handling EventA messages, and the other for handling the other two event types.

Logging and Tracing

The runtime provides different levels of tracing to capture all information needed for troubleshooting or monitoring. The trace information is logged to local trace files that can be read using the Runtime Management Console. For more information, refer to Chapter 4, "Runtime Management Console".

Fault Tolerance

If at any time one or more of the runtime components or applications fail, none of the messages will be lost.

Load Balancing

Adapters (see components) offer multi-threaded support for load balancing. There can be multiple adapter instantiations attached to each application.

Components

There are three major components in the runtime system:

* Adapter (composed of an Agent and a Bridge)
* Oracle Message Broker

- Repository

Adapters

An adapter is the Applications InterConnect component that sits at the spoke with the application to make it InterConnect enabled. Internally, the adapter is written as two components for improved reuse of existing interfaces. These components are:

Bridge. This is the application specific piece of the adapter. The bridge communicates with the specific application interface to transfer data between the application and Applications InterConnect. For messages outbound from an application, the bridge is responsible for converting the data from the application's native system format to the agent's internal format (and conforming to the application view of data defined in iStudio). It then passes it on to the agent (described below) for further processing. For inbound messages, the bridge receives the message from the agent in the agent's internal format (and conforming to the application view of data defined in iStudio). It then converts the message back to the application's native format and pushes the data contained therein into the application. Each communication protocol requires a unique bridge.

Two applications using the same protocol may use the same bridge. code, though at runtime two separate processes are created. The bridge is also called the technology/protocol adapter.

Agent. The agent is a generic engine that carries out instructions for transformations and routing captured in repository metadata (populated by iStudio). The agent does not know how to talk to a particular application. For messages outbound from the application, the agent receives the message in it's internal format from the bridge. This internal format conforms to the application view of data (see iStudio description). The agent then queries the repository for metadata to transform this message to the common view and pushes the message to OMB.

For inbound messages, the agent receives a message from OMB that conforms to the common view defined in iStudio. The agent queries the repository for metadata and transforms the message from the common view to the application view. The message is then pushed to the bridge. The agent is also know as the integration adapter.

The following adapters are available with OAI 4.0:

- Database Table Adapter
- AQ/XML Adapter
- CRM 11i Adapter
- SAP Adapter

- HTTP Adapter

Oracle Message Broker

Oracle Message Broker (OMB) is the message store and forward component of the runtime. Adapters send messages to OMB which stores them in an Oracle 8i database using Advanced Queuing (AQ). OMB then delivers the messages to adapters who have subscribed to them. Messages are deleted from the persistent store after each recipients who expects the messages have received them. OMB conforms to the Java Messaging System (JMS) specification for a messaging server. JMS communication between each adapter and the broker is built on top of CORBA.

Repository

The Repository communicates with adapters at runtime using CORBA to provide translation information for messages. This translation information is called metadata. The Repository is populated with metadata during the design phase using iStudio. Metadata customizes a generic adapter to tend to a specific application's integration reqirements.

Example

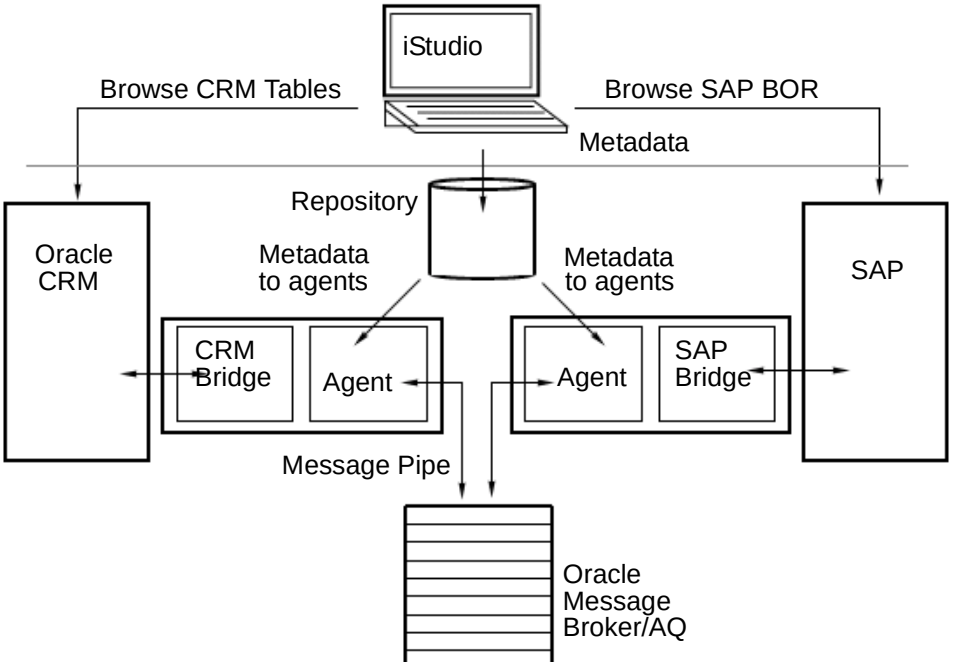


Figure 3–1 CRM Application and SAP communication - A graphical interpretation

The CRM application and SAP communication interpretation from the above diagram is as follows:

1. The CRM Adapter at startup, queries the Repository for message translation information. It is aware that it is plugged into an Oracle CRM Application, so it queries the Repository for all Oracle CRM Application-related message translation information. The metadata containing this information is cached in the CRM Adapter.
2. The SAP Adapter at startup, expresses an interest in all (or some) messages that the CRM application is publishing. It also caches all SAP-related metadata from the Repository. Note that the Repository was populated with the metadata at design time using iStudio.

3. An event occurs in the Oracle CRM application.

   Note: It is the application's responsibility to **capture the event** and the data associated with it internally and expose it to the OAI adapter.

4. As a result of the event, the application transfers all event-related information to the bridge through bridge APIs called by the application infrastructure.

   Note: This bridge was custom-developed as an extension to the agent to tailor it to support an Oracle CRM application.

5. The bridge creates a message with the event information using published agent APIs. It then transfers the message to the agent also using published agent APIs.
6. On receiving the message in question, the CRM agent looks up the metadata information in the cache and performs the necessary translations. These translations are from the Oracle CRM Application's view to the hub or common view as described above in the hub and spoke architecture section.
7. The common view message is now shipped to the broker which stores it in an Oracle 8i database.
8. The SAP Adapter (using JMS APIs) receives this message into the adapter space.
9. On receiving the message, the SAP agent reviews its cached metadata information and translates this message from the hub view to the SAP application view.
10. It then transfers the message to the SAP bridge.

    Note: This bridge was custom developed as an extension to the agent to tailor it to handle an SAP application.

11. The bridge calls into SAP to deliver the message contents to the application infrastructure.

    Note: It is the application's responsibility to **resolve the event** internally and apply the data and the logic to its internals.

4

Management Infrastructure

This chapter describes how to manage OAI components using the Oracle Enterprise Manager Console.

This chapter contains the following sections:

Introduction to OAI Management

After installing the OAI Management Infrastructure, you can use the Oracle Enterprise Manager to administer the Applications InterConnect runtime components, the Repository and the Adapters.

Using the console, the administrator can start, stop, monitor and troubleshoot the Applications InterConnect runtime.

The Oracle Applications InterConnect runtime includes two OAI components—the Adapters and Repository. The console locates each instance of Oracle Applications InterConnect components in the system using a directory service. The console allows you to determine what components are started and stopped and to start/stop them. The administrator can troubleshoot the system in case of component failures or malfunction using the console.

Console Features

The console's navigator tree has two main components:

- Adapters
- Repositories

Adapters

Under Adapters you will see a list of the Adapters that you have running. Right click on the Adapter that you are interested in and select:

- Get State

  Get State to determine the Adapter's state. Red indicates that the Adapter is suspended or stopped. Green indicates that the Adapter is running. Yellow indicates that the status has not been determined.

- Stop

  Stop disables the Adapter from sending and receiving messages.

- Start

  Start enables the Adapter to send and receive messages.

- Suspend

  Suspend disables the Adapter from processing the messages being sent from the application and received from OMB.

- Resume

  Resume enables the Adapter to resume processing of messages that have been sent from the application and received from OMB.

- Shutdown

  Shutdown shuts down the Adapter process and removes the Adapter from the list of Adapters.

- Errors

  Errors retrieves all errors logged by this Adapter.

Repositories

Under Repositories you will see a list of the Repositories that you have running. Right click on the Repository that you are interested in and select:

- Stop

  Stop disables the Repository so that all connections will be refused.

- Start

  Start enables the Repository so that it can accept all connections.

- Shutdown

Shutdown shuts down the Repository process and removes the Repository from the list of Repositories.

Glossary

Advanced Business Application Programming (ABAP)

Advanced Business Application Programming (ABAP) is a programming language developed by SAP for application development purposes.

Business Application Programming Interface (BAPI)

Standardized programming interface that enables external applications to access the business processes and data of the R/3 System. Business Application Programming Interfaces (BAPIs) are defined in the Business Object Repository (BOR) as methods applied to SAP business objects, in order to perform specific business tasks.

Business Object

Represents a central business object in the real world. R/3 business objects describe complete business processes. This type of encapsulation reduces complexity because the inner structure of the business object remains concealed. By invoking methods known as BAPIs (Business APIs), external applications can access and manipulate the business objects via the Internet, DCOM or CORBA.

BAPI Browser

BAPI-oriented view of the Business Object Repository (BOR). The BAPI Browser displays all the business objects, for which BAPIs have been implemented.

IDoc

A specific instance of an IDoc type.

IDoc type

The IDoc type indicates the SAP format that is to be used to transfer the data for a business transaction. An IDoc is a real business process in the form of an IDoc type. An IDoc type is described using the following components:

•a control record is the format of the control record which is identical for all IDoc types.

•one or more data records is a data record consists of a fixed administration part and a data part (segment). The number and format of the segments can be different for each IDoc type.

•status records is the status record describing the processing stages which an IDoc can pass through, and have an identical format for each IDoc type.

Example: Purchase order no.4711 was sent to a vendor as IDoc no.0815. IDoc no.0815 is formatted in IDoc type ORDERS01 and has the status records "created" and "sent".

IDoc interface

Definition of IDoc types and data interchange methods (port definition) between SAP systems and partner systems. Partner systems can be:

- EDI subsystem
- other R/3 System
- other R/2 System
- third-party software

Index

I

L

M

O

P

R

S

T

V

W