IBM WebSphere Business Integration Express for
Item Synchronization

Business Object Development Guide

IBM WebSphere Business Integration Express for
Item Synchronization

Business Object Development Guide

Note!
Before using this information and the product it supports, read the information in "Notices" on page 83.

10October2003

This edition of this document applies to IBM WebSphere Business Integration Express for Item Synchronization, version 4.3, and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email doc-comments@us.ibm.com. We look forward to hearing from you.

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

Contents

About this document

The IBM(R) WebSphere(R) Business Integration Express for Item Synchronization product includes InterChange Server Express, the associated Toolset Express product, the Item Synchronization collaboration, and a set of software integration adapters. Together they provide business process integration and connectivity among leading e-business technologies and enterprise applications as well as integration with the UCCnet GLOBALregistry.

This document describes how to use Business Object Designer Express to create business object definitions, both manually and using an Object Discovery Agent (ODA). Object Discovery Agents are designed to "discover" business object requirements specific to a data source and to generate definitions from those requirements. Business Object Designer Express provides a graphical user interface (GUI) to the available Object Discovery Agents, and helps manage the discovery and definition generation processes.

Audience

This document is for IBM customers, consultants, or resellers who create or modify business objects. Before you start, you should be familiar with all the concepts explained in the *User Guide for WebSphere Business Integration Express for Item Synchronization*.

Related documents

The complete set of documentation describes the features and components common to all installations of IBM WebSphere Business Integration Express for Item Synchronization, and includes reference material on specific components.

This document contains many references to the *User Guide for WebSphere Business Integration Express for Item Synchronization*. If you choose to print this document, you may want to print the User Guide as well.

You can download, install, and view the documentation at the following site:

http://www.ibm.com/websphere/wbiitemsync/express/infocenter

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic, *`italic`*	Indicates a variable name or a cross-reference.
blue outline	A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.

`[ ]`	In a syntax line, square brackets surround an optional parameter.
`...`	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
`< >`	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><connector_name>tmp.log`.
`\`	In this document, backslashes (\) are used as the convention for directory paths. All IBM product pathnames are relative to the directory where the IBM product is installed on your system.
ProductDir	Represents the directory where the product is installed.
%text%	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable.

New in this release

Business Object Designer Express is the first release as part of the IBM WebSphere Business Integration Express for Item Synchronization release.

Chapter 1. Introduction to business objects

A business integration system uses business objects to carry data and processing instructions between an integration broker and connectors or an access client. Business objects represent a request from an integration broker, an event in an application or Web server, or a call from an external site. This manual presents information on developing and designing business objects. The main topics in this chapter are:

The material presented here assumes that you understand the basic concepts described in the *User Guide for WebSphere Business Integration Express for Item Synchronization*.

Business objects in the WebSphere business integration system

The WebSphere business integration system consists of the following components:

- A set of WebSphere Business Integration Adapters

  A WebSphere Business Integration Adapter, called simply an *adapter*, is a set of software modules that communicate with an integration broker and with applications or technologies to perform tasks such as executing application logic and exchanging data.
- An integration broker

  The task of an *integration broker* is to integrate data among heterogeneous applications. WebSphere Business Integration Express for Item Synchronization uses InterChange Server Express.

In the WebSphere business integration system, information sent or received between components is packaged in the form of a *business object*, as follows:

- For data that is transferred between a connector and an integration broker, you design *application-specific business objects* that model the appropriate application entities.
- For data that is processed within the business logic of an InterChange Server Express collaboration object, you design *generic business objects* that contain a superset of information for the application entities that need to communicate. Maps are used to transform data between generic business objects and application-specific business objects so that connectors can communicate with their applications using application-specific entities, while collaboration objects can apply business logic in an application-independent way.

Both application-specific business objects and generic objects are modeled during design-time as *business object definitions*, which are stored in the business integration system. At runtime, data is populated in a *business object instance* (often just called a "business object") that is based on the proper definition, and is moved through the business integration system as dictated by its routing and business logic rules.

Business object definitions

A *business object definition* represents a template for data that can be treated as a collective unit. It contains a business object header that specifies the name and version of the business object definition. In addition, the business object definition contains the following information:

- "Business object attributes and attribute properties"
- "Business object verb" on page 5
- "Business object application-specific information" on page 5

Figure 1 shows the parts of a business object definition.

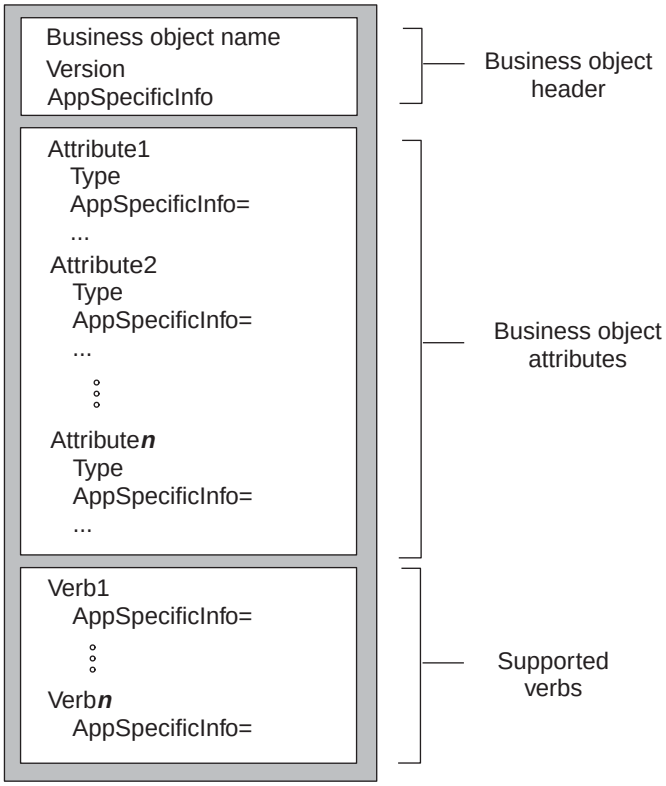


Figure 1. A business object definition

Business object attributes and attribute properties

A business object contains attributes, each *attribute* representing a single entity of data. In the business object definition, you define the name of each attribute as well as other *attribute properties*. The business object instance holds a value for each attribute (or indicates that the attribute does not have a value).

Business object definitions include various properties that apply to attributes. These properties provide the connector, data handler, and other components with information on the types, sizes, and default values of attributes. The attribute properties are discussed in the sections that follow.

Name property: Each business object attribute must have a unique name within the business object definition. The name should describe the data that the attribute

contains. The name can be up to 80 characters; it can contain alphanumeric characters and underscores but cannot contain spaces, punctuation, or special characters.

Notes:

1. When designing an application-specific business object, check its adapter User Guide or the *Data Handler Guide* for specific naming requirements and recommendations.
2. This attribute name must use *only* characters defined in the code set associated with the U.S. English locale (`en_US`).

Type property: The Type property defines the data type of the attribute:

- For a simple attribute, the supported types are `Boolean`, `Integer`, `Float`, `Double`, `String`, `Date`, and `LongText`.
- For a complex attribute, the type is a business object definition:
  - If the attribute represents a child business object, specify its type as the name of the child business object definition and specify the cardinality as `1` (single cardinality).
  - If the attribute represents an array of child business objects, specify the type as the name of the child business object definition and specify the cardinality as `n` (multiple cardinality).

Note: All attributes that represent child business objects also have a `ContainedObjectVersion` property (which specifies the version number of the child object's business object definition) and a `Relationship` property (which specifies the value `Containment`).

Cardinality property: Each simple attribute has single cardinality (cardinality `1`). Each complex attribute, which represents a child business object or array of child business objects has single cardinality or multiple cardinality (cardinality `n`), respectively. For more information on cardinality, see "Hierarchical business objects" on page 9.

Note: When specified for a required attribute, single cardinality indicates a child business object *must* exist, and multiple cardinality indicates zero to many instances of a child business object.

Key property: At least one attribute in each business object must be specified as the key. The key attribute contains a value that uniquely identifies the business object. To define an attribute as a key, set its Key property to `true`.

Note: A key value in a business object is often referred to as its *primary key*.

When you specify as key a complex attribute:

- If the attribute represents a child business object, the key is the concatenation of the keys in the child business object.
- If the attribute represents an array of child business objects, the key is the concatenation of the keys in the child business object at location 0 in the array.

Foreign key property: The Foreign Key property is typically used in application-specific business objects to specify that the value of an attribute holds the primary key of another business object, serving as a means of linking the two business objects. The attribute that holds the primary key of another business object is called a *foreign key*. Define the Foreign Key property as `true` for each attribute that represents a foreign key.

You can also use the Foreign Key property for other processing instructions. For example, this property can be used to specify what kind of foreign key lookup the connector performs. In this case, you might set Foreign Key to `true` to indicate that the connector checks for the existence of the entity in the database and creates the relationship only if the record for the entity exists.

Required property: The Required property specifies whether an attribute must contain a value. If a particular attribute in the business object must contain a value to be able to process the business object data, set the Required property for the attribute to `true`.

AppSpecificInfo: The AppSpecificInfo property can contain a `String` of up to 1000 characters that is specified primarily for an application-specific business object. For information on this property, see "Business object application-specific information" on page 5.

Note: Application-specific information is *not* available in the mapping process.

Max Length property: The Max Length property is set to the number of bytes that a `String`-type attribute can contain. Although this value is not enforced by the WebSphere business integration system, specific connectors or data handlers might use this value. Check the guide for the specific adapter or the guide for the data handler that processes the business object to determine minimum and maximum allowed lengths.

Important: The Max Length property is very important when you use a fixed-width data handler.

Note: Attribute length is *not* available in the mapping process.

Default value property: The Default Value property can specify a default value for an attribute.

If this property is specified for an application-specific business object, and the `UseDefaults` connector configuration property is set to `true`, the connector can use the default values specified in the business object definition to provide values for attributes that have no values at runtime.

Notes:

1. The attribute's default value can use any characters defined in the code set associated with the current locale.
2. For an attribute whose type is String, you can specify a blank character as a default value.

Comments property: The Comments property allows you to specify a comment for an attribute. Unlike the AppSpecificInfo property, which is used to process a business object, the Comments property provides only documentation information, which may assist other developers in understanding your design decisions.

Note: The attribute's comments can use any characters defined in the code set associated with the current locale. However, the newline character is invalid.

ObjectEventId attribute: The `ObjectEventId` attribute is not only required, but it must be the last attribute in every business object. The WebSphere business integration system uses this attribute to identify and track an event flow in the system.

The ObjectEventId attribute stores a unique value that identifies each event in the WebSphere business integration system.The connector framework generates values for this attribute in the parent business object and in each child.

Important: Do *not* map the ObjectEventId attribute or have a connector or data handler populate it. The business integration system handles the value of this attribute.

Business object verb

The business object definition includes a list of the verbs that the business object can support. These verbs correspond to operations that are valid on the data within the business object. At runtime, a business object contains a single active verb, which describes the operation to perform on the data in that particular business object.

Table 1 lists the basic verbs that a business object definition can support.

Table 1. Basic verbs

Verb	Description
Create	Make a new entity in the application.
Retrieve	Using key values, return a complete business object.
Update	Change the value in one or more fields in the application entity.
Delete	Remove the entity from the application. This operation must be a true physical delete.

Note: Because the verb operations in Table 1 are the most common to perform, they are often collectively referred to by an acronym of their initial first letters: CRUD.

In addition to the basic verbs in Table 1, a business object definition might also need to support one or more of the following verbs:

- RetrieveByContent—Using non-key values, return a complete business object.
- Exist—Check for existence of a specified entity but do not retrieve it.
- Custom—Perform some application-specific operation.

Business object application-specific information

A business object definition can provide *application-specific information*, whose content provides *meta-data* to the component that processes the business object. A common use of application-specific information is to provide a connector or data handler with application-dependent instructions on how to process the business object. The application-specific information is a string that is entered during business object design and read at runtime by a connector or data handler.

Note: Connectors that are designed to use the application-specific information in definitions of their application-specific business objects are called *meta-data-driven connectors*. Because the processing information is configurable, rather than hard-coded, a meta-data-driven connector is much more flexible and easier to maintain than one that is not meta-data-driven.

Within a business object definition, you can provide application-specific information at one of three levels:

- The business object definition
- An attribute within the business object definition
- The business object verb

Application-specific information is stored in a field in the business object definition called the AppSpecificInfo property. The value of the AppSpecificInfo property is a text string that can include any information about the business object or application. Figure 2 illustrates the major elements of a business object definition and the application-specific property for each element.

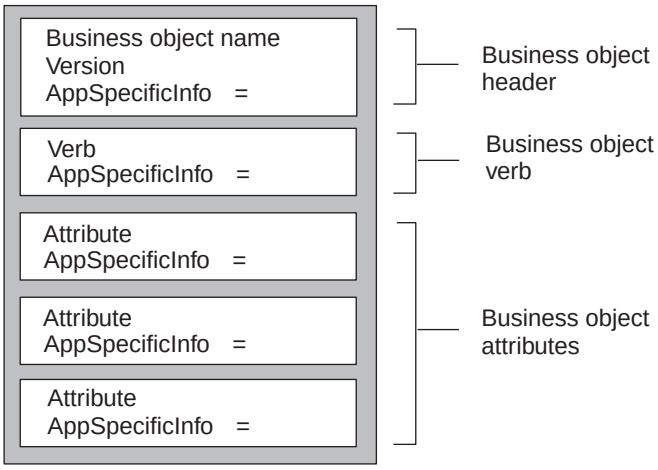


Figure 2. Business object definition showing the application-specific property for each element.

This section covers the following topics:

- Application-specific information for a business object
- Application-specific information for an attribute
- Application-specific information for a verb

Application-specific information for a business object: The application-specific information at the business object level provides information that the connector or data handler uses to process the data. Business object-level application-specific information is used whenever processing instructions are relevant for an entire business object hierarchy. For example, the object-level application-specific information might do one or more of the following:

- Define the scope of business object transaction processing
- For applications that require object processing in an application extension, contain the name of the function to call to handle the business object
- Specify the name of the table or form in which the record belongs
- Specify the name of an attribute within the business object that represents a logical or "soft" delete

Figure 3 illustrates application-specific information that identifies a form or table name in an application. The connector can get the table or form name from the AppSpecificInfo property and use it in an API call to retrieve data from the application.

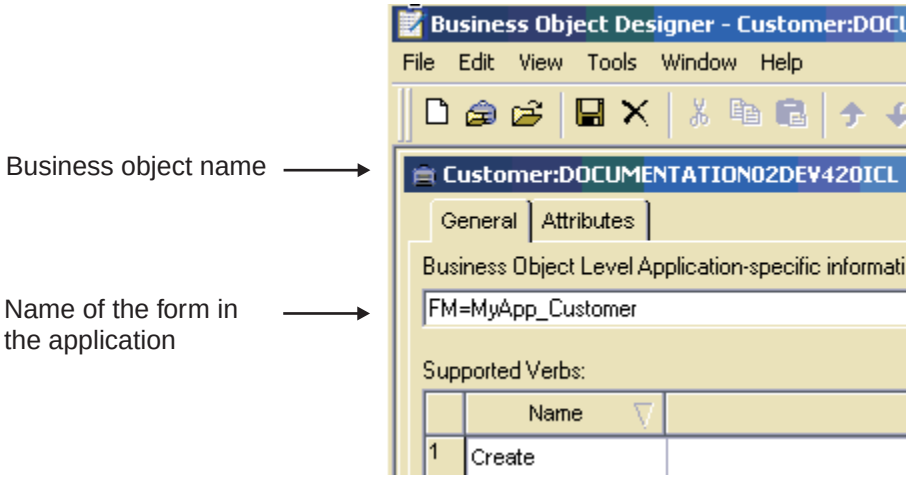


Figure 3. Application-specific information for a business object.

Application-specific information for an attribute: Each attribute of a business object definition can have application-specific information associated with it. Attribute-level application-specific information is used whenever processing instructions are relevant for the single attribute. For example, this information can specify a field on a form, a column in a table, or whatever the connector needs to locate or work with the attribute. If certain attributes of a business object are located on a particular subform in the application, the AppSpecificInfo property is a good candidate for a place to encode this information.

Figure 4 illustrates the AppSpecificInfo property for an attribute. In this example, the application-specific information specifies the name of a subform and field.

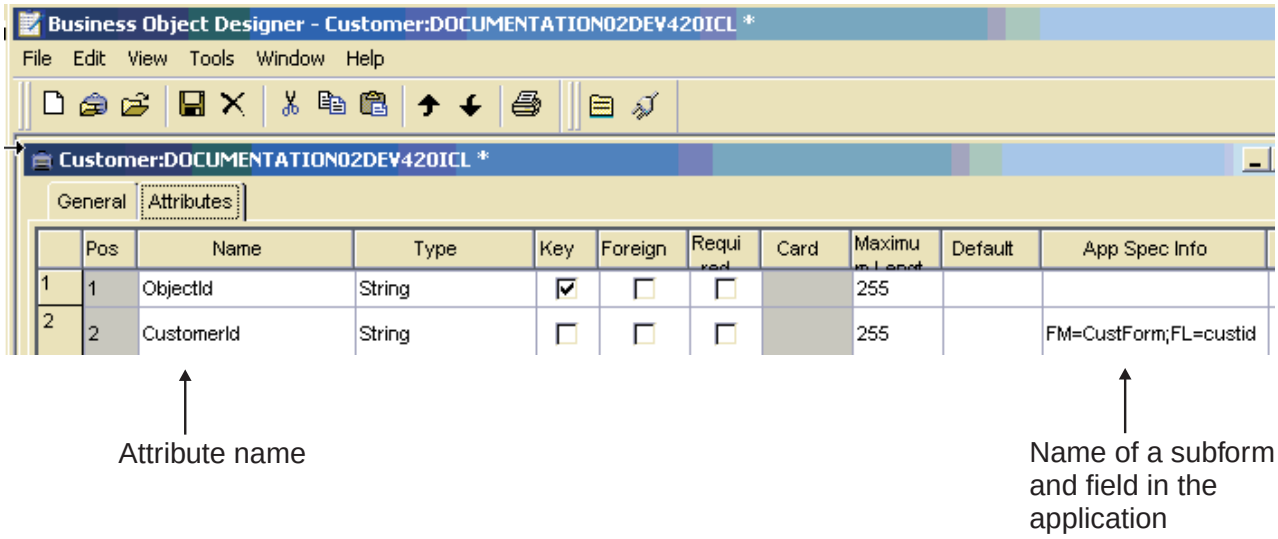


Figure 4. Application-specific information for an attribute.

Figure 5 illustrates the relationship of form, subform, and field name as provided in the object-level and attribute-level application-specific information. This example assumes that a billing application is based on forms, and that the way to interact with invoices in this application is through the Invoice form, which is a subform of

a main CustAccount form. On the Invoice subform, there are the following fields: CustName, CustAddr, InvNum, DollarAmount, and Terms.

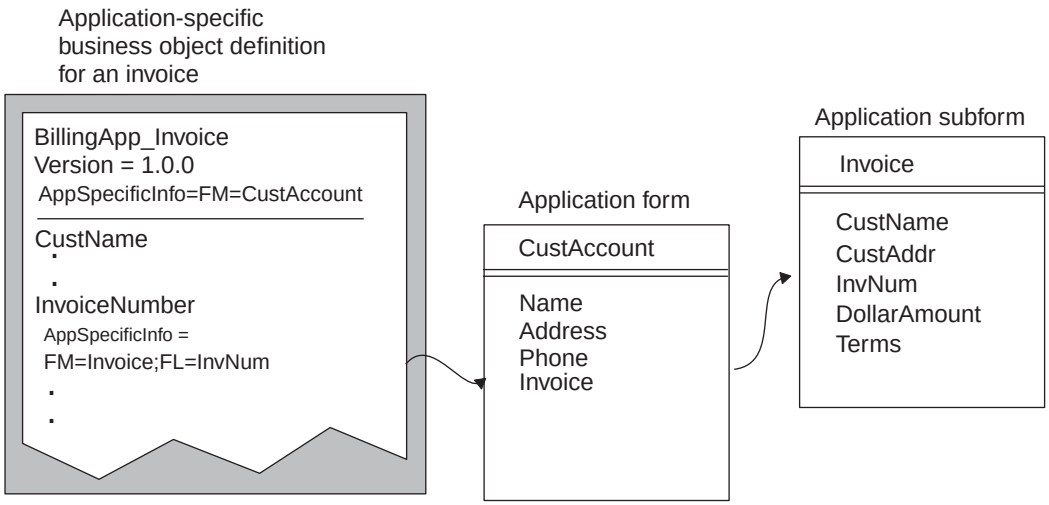


Figure 5. Using business object definition application-specific information.

Figure 5 uses the attribute-level AppSpecificInfo property to store the name of the Invoice subform and the attribute's corresponding field name. The example uses name-value pairs to specify the information.

Application-specific information for a verb: Each verb definition can include application-specific information that provides the connector or data handler with instructions on how to process the business object when that verb is active.

Note: The business object handler, which is the part of a connector that handles requests sent from the integration broker to the connector, can be designed to use the application-specific information in the verbs of their application-specific business object definitions. Such business object handlers are called *meta-data-driven business object handlers*. Because the processing information is configurable, rather than hard-coded, a meta-data-driven business object handler is much more flexible and easier to maintain than one that is not meta-data-driven.

For example, if the connector is using an API to handle updates to the application database, the application-specific information can provide the connector with information to execute an API.

The verb application-specific information can also specify the name of a function to call in the application to handle the processing of a business object.

Business object instances

While the business object definition represents the template for a collection of data, a business object instance (often just called a "*business object*") is the runtime entity that contains the actual data. The business object is what is passed between components of the business integration system.

The business object contains the following information:

- Attributes, each of which contains the data for the associated attribute. One of the attributes is usually a key attribute, which contains a value that uniquely identifies this business object among all business objects of the same definition.

- An active verb, which should be one of the supported verbs for the business object definition

Figure 6 shows the `Customer` business object definition and a corresponding business object instance for this definition.

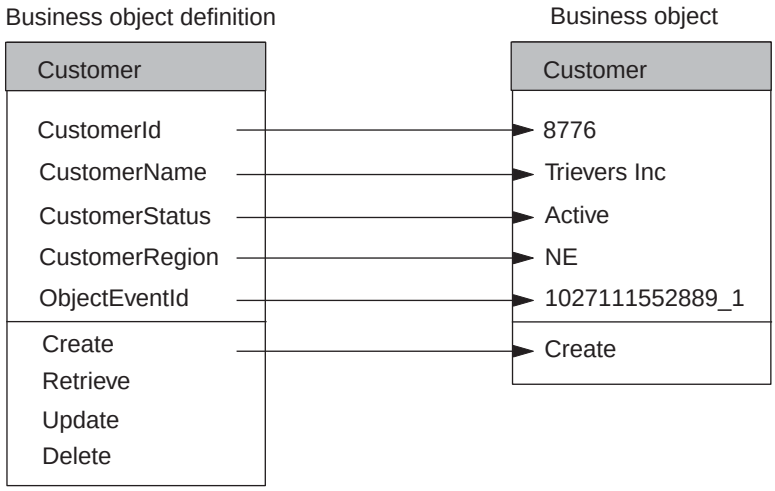


Figure 6. Business object definition and sample business object.

Business object structure

The structure of a business object can be either of the following:

- Flat business objects
- Hierarchical business objects

The following sections show examples of flat and hierarchical business object structures, and provide information on how the business object structure affects connector logic.

Flat business objects

A business object definition for a *flat business object* contains one or more simple attributes and a list of supported verbs. A *simple attribute* represents a single value, such as a `String` or `Integer` or `Date`. All simple attributes have single cardinality. The Customer business object in Figure 6 is an example of a flat business object. For more information, see "Business object attributes and attribute properties" on page 2.

Hierarchical business objects

Hierarchical business object definitions define the structure of multiple related entities, encapsulating not only each individual entity but also aspects of the relationship between entities. In addition to containing at least one simple attribute, a hierarchical business object has one or more attributes that are *complex*; that is, the attribute itself contains one or more business objects, called *child business objects*. The business object that contains the complex attribute is called the *parent business object*.

There are two types of relationships between parent and child business objects:

- *Single cardinality*—When an attribute in a parent business object represents a *single* child business object. The type of the attribute is set to the name of the child business object, and the cardinality is set to 1.
- *Multiple cardinality*—When an attribute in the parent business object represents an *array* of child business objects. The type of the attribute is set to the name of the child business object, and the cardinality is set to `n`.

In turn, each child business object can contain attributes that contain a child business object or an array of business objects, and so on. The business object at the top of the hierarchy, which itself does not have a parent, is called the *top-level business object*. Any single business object, independent of its child business objects it might contain (or that might contain it) is called an *individual business object*.

In a typical business object hierarchy, a top-level business object definition contains one or more simple attributes, one or more attributes that represent a child or array of child business objects, and a list of supported verbs. Figure 7 shows a typical hierarchical business object. The top-level business object, Customer, has both single-cardinality attributes and multiple-cardinality attributes with child business objects:

- Its Address attribute is a complex attribute with multiple cardinality. Customer is the parent business object for each of the Address child business objects.
- Its CustProfile attribute is a complex attribute with single cardinality. Customer is the parent business object for the single CustProfile child business object.

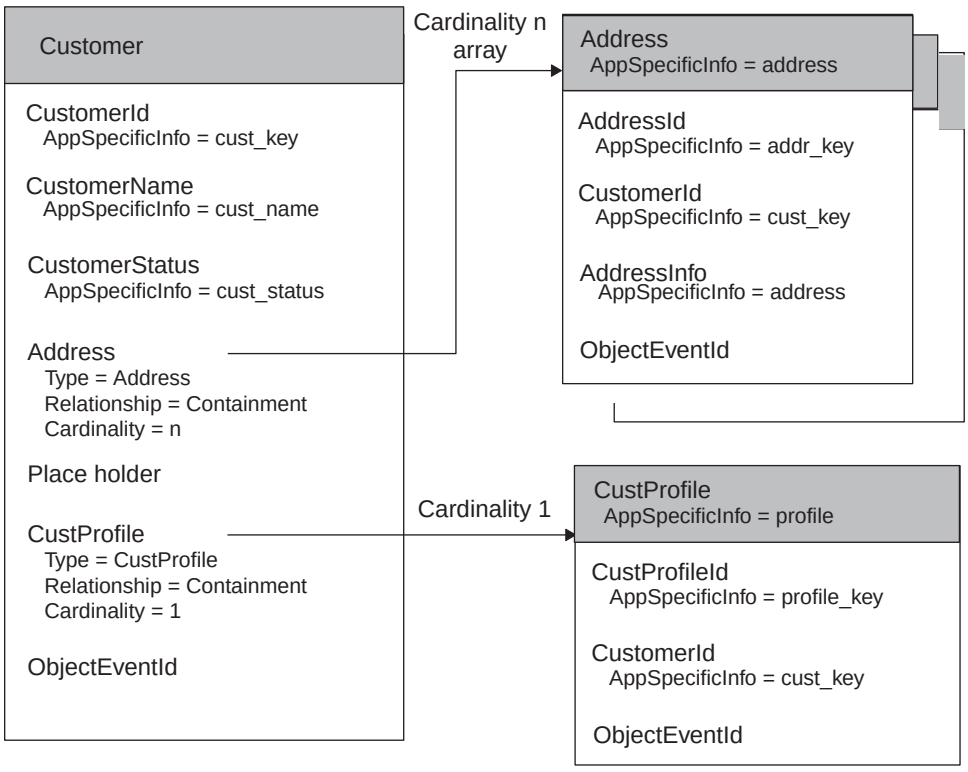


Figure 7. Example of a hierarchical business object definition

In Figure 7, the Customer and CustProfile business objects, as well as each of the Address business objects is an individual business object.

Note: When a top-level business object contains information used to process its child business objects, it is called a wrapper business object. For example, the XML connector requires a wrapper business object to contain information that determines the format of its child data business objects and routes the children.

When designing the structure of a hierarchical application-specific business object, you need to determine:

- How entity data is represented in the business object
- How the primary application entity relates to child entities
- If an application entity includes data from different entities, you must decide:
  - Whether the application-specific business object needs to include related data
  - How to define the relationship between the related data

For more information, see "Design considerations for multiple entities" on page 21.

Overview of the development process

This section provides an overview of the business object development process.

Setting up the development environment

Before you start the development process, the following must be true:

- The WebSphere business integration system is installed on a computer that you can access.

  For information on how to install and start up the WebSphere business integration system, refer to *Installing WebSphere Business Integration Express for Item Synchronization*.
- InterChange Server Express and its repository's database server are running. This step is required only when you are ready to save the definition to the repository or to delete a definition from the repository. For development only, you can run Business Object Designer Express locally, without connecting to InterChange Server Express.

Stages of business object development

The stages of business object development are as follows:

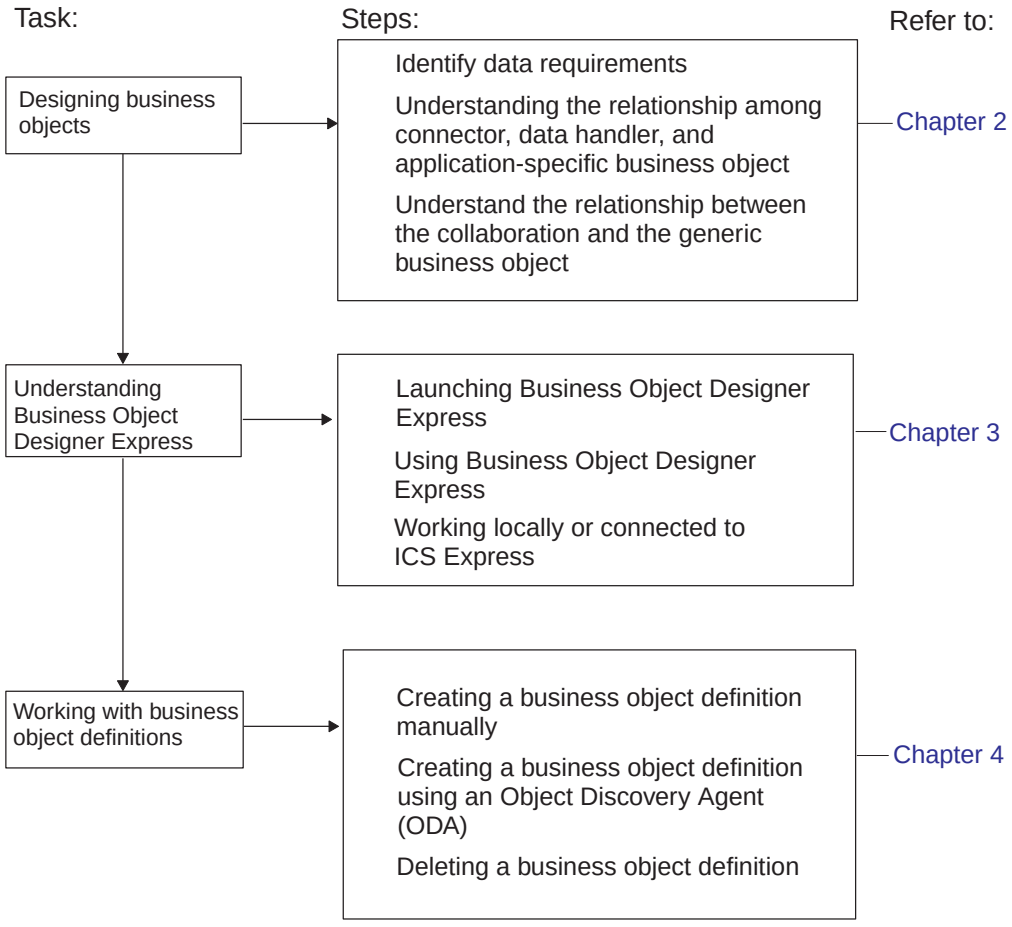
1. Understand the data requirements that are critical to the business process integration.
   - If creating an application-specific business object, understand the relationship among the connector, the data handler, and the supported application-specific business objects.
   - If creating a generic business object, understand the relationship between the collaboration object and the business object.
2. Develop the business object definitions in one of two ways:
   a. Generation from a data source—the WebSphere business integration system provides tools that facilitate generation of a business object definition for some connectors. Such tools are Object Discovery Agents or command line tools that are designed to connect to an application and "discover" business object requirements specific to a business entity and to generate definitions from those requirements. Business Object Designer Express presents a graphical user interface to Object Discovery Agents, and helps manage the discovery and definition generation processes. Check the guides for the adapter and data handler you will be using to determine if they have an

available tool or utility. You can also check the Connector Feature Checklist, which is available on the main documentation page under the Connectors category.

b. Manual—Business Object Designer Express is a graphical user interface that facilitates the manual creation of business object definitions. This interface is most useful for developing generic business objects to use with InterChange Server Express, as there is no application in which object discovery might be performed.

3. If you used a tool to automatically generate the business object definition from a data source, verify that the generated structure and application-specific information conforms to requirements. Reference the adapter user guide for the connector that uses the business object definition determine any special configuring that you must do manually.
4. Test and debug the business object by running it through the system; edit it as necessary.

The following illustration shows a visual overview of the business object development process and provides a quick reference to chapters where you can find information on specific topics.



Chapter 2. Designing business objects

The key to the design of business objects is to develop a business object definition that models as closely (and efficiently) as possible the data that needs to be transmitted between components of the business integration system:

- For data that is transferred between a connector and an integration broker, you design *application-specific business objects* that model the appropriate application entities. These entities might correspond to data structures or technology standards used by a particular application, or to specific technology standards used by a web server.
- For data that is processed within the business logic of an InterChange Server Express collaboration object, you design *generic business objects* that contain a superset of information for the application entities that need to communicate. When the collaboration object exchanges information with an application, maps convert the data between the generic business object and application-specific business object structures.

This chapter presents information on the structure of business objects for the WebSphere business integration system, and makes recommendations for designing both application-specific and generic business objects. The material presented here assumes that you understand the basic object concepts described in the *User Guide for WebSphere Business Integration Express for Item Synchronization*.

The main topics of this chapter are:

Determining business object structure

The purpose of a business object is to transport data between components of a business integration system and the applications that it integrates. Therefore, the business object should model the data that needs to be transported. This data is usually associated with some entity in an application or technology that the business integration system integrates. The structure of a business object can be either of the following:

In addition, this section provides "Design considerations for multiple entities" on page 21.

Representing a single entity

The simplest business object design is a flat business object that represents a single entity. All the attributes of a flat business object are simple (that is, each attribute represents a single value, such as a `String` or `Integer` or `Date`). For more information, see "Flat business objects" on page 9.

In the case of an application-specific business object, a flat business object can represent a single entity in an application or in a technology standard. For example, assume an application has a database table that describes a record. Assume further that this table has five columns named ObjectID, UserName, TimeStamp, Detail, and Status (see Figure 8). The ObjectID is the primary key for each row, and its value is generated by the application. This table has no relationships to other tables.

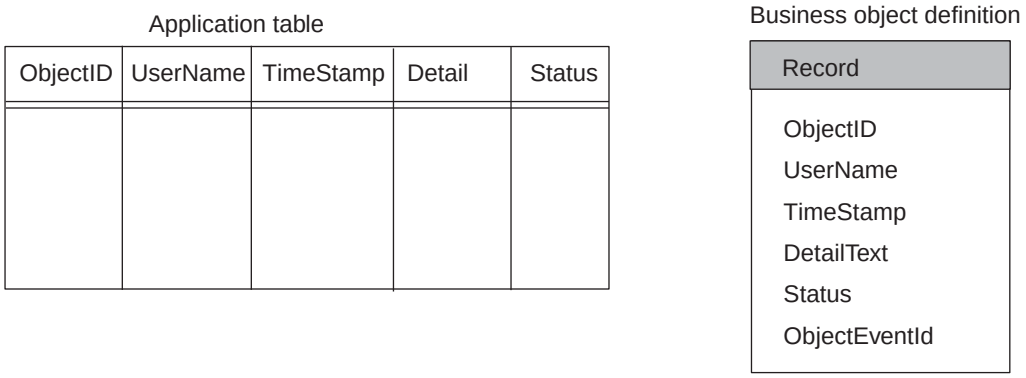


Figure 8. Flat business object representing a single entity

As Figure 8 shows, the Record business object you design to represent the table might have five attributes, one for each column, with the key attribute corresponding to the ObjectID column.

Use of flat business objects can simplify corresponding connector design in the following ways:

- On a Create operation, the connector might cycle through the attributes, extracting the non-key attribute values from the business object instance and extracting processing instructions from the business object definition. When it has assembled the information it needs to process the business object, the connector might execute an application function call or SQL statement to create a new row for the record in the table. The connector then returns a value for the key to the business integration system.
- On a Retrieve operation, the connector might extract the primary key from the business object request, use the key value to retrieve the current set of data for the row, and return a business object with the complete set of values.

This type of business object is straightforward in its design and in the connector logic required to process it. Typically, however, application entities are more complex and include information that is stored in other objects.

Representing multiple entities

A business object can represent application entities that include data from other entities in one of the ways shown in Table 2..

Table 2. Representing multiple entities.

Structure of business object	Type of data organization	Type of parent/child relationship
Parent business object can have one or more child business objects that represent the other entities.	One-to-one One-to-many	Structural

Table 2. Representing multiple entities. (continued)

Structure of business object	Type of data organization	Type of parent/child relationship
Parent business object can have one or more foreign-key attributes that reference other top-level business objects that represent the other entities.	One-to-one One-to-many Many-to-many Many-to-one	Semantic
If the application and its interface permit, a flat business object can include attributes that directly reference other entities.	One-to-one	None

When deciding how to structure business objects that represent multiple entities, consider these guidelines:

- If the relationship between entities is a one-to-many relationship, represent the data in the subordinate entities as child business objects. For example:
  - When working with database tables, if an entity row is related to one or more rows in another entity and is the only entity that relates to the subordinate entity, create a separate child business object for each related row.
  - When working with a DTD, if an XML element has an attribute with a cardinality of `*`, create a separate child business object for each related element attribute.
- If the relationship between entities is a many-to-many relationship, represent the data in the related entities as top-level business objects that are referenced by the parent rather than contained by the parent.
- If a business object definition for an entity includes many attributes from another entity and the attributes from the second entity form a logical grouping, you may want to create a child business object definition for the second entity rather than locate all the attributes for both entities in the same business object definition.
- If an existing business object already contains other child business objects, creating one or more child business objects that represent new entities makes the business object structure consistent.

The following sections describe each of these representations is described in more detail.

Structural relationships

In a structural relationship, the parent business object physically contains the child business object. Such a business object is a hierarchical business object: at least one of its attributes is complex (that is, it contains either a child business object or an array of child business objects). The Relationship attribute property for this attribute is `containment`, to indicate a containment relationship. The type of this attribute is the type of the child business object (or objects) it represents. For more information, see "Hierarchical business objects" on page 9.

The following hierarchical business objects represent structural relationships:

- Because an order is composed of line items, an Order business object contains an array of LineItem business objects. The containment relationship has multiple-cardinality because each order can contain multiple line items. This structure represents a one-to-many relationship.
- Because an employee is associated with a single home address, an Employee business object contains a single Address business object. The containment

relationship has single-cardinality because each employee can be associated with only one home address. This structure represents a one-to-one relationship.

In both cases, because the parent business object contains the child or array of children, the relationship is defined structurally.

A structural relationship assumes that the parent business object owns the data within the child object. Thus, when a new employee is created, a new row is inserted into the address table to hold that employee's address. Similarly, when an employee is deleted, the employee's address is also deleted from the address table.

Semantic relationships

In a semantic relationship, either the parent business object references the child, or the child references the parent. When one business object references another, it stores a value that uniquely identifies the other, but it does not contain the other. In this case, the component that processes the business object derives the relationship semantically.

A semantic relationship is typically defined by a simple attribute that serves as a *foreign key*. The foreign key attribute is located in one business object and contains the unique identifier (called the *primary key*) of the other. In other words, both business objects have a primary key attribute that holds its unique identifier. In addition, one of the business objects also has a foreign key attribute that holds the primary key value of the other. The foreign key establishes the link semantically between parent and child.

Semantic relationships are important when there is a many-to-many or many-to-one relationship between entities, in other words, when more than one parent has a relationship to the same child. Relating the entities semantically rather than structurally isolates the child's data, which is important to maintain data consistency.

Because the parent does not contain the child in a semantically defined relationship, the connector that handles requests for the parent and child receives them in separate operations. In other words, the requests are sent separately to the connector, which handles the parent and child in separate operations. For more information, see "Data ownership in relationships" on page 21 and "Choosing between a semantic and a structural relationship" on page 22.

Consider the design options in Table 3 for specifying a semantic relationship.

Table 3. Design options for semantic relationships.

Design option	Type of relationship
"Storing the foreign key in the parent object"	One-to-one Many-to-one
"Storing the foreign key in the child object" on page 17	One-to-many
"Storing foreign keys in an array of child objects" on page 18	One-to-many Many-to-many
"Storing the foreign key in a business-object tree" on page 19	One-to-one

Storing the foreign key in the parent object: In the simplest use of foreign keys, the foreign key that establishes the relationship is stored in the parent. In this case, a parent can contain a reference to only one child of a given type. The relationship

between parent and child is clearly defined in the parent. Therefore, this structure represents a one-to-one relationship. However, multiple parent business objects can reference the same child business object to implement a many-to-one relationship.

Note: When the foreign key that establishes the relationship is stored in the parent, a parent can contain multiple attributes that each contain a reference to a child, but each of these attributes typically references a different type of child.

In Figure 9, the Customer business object has two attributes (`AddressId` and `CustInfo`) each of which contain a reference to a child business object. The foreign key attributes in Customer readily identify the parent's relationship to the two children.

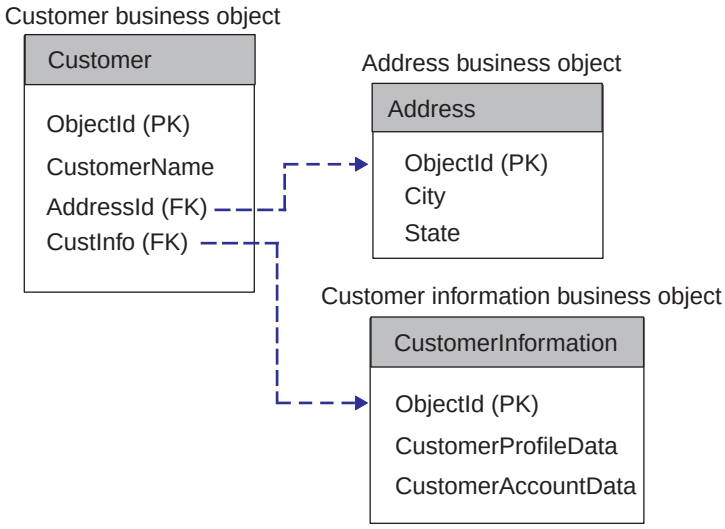


Figure 9. One-to-one: Multiple foreign key attributes stored in the parent.

Note: In Figure 9, the acronym "PK" is used to indicate a primary key and "FK" is used to indicate a foreign key. In addition, these business objects follow the naming convention for generic business objects by naming their primary key attribute `ObjectId`. In an application-specific business object, it is usually best to name the attribute after the name of its equivalent field or column in the application.

Figure 11 on page 19 illustrates an example of a generic business object, named Order. It is a parent business object that stores a foreign key reference to another object. Order contains the CustomerId attribute, which references a top-level generic Customer business object.

Storing the foreign key in the child object: Alternatively, the foreign key that establishes the relationship can be stored in the child. This case represents a one-to-many relationship; that is, multiple children can reference the same parent. However, because the relationship between parent and child is defined in the child, the relationship is invisible when you examine only the parent. Therefore, if the parent business object triggers an integration flow, those children cannot be retrieved—there is no reference to them in the parent business object that is traveling through the system.

In Figure 10, the foreign key attribute is stored in each child business object rather than in the parent. This structure allows multiple children to be semantically related to the same parent. In this case, however, because the parent business object has no attributes that contain a reference to a child business object, there is no way to identify the parent's relationship to its children or, given the parent, to retrieve all of its related children.

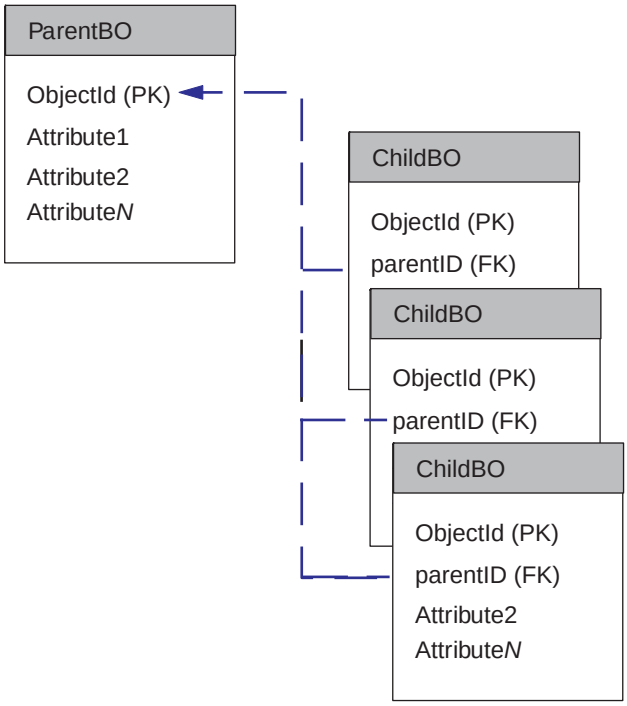


Figure 10. Many-to-one: Foreign key stored in multiple children.

Note: In Figure 10, the acronym "PK" is used to indicate a primary key and "FK" is used to indicate a foreign key.

Storing foreign keys in an array of child objects: To represent a one-to-many relationship, the foreign key that actually establishes the relationship is stored in a simple attribute in a child business object. The parent business object structurally contains an array of these children. In other words, the parent contains an array of child business objects, each one of which contains a foreign-key reference to another top-level business object. In addition, multiple parent business objects can reference the same child business object in their child business object arrays to implement a many-to-many relationship.

Figure 11 illustrates a parent object, the Order business object, that stores a foreign key reference to another object. The Order business object contains a reference to a single Customer business object and structurally contains an array of ContactRef business objects. Each ContactRef business object contains a reference to a single Contact business object.

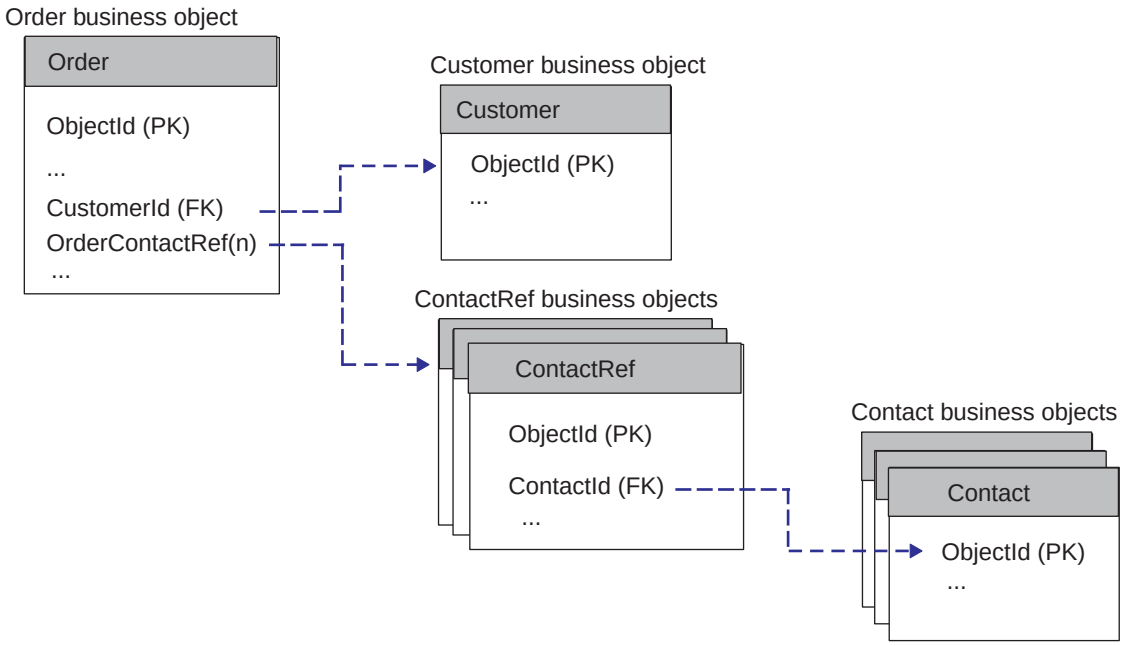


Figure 11. Parent containing a child that stores foreign keys.

Note: In Figure 11, the acronym "PK" is used to indicate a primary key and "FK" is used to indicate a foreign key.

Storing the foreign key in a business-object tree: In this design, the foreign key that establishes the relationship is stored in a "child" business object whose parent is another business of the same type as itself. Such a business object contains a ParentId attribute, which can contain a reference to another business object of the same type as the current business object, which is the direct parent of the current business object.

In Figure 12, the ParentId attribute of one SampleA business object contains a reference to the primary key (ObjectId) attribute of its immediate parent SampleA business object. The head of the hierarchy is the business object whose ParentId attribute does not contain a value.

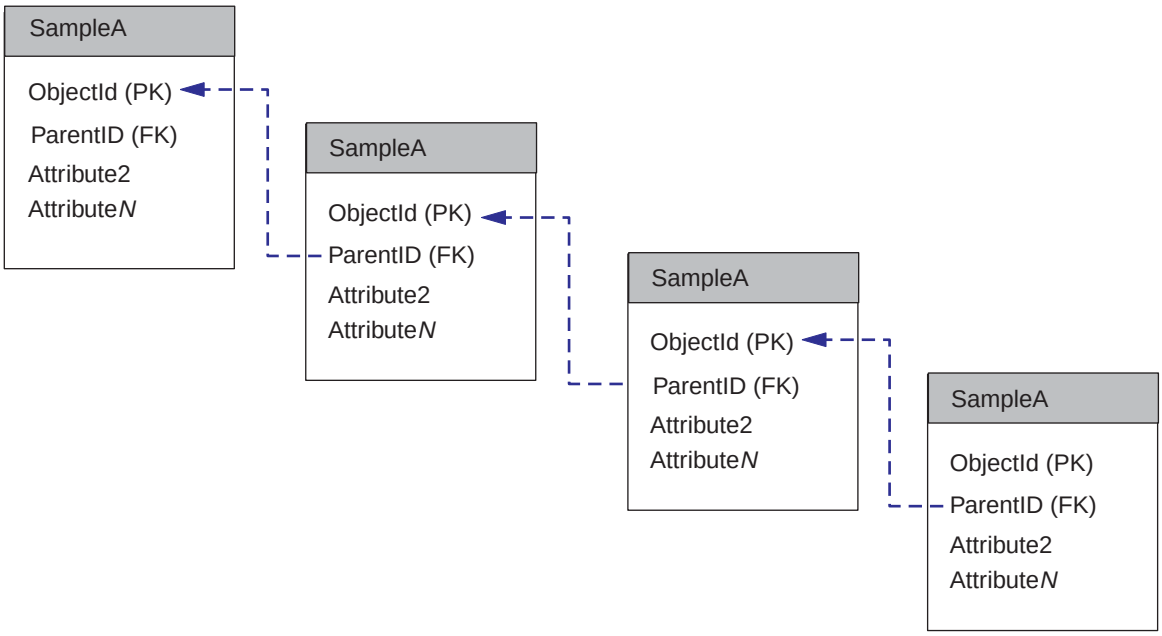


Figure 12. Business object storing a foreign key to its parent of the same type.

Note: In Figure 12, the acronym "PK" is used to indicate a primary key and "FK" is used to indicate a foreign key.

Flat business object representing related entities

If the application interface provides the capability of joining multiple application entities in one business object, you may be able to define a flat business object that contains attributes referring to a primary entity and to related entities. If the relationship between the entities is a one-to-one relationship, where one instance of the primary entity can be associated with one instance of each related entity, attributes from multiple entities can be included in a single business object.

When designing an application-specific business object of this type, you may need to use application-specific information to specify the location of attribute data in the application so that the connector can find and process the data correctly.

Figure 13 provides an example of a flat WebSphere business integration system business object that represents data in two entities, one a table containing address data and the other a table containing lookup data for state/province and country abbreviations.

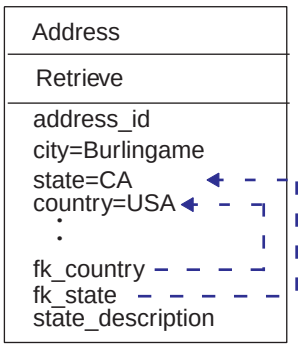


Figure 13. Flat business object that represents two entities.

This example uses application-specific information to establish a foreign key relationship between the entities. In this case, the connector performs a lookup from a value in an attribute that represents one table to provide a value for an attribute that represents another table. To retrieve this data, the connector performs two table reads.

Although flat business objects can encapsulate information from or included in multiple application entities, cross-application integration problems often require more complex integration logic and more complicated data structures than flat business objects can represent. To handle more complexity in application entities and integration requirements, the WebSphere business integration system provides hierarchical business objects.

Design considerations for multiple entities

This section provides the following considerations when you design business objects for multiple entities:

- "Data ownership in relationships"
- "Choosing between a semantic and a structural relationship" on page 22

Data ownership in relationships

The way you design your business objects to represent multiple entities has an effect on the ownership of the data:

- A structural relationship assumes that the parent business object owns the child data.
- A semantic relationship does *not* assume that the parent business object owns the data within the child object.

This distinction is significant when considering the data consistency of an entity that is shared by multiple business objects.

For example, assume that a customer and a contact share an address. If the Customer and Contact business objects contain a reference to the Address business object (a semantic relationship) instead of containing the business object (a structural relationship), changes to the Address can be made independently of changes to the Customer or Contact.

However, if the Customer and Contact business objects each contain the Address business object, changes to the Address made by Customer might overwrite changes made by Contact. In this case, two different collaboration objects (CustomerSync and ContactSync) might update the same address data at the same time, causing data inconsistency.

If Customer and Contact have a semantic rather than structural relationship to the Address business object, you can limit modification of Address data to a third interface. For instance, you might have one interface for each of the Contact and Customer business objects. Then both of those interfaces could delegate management of Address business objects to a third interface. This is done by having the CustomerSync and ContactSync collaboration objects call AddressSync through a wrapper collaboration object rather than directly making the changes themselves.

Figure 14 illustrates the difference between semantically and structurally defining the relationship to a child business object.

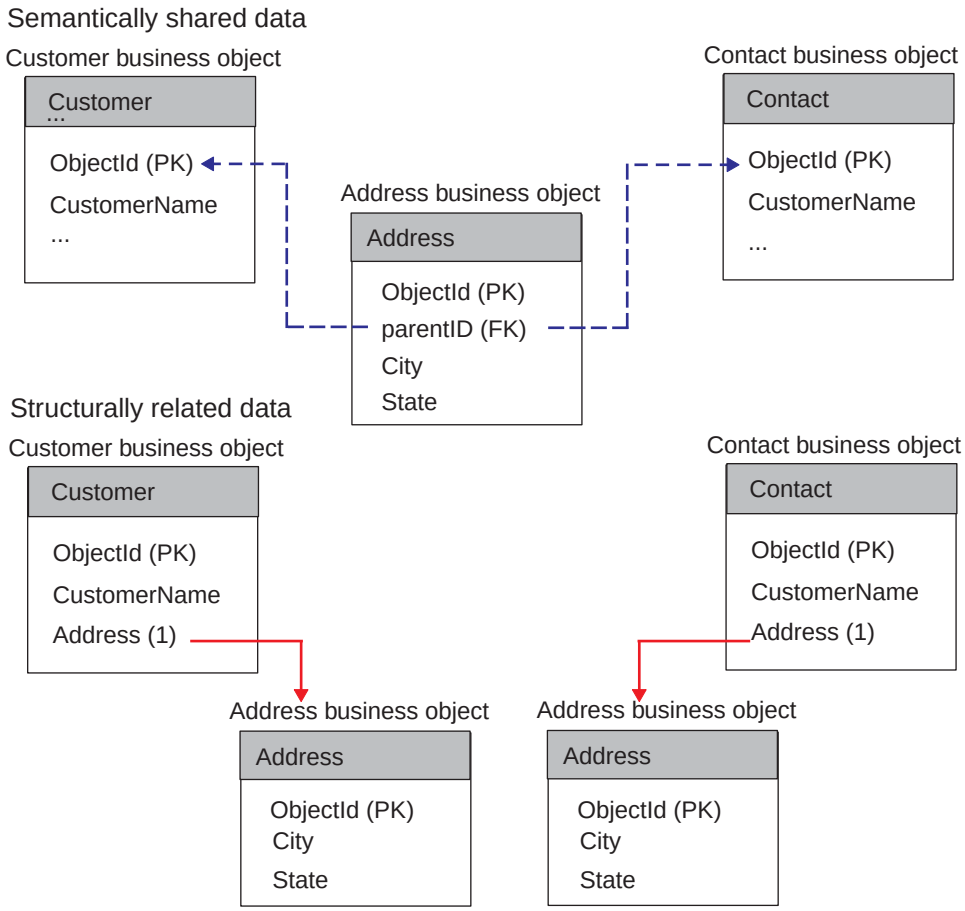


Figure 14. Comparing semantic and structural relationships.

The figure above illustrates two kinds of relationships to child data:

- Semantic—The child Address business object, which is semantically linked to both Customer and Contact, contains the value of its parent's primary key in a simple foreign-key attribute. In this case, the name of the primary key attribute in both parents is the same, which simplifies the link from child to parent.
- Structural—The two business objects that structurally link to Address have an attribute that represents an instance of the child. In this case, the data in the child is related only to the parent that contains it and is not shared.

Choosing between a semantic and a structural relationship

As Table 2 on page 14 shows, both the one-to-one and one-to-many relationships can be represented by a structural or semantic relationship. Table 4 on page 23

summarizes these structural and semantic representations.

Table 4. Representations for one-to-one and one-to-many relationships

Type of relationship	Structural representation	Semantic representation
One-to-one (single cardinality)	An attribute in a parent business object represents a single child business object.	An attribute in a parent business object is simple and contains the foreign key to reference one child business object.
One-to-many (multiple-cardinality)	An attribute in a parent business object represents an array of child business objects.	Multiple child business objects each contain a foreign key attribute that stores the parent's primary key.

Figure 9 and Figure 10 illustrate business objects whose single- and multiple-cardinality relationships are defined semantically. The business objects in the example might represent data stored in a database. Relationships between business objects that represent such data can be defined both semantically and structurally. For such data, the relationship between a parent and child can be defined both semantically and structurally in the same two business objects.

Choosing a semantic relationship: To implement a semantic relationship, the underlying application should be able to support foreign keys. For example, when a business object represents database data, it can establish the relationship between entities both semantically and structurally. Such business objects are designed redundantly. In other words, the component that processes them can locate the child through the parent and the parent through each child.

For example, assume an application has a table that represents purchase orders. This table is related by foreign keys to a table that contains line items for a purchase order. Multiple rows in the line items table reference a single row in the purchase orders table. Figure 15 illustrates these tables.

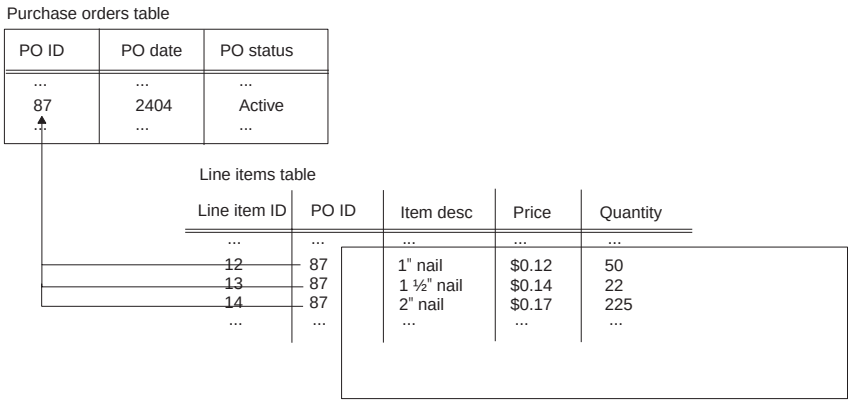


Figure 15. Example application tables with a one-to-many semantic relationship.

Figure 16 illustrates business objects that might correspond to these tables. This figure shows a top-level PurchaseOrder business object and three child LineItem business objects.

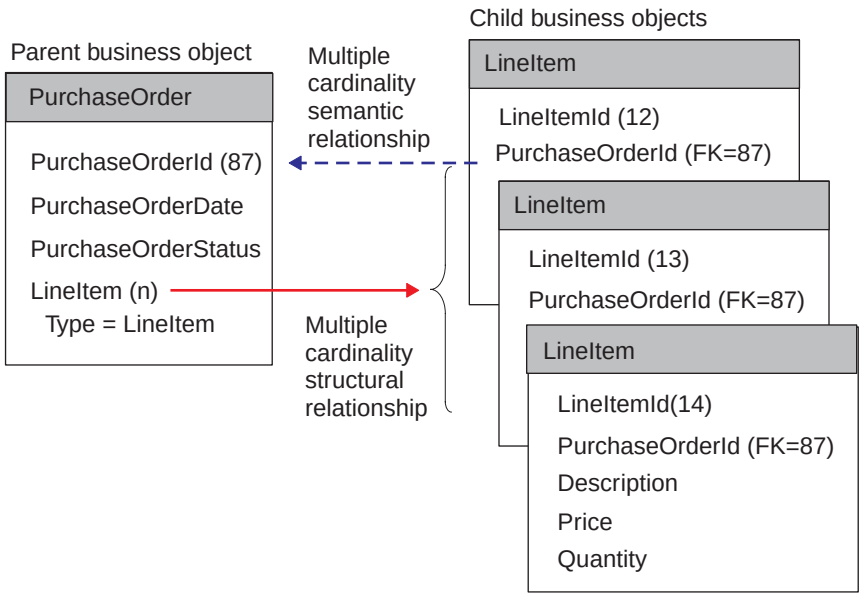


Figure 16. Sample business objects with multiple-cardinality semantic and structural relationship.

The illustrated PurchaseOrder business object has both a semantic and structural relationship to its LineItem children. The PurchaseOrderId attribute in each child creates the foreign-key semantic link to the parent from the child. The LineItem attribute in the parent, which is defined with `cardinality n`, creates the structural link to the child from the parent.

Note: IBM does not deliver any business objects that have the foreign key stored in the child. This document presents the above example only to illustrate different ways to link parent and child data.

Choosing a structural relationship: If the underlying application does not support foreign keys, you probably need to implement a structural relationship. For example, a DTD, which represents a single XML document does *not* support foreign-key information. Therefore, any one-to-one or one-to-many relationships must be defined structurally. The following Order DTD, which contains elements that correspond to an application `Order` entity, illustrates single- and multiple-cardinality relationships:

```
<!--Order -->
<!-- Element Declarations -->
<!ELEMENT Order (Unit+)>
<!ELEMENT Unit (PartNumber?, Quantity, Price, Accessory*)>
<!ELEMENT PartNumber (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
<!ELEMENT Accessory (Quantity, Type)>
<!ATTLIST Accessory
Name CDATA >
<!ELEMENT Type (#PCDATA)>
```

Figure 17 illustrates a business object that represents the Order DTD. The top-level business object contains the Order business object with a single-cardinality

relationship, and Order contains the child Unit business objects with a multiple-cardinality relationship. In turn, Unit contains the Accessory business objects with a multiple-cardinality relationship.

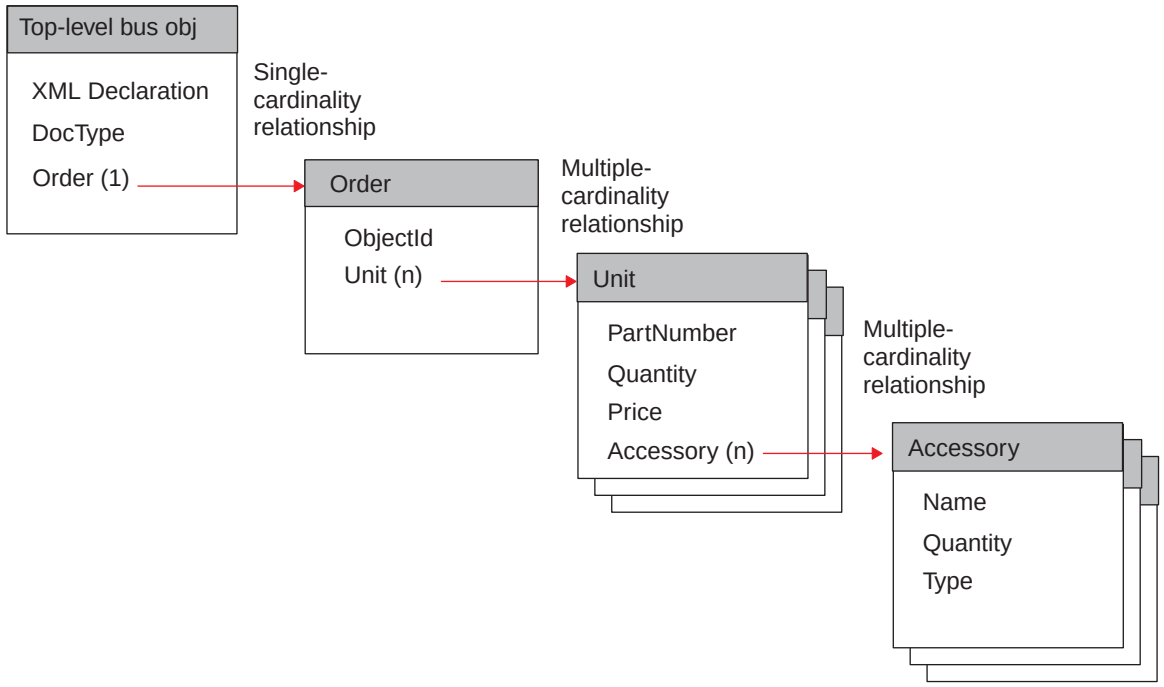


Figure 17. Single- and multiple-cardinality structural relationships.

The relationship of business objects illustrated in Figure 17 is defined structurally; that is, each parent business object contains an attribute whose type is the same as the child's and whose relationship is specified as `containment`.

Important: The XML data handler has specific requirements of the top-level business object that represents a DTD. For information about these requirements, see the *Data Handler Guide*.

Designing application-specific business objects

An application-specific business object contains data, actions to be performed on the data (verbs), and information about the data (application-specific information). Many connector methods pass an application-specific business object as an argument. For example:

- When an application event occurs, some connectors invoke a data handler to convert the data's format into a business object, which the connector sends to the integration broker.
- When an integration broker sends a request to a connector, the connector framework sends the business object as an argument to the connector's business object handler. In this case, some connectors invoke the data handler to convert the information in the business object to the format used by the application, which enables the connector to perform operations in the application.

Designing the relationship among the connector, the data handler, and their supported application-specific business objects is one of the tasks in connector and data handler development. Because application-specific business object design can generate requirements for connector and data handler programming logic that

must be integrated into the connector development process, the developers of the connector, data handler, and application-specific business objects must work together to develop specifications for those components. The layout and design of an application-specific business object should be determined by the connector or data handler that processes it.

This section covers the following topics:

Contents of application-specific business object definitions

A business object definition includes the following information:

Table 5. Contents of a business object definition.

Contents of business object definition	Description	For more information
Business object structure	The structure of an application-specific business object is typically designed to correspond closely to the application entity (data structure) at the level that the connector or data handler interacts with the application (such as at the table level, the API level, or at different levels within an API).	"Structure of application-specific business objects" on page 26
Attribute properties	Attributes contain individual pieces of data within an application entity. They also have properties that provide information such as the data's type, cardinality, and default value. Attribute properties also specify whether the attribute is required or key.	"Attributes in an application-specific business object" on page 27
Application-specific information	An application-specific business object definition often includes text strings that tell the connector how the business object is represented in the application or how to process it.	"Business object application-specific information" on page 28

Structure of application-specific business objects

The way a connector or data handler processes business objects is determined in part by the structure of the business objects that it supports. As you design the structure of an application-specific business object, you need to determine what structure best represents a particular application entity and how this structure affects the design of connector and data handler logic or how the structure is processed by an existing connector or data handler.

While a goal of connector and data handler design is to code a connector or data handler so that it can handle new and changed business objects without modification, it is difficult to create a connector or data handler that can handle any possible business object.

Typically, a connector or data handler is designed to make assumptions about the structure of its business objects, the relationships between parent and child business objects, and the possible application representation of business objects. If

designing for an existing connector or data handler, your task is to understand these assumptions and design business objects accordingly.

A beginning set of questions to consider about the structure of an application-specific business object is:

- What is the organization or database schema for the application entity that will be encapsulated in the business object. Does the application entity represent hierarchical data or one-to-many relationships?
- Does a business object represent one application entity or more than one application entity? In other words, can attribute values in an individual business object be stored in different application entities?
- What kind of relationships between business objects does the connector or data handler handle? How are the relationships modelled in the business objects or how are they processed by the connector or data handler?

For more information about the business object structures that can represent single or multiple application entities, see "Determining business object structure" on page 13.

Note: Certain connectors may require the top-level business object to contain specific information. For example, the XML connector requires its top-level business object to contain simple attributes for a URL, MIME type, and business object prefix, as well as complex attributes to contain a request business object and a response business object. If you are designing a business object for an existing connector, refer to its adapter user guide for specific structure requirements. For more information, see "Designing for an existing connector or data handler" on page 32.

Attributes in an application-specific business object

The attributes hold the individual pieces of data in an application entity. When defining attributes for an application-specific business object, consider these questions:

- What piece of data in the application entity will each attribute represent? For a business object representing a database entity, will each attribute represent a field, such as a table column? For a business object representing an XML document, will each attribute represent an element?
- Is it necessary to create an attribute for every single field in the application entity? Some pieces of data in one application entity may not be significant to the other applications in the integration; by leaving them out of the business object definition you can reduce the complexity of the design and prevent the transfer of unnecessary data from decreasing performance.
- Will the application-specific business object have fewer simple attributes than the application entity? For example, do you need an attribute for every database table column?
- How will the connector operate when the individual business object has more simple attributes than the corresponding database table has columns or the corresponding DTD has tags? In other words, some attributes in the business object are not represented in the database or the DTD. In most cases, these attributes convey information about specific access mechanisms or are used to separate attributes that represent child business objects. The connector or map may employ special logic that requires connector-specific attributes to handle certain application-specific business objects.

As a rule, keep the structure of the business object the same as the structure of the corresponding application entity (such as database tables or DTDs). If the business object is large (contains many attributes), define only the attributes that are used in the business process for which you are designing the business object. However, if the business object is small, define all of the attributes to be available for future use. The number of attributes you define depends on the size of the business objects and the complexity of the relationships between them.

In addition to identifying which application entities must exist as attributes in the business object, you should also examine the business process to determine if any additional attributes are required. As part of the analysis of the business process, identify the business object's requirements. Stepping through a business process reveals how a business object is handled and how the required attributes are used. Variations in the business process and the handling of exceptions might identify additional attributes that are required to process the business object. These additional attributes might not correspond to data that is retrieved or updated in the application.

For example, you may need attributes that:

- serve as a priority indicator whose value is derived during processing according to the value of an attribute value
- serve as lookup that contains routing information based on one or more attribute values

Business object application-specific information

After you have defined the structure of an application-specific business object definition and defined the set of attributes that the business object definition contains, you can determine whether the connector or data handler needs additional information about how to process the business object to enable it to handle the requests it receives from the integration broker. The business object definition can include this additional information in application-specific information.

Application-specific information provides the connector or data handler with application-dependent instructions on how to process business objects. The recommended approach to designing the relationship between business objects and connectors is to store information in the business object definition that helps a connector interact with an application or data source. Such information, called **meta-data**, can be specified in the application-specific information of each business object, business object attribute, and business object verb.

The application-specific information is a string that is entered during business object design and read at runtime by a connector or data handler. The connector or data handler uses the meta-data in the business object definition to process business object instances. Because the connector or data handler has access to its supported business object definitions at runtime, it can dynamically determine how to process a particular business object.

Consider the following advantages and limitations of application-specific information when designing a business object:

- Application-specific information enables a business object to be self-contained with all the information required to process it.

The application-specific information in the business object definition can include table and column names, processing instructions, names of functions that the connector calls, or other information about how to process the data in the application.

Because an application-specific business object contains all the information needed to process it, the connector can handle new or modified business objects without requiring modifications to the connector source code. The connector can be written in a generic manner, with a single business object handler that does not contain hard-coded logic for processing specific business objects.

- A meta-data-driven connector can build application function calls or SQL statements from the values in a business object instance and the application-specific information in the business object definition.

  The function calls or SQL statements perform the required changes in the application database for the business object and verb the connector is processing.
- The application that a business object represents determines how much application-specific information the business object definition can contain.

  Depending on the application and its programming interface, a connector and its business objects might be designed so that the connector is almost entirely driven by the application-specific information in its business objects. In this case, the connector may require only one business object handler to transform business objects into requests for application operations.

  For some applications, however, the application interface may have constraints that force entirely different processing logic for different business objects and, therefore, the implementation of multiple business object handlers. For these applications, only a partially meta-data-driven implementation or no data-driven implementation is possible.

  Depending on the application, business objects vary in how much application-specific information they contain. Most application-specific business objects, however, can be designed to contain some information that assists the connector or data handler with business object processing.

Suggested format of application-specific information: It is recommended that you use name-value pair syntax when you define application-specific information. This syntax specifies the name of the property and its associated value, separated by a equals sign (=), as the following syntax shows:

```
name1=value1;name2=value2
```

For example, the following name-value pair defines a "table name" property:

```
TN=TableName
```

Name-value pairs allow values to be specified in random order. The connector evaluates the name of each parameter before interpreting the value. It is recommended that you separate name-value pairs with a delimiter that:

- defaults to a semicolon (**;**)
- is configurable

Note: If you are creating a business object for an existing connector, check its adapter User Guide to determine the syntax that it requires. Not all connectors may default to use a semicolon as a delimiter, or be configurable in that respect.

Table 6 on page 30 provides examples of parameters that might be included in an attribute's application-specific information. These parameters would be relevant

only to a business object that represents data in a database table.

Table 6. Example name-value parameters for attribute application-specific information.

Parameter	Description
TN=*TableName*	The name of the database table.
CN=*col_name*	The name of the database column for this attribute.
FK=[..]*fk_attributeName*]	The value of the Foreign Key property defines a parent/child relationship.
UID=AUTO	This parameter notifies the connector to generate the unique ID for the business object and load the value in this attribute.
CA=*set_attr_name*	The Copy Attribute property instructs the connector to copy the value of one attribute into another. If set_attr_name is set to the name of another attribute within the current individual business object, the connector uses the value of the specified attribute to set the value of this attribute before it adds the business object to the database during a Create operation.
OB=[ASC|DESC]	If a value is specified for the Order By parameter and the attribute is in a child business object, the connector uses the value of the attribute in the ORDER BY clause of retrieval queries to determine whether to retrieve the child business object in ascending order or descending order.
UNVL=*value*	Specifies the value the connector uses to represent a null when it retrieves a business object with null-valued attributes.

A single attribute's application-specific information might combine several of the example parameters listed above. This example uses semicolon (;) delimiters to separate the parameters:

```
TN=LineItems;CN=POid;FK=..PO_ID
```

The application-specific information in this example specifies the name of the table, the name of the column, and that the current attribute is a foreign key that links the child business object to its parent.

Content of application-specific information: The content of application-specific information can vary considerably in complexity. Some examples are:

- The application-specific information in a business object definition can encode the name of the table that the business object corresponds to, and, for each attribute, it can encode the name of the column that the attribute corresponds to. This is a relatively simple implementation of application-specific information, but it may be all that a connector needs.
- A more complex implementation of application-specific information might contain a set of parameters that specify how the connector handles various business object operations.
- At its most complex, application-specific information might include conditions, direct connector transaction processing, specify methods of data retrieval, and provide preprocessing capabilities.

If the business object definition includes application-specific information and the connector has been designed to make use of it, the connector can extract the content of the application-specific information from the business object definition and use it for processing.

Example: How a connector processes application-specific information: As an example of how a connector processes application-specific information, assume that your application is based on tables and that you want to work with an application table called CURRENTCUST, which stores information on customers. This table has two columns: CSTName and CSTCity.

In the AppSpecificInfo property of the business object header, you can store the table name. In the AppSpecificInfo property of each attribute, you can store the column names. In addition, because the connector for this application uses SQL statements to interact with the database, you can design the verb application-specific information to hold SQL verbs and keywords. Figure 18 illustrates how this Customer business object definition might look.

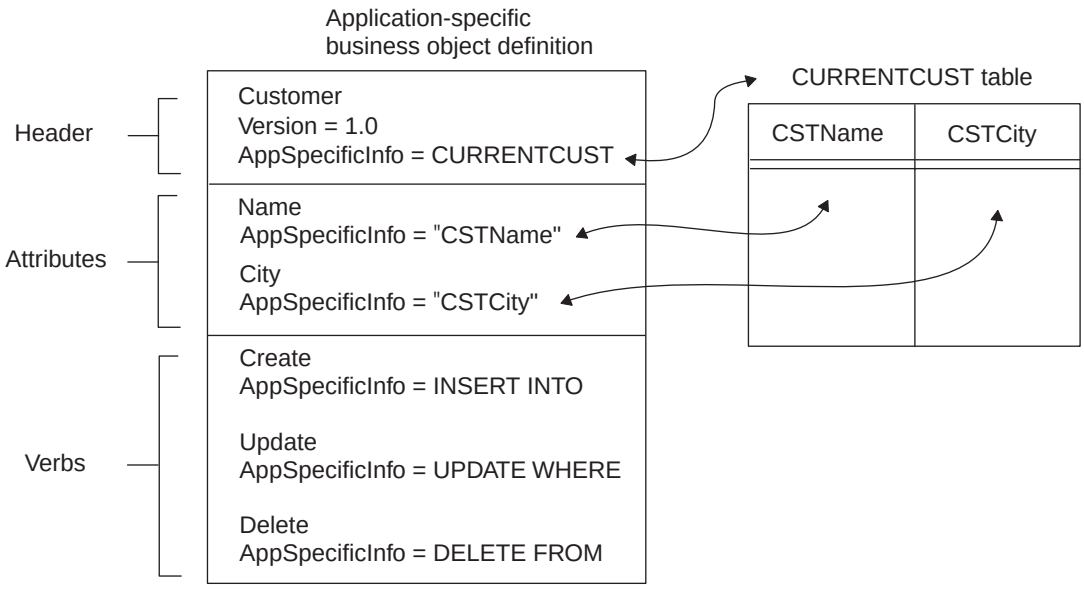


Figure 18. Application-specific information in a business object definition.

When a meta-data-driven connector receives an instance of this business object from an integration broker, it extracts the table name and column names from the application-specific information properties in the business object definition, and then extracts the attribute and verb values from the business object instance. Using the table and column names, the attribute values, and the SQL keywords in the verb application-specific information, the connector can build an SQL statement.

Figure 19 shows an example of this type of processing. The connector extracts the verb processing instructions and the table and column names from the business object definition. It then gets the attribute values from the business object instance. Using this information, the connector builds an SQL INSERT statement to update the CURRENTCUST table with the new information.

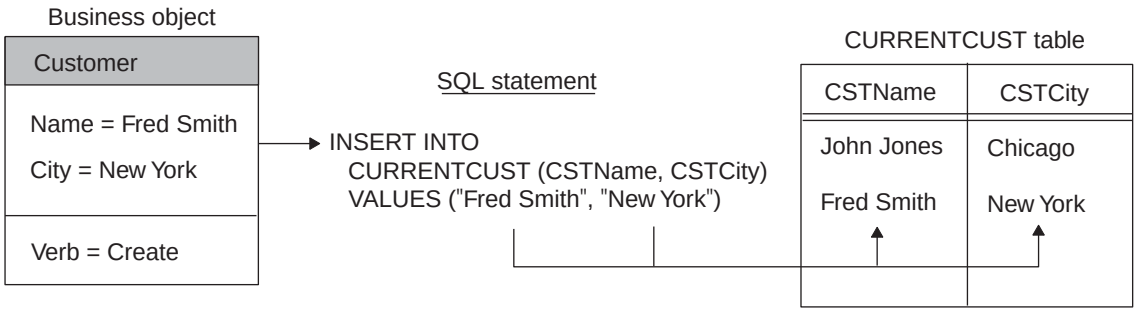


Figure 19. Using application-specific information to build an SQL statement for a Create operation.

As mentioned above, a business object definition can include AppSpecificInfo text for the business object as a whole, and for its attributes and verbs. The following sections provide more information on the use of application-specific information in these components of a business object.

Important
The length of application-specific information is restricted to 1000 characters.

Figure 5 uses the attribute-level AppSpecificInfo property to store the name of the Invoice subform and the attribute's corresponding field name. The example uses name-value pairs to specify the information.

Tips on designing application-specific information: When designing business objects to maximize the meta-data-driven behavior of a connector, follow these general recommendations for storing application-specific information in the business object definition:

- Store entity names, such as table or form names, in the business object-level AppSpecificInfo property of a top-level business object. Store subform names or table names in the business object-level AppSpecificInfo property of a child business object.
- Store field names, column names, and other information related to business object attributes in the attribute AppSpecificInfo property.
- Store verb processing instructions in the verb `AppSpecificInfo` property.

The careful use of the `AppSpecificInfo` property enables a connector to handle a variety of business objects in the same way. If an application is consistent in how it handles data operations, and if for all operations the connector performs consistent tasks, the business object can be designed to enable a completely meta-data-driven connector.

Designing for an existing connector or data handler

If you are designing an application-specific business object for an existing connector or data handler, your first step is to consult its adapter User Guide for its requirements on specifying application-specific information and using business object handlers. Keep the following points in mind when designing business objects for an existing connector or data handler:

- To determine if there is an available Object Discovery Agent, check the adapter User Guide for the connector and the documentation for the data handler that

will process your business object. Using the Object Discovery Agent can greatly facilitate the business object design effort, particularly when the entity involved is large.

- Determine whether there is an existing business object available that models the application entity, such as a sample. Determine whether the effort to customize the existing business object is less than creating an entirely new one and if so then consider using the sample business object.

  Important
  IBM does not support sample business objects, but they can be very useful as a starting point for business object design.

- Whether you use an Object Discovery Agent or an existing business object, it is still important to examine and confirm all the data definition requirements, such as the object key, foreign keys, child business objects, default values, data types, and size limitations. The following factors result in this requirement:
  - Object Discovery Agents can facilitate the design effort, but cannot discover all of the requirements surrounding an application entity.
  - With existing business objects, the threat is that applications can be installed and configured different ways to accommodate customer-specific needs. A business object that accurately models an entity in one application installation may not accurately model that entity in another installation of that application.

While designing the application-specific business object, keep in mind that its primary role is to model the entity in the data source. It is also important to identify how its associated connector or data handler handles its processing, and what are the requirements of the business process in which it participates.

Designing generic business objects

A *generic business object* reflects a superset of information that represents entities used by multiple diverse applications or programmatic entities. Collaborations use generic business objects so they can provide information for a variety of diverse applications. Therefore, designing generic business objects is part of the task of collaboration development. When designing a generic business object, take into account the following:

- Understand the data requirements that are critical to the business process integration.
- Study the business logic that the business object participates in, and all requirements based on that logic.

  The following two considerations illustrate the complexity of business logic analysis:

  - Processing prerequisite data

    Sometimes the application that triggers the collaboration object does not provide all the data required to process the triggering business object. The additional data may reside in other applications, including the destination application.

    For example, a Sales Force Automation (SFA) application (such as Siebel) may generate a quote that needs to be logged as an Order in an order management system (such as SAP). However, before the Quote can become an Order, it may require additional information not available in the SFA

application. For example, an Order may require such additional data as customer credit status (from a financial system), contact information (from a customer support system), or Availability To Promise information (from a warehousing system).

When you design the generic Order business object, you may have to include attributes and design a structure that supports data which is not present in the source application necessarily, but may be present in other applications involved in the interface.

– Cross-referencing between individual application entities

Determine how and whether individual application entities correspond to each other or are cross-referenced to each other generically in the business process.

For example, a customer in Oracle is represented as a Customer whose address is represented as an Address. A customer in SAP is represented as a "SoldTo" entity whose address is represented as a "ship-to" entity. A customer in Clarify is represented as a "Business Organization" whose address is represented as a "Site".

Study the functionality and relationships between an application's entities to determine the business objects and processes involved in integrating data between applications.

- Understand what required data is common or shared across all applications participating in the business process and that is critical to the integration of that process. The attributes and their relationships should determine at a minimum the mandatory attributes (the lowest common denominator of attributes) and the transformations that the business integration system must perform between these application-specific business objects.

  Consider the following integration possibilities:

  – The business process integrates data from an Enterprise Resource Management (ERP) system to a Customer Relationship Management (CRM) system. In this case, the business object probably does not need to be very complex because most CRM systems do not accommodate most of the data stored in an ERP system.

  – The business process integrates data between two ERP systems. In this case, it is likely that the business object will be highly complex.

  – The business process integrates data from a CRM system to an ERP system. In this case, your design must reflect how much of the data actually originates in the CRM system (and thus must be represented by attributes in the generic business object) and how much of it can be defaulted in the destination application itself (and thus provided as default values in the application-specific business object).

- If they exist, study the application-specific business objects to which the generic business object will map. Analyze the structure and the attributes of all business objects to derive a generic business object that is suitable to all applications.
- Consider whether there is a standard for the type of business object you are designing. For instance, there might be an appropriate model for the entity provided by the Electronic Data Interchange (EDI), Open Applications Group (OAG), or Object Management Group (OMG) initiatives.

Generic business object design standards

To be consistent with IBM-delivered generic business objects, use the following standards when designing a generic business object:

- The first attribute of every object should be its key and should be named `ObjectId`.
- If an attribute represents a foreign key, its name should concatenate the name of the foreign business object and `Id`; for example: `CustomerId`, `ItemId`, and `OrderId`.
- Be consistent. If you use an abbreviation in an attribute name, use the same abbreviation in parent and child business objects. If possible, use the same abbreviation for all relevant attribute names. For example, if you abbreviate `Number` to `Num`, do so consistently.

Designing for event isolation

When designing a generic business object, it is recommended that you consider the needs of event isolation.

To prevent more than one collaboration object from updating the same data at the same time, each business object should be modified by only one type of collaboration object. In other words, a Customer business object should be modified only by a CustomerSync collaboration object.

If a collaboration object modifies a business object that contains a child business object, and the child business object is also contained by a different top-level business object that has its own modifying collaboration object, design the top-level business objects to contain the child semantically rather than structurally. Develop a third collaboration object to modify the shared child. The collaboration objects that own the two top-level business objects should then delegate processing of the shared child to the third collaboration object.

For example, if both Customer and Contact business objects contain the same address data, design the Address business object as a top-level business object that is referenced by Customer and Contact, but not contained by them. Then develop a separate Address collaboration object to modify address data.

In another example, however, if the Order business object is the only business object that modifies OrderLineItem data, you can design Order to contain the OrderLineItem child business objects rather than merely reference them.

In other words, design the Customer and Contact business objects so that they contain a foreign-key attribute that references the Address business object, that is, that contains only the key value for Address. Do not design them to contain an attribute that represents a full-valued Address business object. But design the Order business object to contain an attribute that represents a full-valued OrderLineItem business object.

Note: Designing shared business objects as referenced rather than contained can simplify business object distribution. If the same child business object is defined in multiple business object definitions, the `repos_copy` utility attempts to load the same business object twice during installation, causing rollback. For information on `repos_copy` flags that change this default behavior, see the *User Guide for WebSphere Business Integration Express for Item Synchronization*.

Attributes in a generic business object

When defining attributes for a generic business object, study the attributes of the application-specific business objects to which the generic business object will map. Consider these guidelines:

- Note the similarities between entities in the application-specific business objects's attributes. Define attributes for the generic business object that most simply match those in the application-specific business objects.
- Note the differences between entities in the application-specific business objects' attributes. If one application-specific business object splits data into multiple fields while another combines the same data into a single field, determine which design best simplifies mapping between the two application entities. For more information, see "Designing for an existing connector or data handler" on page 32.
- Consider requirements generated by the processing performed by the collaboration object. For example, if the collaboration object processes prerequisites as described in "Designing generic business objects" on page 33, ensure that the generic business object contains all attributes required to store the prerequisite data.
- Develop the generic business object and interface to accommodate the largest number of applications involved in the interface. For instance, if there are four applications involved in an interface and three of them encapsulate data in a child object but the fourth contains that data at the parent-level object, then design the generic business object so that it encapsulates the data in a child object as well—this results in mapping and other related tasks being that much easier.
- Take future development efforts into account; you might design a generic business object to accommodate data structures that will be required at a later point to minimize the effort and change impact at that time. Do not, however, significantly increase the scope of development for a future project that may never come to be.

In general, a generic business object definition should include attributes that capture all the data elements that are to be transformed among all the application-specific business objects to which the generic business object will map.

Names of the attributes should be as intuitive as possible. For example, if several applications refer to an entity as a `Customer` and one application refers to the same entity as a `Business Organization`, use the more common terminology to name the generic attributes.

Note: The name of an attribute can contain only alphanumeric characters and underscore (_).

Evaluating existing generic business objects

You might be able to facilitate development of a generic business object by copying and customizing an existing one.

To evaluate a generic business object, examine the data involved in the interface. A guideline is that if 80% or more of the data exists in a delivered generic business object, customize the existing object.

When performing this analysis, it is more important to look at the business object structure than the attributes. Attributes are relatively easy to add and remove, whereas structural or hierarchical changes can require much more effort.

If you decide to customize an existing generic business object, examine the business object definition to determine whether it is missing one or more desired attributes. Missing attributes become more apparent when you conduct the mapping design. If the generic business object requires one or more additional attributes, create a child business object that contains the additional attributes. Isolating custom attributes in child business objects can greatly facilitate future upgrades of IBM-delivered business objects.

If you embed custom attributes in an IBM-delivered business object, upgrading to a new version of the business object requires re-embedding those attributes in the new business object. Isolating the custom attributes in their own business object allows you to add a single attribute to the new IBM business object—the attribute that creates the relationship between the parent and the custom child business object. If you are customizing a hierarchical business object that requires additional attributes in both the parent and the child, create separate child business objects for each.

It is recommended that you name custom attributes and business objects in a way that readily identifies them. A simple convention is to add an `_x` suffix to each custom name. For example, if you create a custom child business object that adds attributes to the generic Order business object, name the child `Order_x`. Doing so allows alphabetic listing to keep related names together. If it is more important to identify custom business objects or attributes than to alphabetize the custom object with its generic object, add an `x_` prefix to each custom name.

Determining mapping requirements for business objects

When an application-specific business object has been designed to match an application entity, it may not match its corresponding generic business object. Therefore, you must create maps between the application-specific business object and the generic business object so that the application data can be transported across the WebSphere business integration system.

An application-specific business object may not need to include all the fields or columns or elements in an application entity. Use the functional requirements of the application and the business processes in which it participates to identify which attributes belong in the application-specific business object.

You can also examine the correspondence between the generic business object and the application entity. You may choose to include fields in the application-specific business object that correspond those in the generic, which allows these data elements to participate in the business process.

When designing the business object, note the differences between the application entity and the generic business object. These differences define what kind of data transformation needs to take place. You may need to design mapping to:

- Combine multiple fields in the application entity to fill one attribute in the generic business object
- Split a field in the application entity to fill multiple attributes in the generic business object
- Ignore a field that is present in the generic business object but that is not relevant to the application entity
- Handle differences in semantic or structural relationships between an application-specific business object and a generic business object

- Handle foreign key relationships and other types of relationships between application entities
- Establish associations between data, for example:
  - Establish a lookup association between data in non-key attributes, such as an association that transforms code values (for example, marital status or currency code) between applications.
  - Establish an identity association between data in business objects, such as an association that transforms the key attributes (for example, unique identifiers and product codes) between applications.

To assist with mapping and design concepts, the relationship among fields in a table, attributes in an application-specific business object, and attributes in a generic business object is shown in a highly simplified way in Figure 20. The differences between the application-specific business object and the generic business object are handled in mapping. If the business object has attributes that do not have a representation in the database, the connector can provide a default value for the attribute.

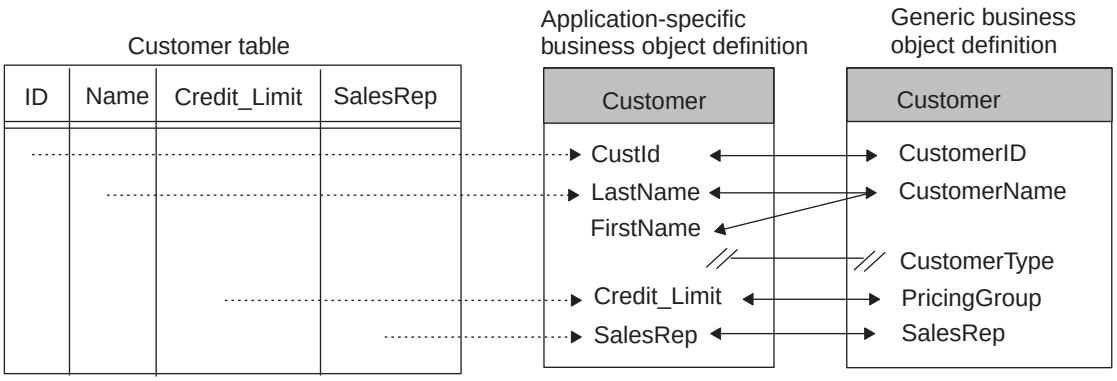


Figure 20. High-level view of field/attribute relationships.

For information on creating maps, see the *Map Development Guide*.

Chapter 3. Overview of Business Object Designer Express

The Business Object Designer Express tool is used to create, edit, and delete business object definitions. This chapter provides an overview of how to launch and use Business Object Designer Express. The main topics of this chapter are:

Working with projects

Business Object Designer Express uses the concept of a *project* to define a virtual work area in which business object definitions are created, modified, or deleted. Depending on your environment, the "project" to which Business Object Designer Express refers in its dialogs can be either of the following:

Table 7. Projects in Business Object Designer Express.

Business Object Designer Express environment	Project
If you are not running Business Object Designer Express from System Manager.	A virtual work area into which you have imported business object definitions from a local directory to work with during your current Business Object Designer Express session. Also called a *local project*.
If you are running Business Object Designer Express from System Manager.	An Integration Component Library (ICL) on the Windows machine where Business Object Designer Express and System Manager are running. Also called an *ICL-based project*.

The use of each type of project is explained in more detail below.

If Business Object Designer Express is running without System Manager

If you are not running Business Object Designer Express from System Manager, Business Object Designer Express uses a local project as "the project." A local project is a virtual work area into which you can import business object definitions you want to work with.

How Business Object Designer Express works with a local project

Listed below is a high-level summary of how Business Object Designer Express functions operate on a local project. More detailed information about performing these tasks is provided in the topics starting with "Launching Business Object Designer Express" on page 42.

- **Editing existing Business object definitions:** To edit an existing business object definition, click File > Open From File from the menu bar. This menu item imports the business object definition from a local directory into your project and optionally opens it for editing.

  To edit an existing business object definition that has already been imported into your project but that is now closed, click File > Open.
- **Creating a new business object definition:** To create a new business object definition, click File > New or File > New Using ODA from the menu bar.
- **Saving a business object definition:** To save a new or modified business object definition, select Save from the menu bar. You are prompted to save it to a local directory. To save a business object definition under a different name or directory, select Save As.
- **Deleting business object definitions:** To delete a business object definition from the directory where it resides, use the tools provided by Windows. You cannot use the Delete function in Business Object Designer Express to do this. To delete a business object definition from a local project, select File > Delete. You are prompted to select a business object definition to delete from your project.

If Business Object Designer Express is running from System Manager

When you run Business Object Designer Express from System Manager, you have access to additional, more sophisticated, functionality for developing and managing business object definitions. In System Manager, business object definitions, along with other business integration components such as collaborations and maps, are stored in *Integration Component Libraries* (ICLs). ICLs are repositories of business integration components, which you can use as building blocks to construct business integration solutions. Each ICL contains a collection of folders, one for each type of integration component, as shown in Figure 21.

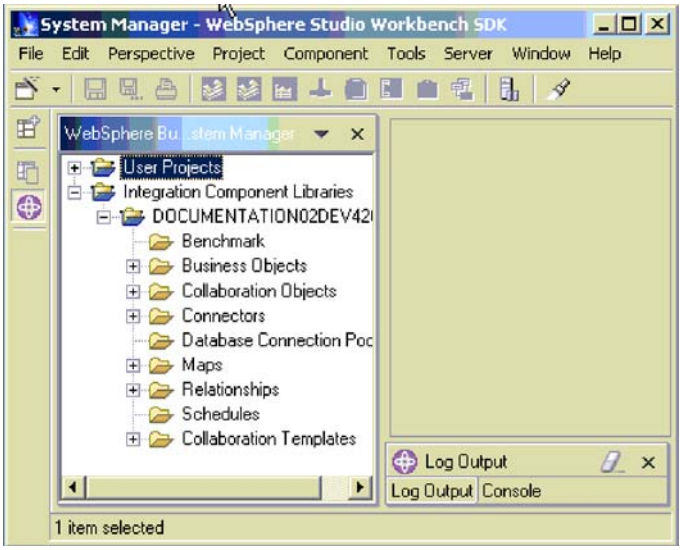


Figure 21. Integration Component Libraries in System Manager.

The methodology for developing and deploying business object definitions is as follows. You develop a business object definition in Business Object Designer Express and save it to the business objects folder in an ICL. When you want to use that business object definition in a business integration solution, you associate the definition with one or more *user projects*. Each user project includes all the business

integration components needed to implement a particular business integration solution. For example, in a user project that contains the components needed for the implementation of the e-Mail adapter, the business objects folder contains all the business object definitions needed by that adapter.

Like an ICL, each user project contains a collection of business integration component folders. However, a user project contains only virtual copies of ICL components. When you change a business object definition, you modify the instance in the ICL. The changes you make are automatically propagated to every user project that includes the business object definition. In other words, if a particular business object definition is included in two user projects, and a change is made to that definition in the Integration Component Library, the change is automatically reflected in the virtual copies residing in the user projects. This linkage between business object definitions in an ICL and their virtual copies in the user projects allows you to modify and maintain business object definitions in one central location while deploying them in multiple business integration solutions.

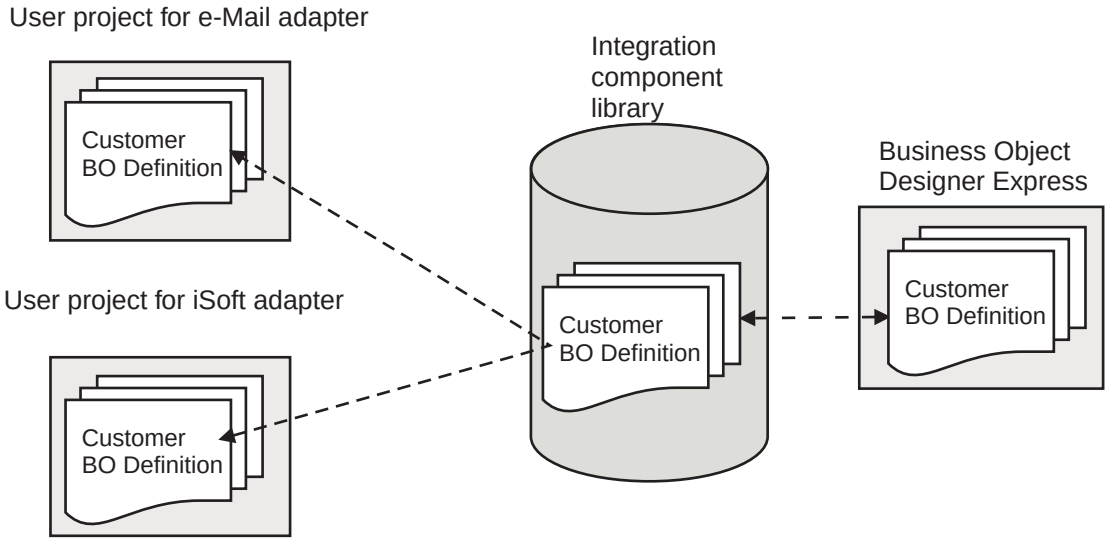


Figure 22. Changes to business object definitions in an ICL propagate automatically to virtual copies in user projects.

For more information about developing business integration components using Integration Component Libraries, see the implementation guide for your system.

How Business Object Designer Express works with an ICL-based project

When you are running Business Object Designer Express from System Manager, it uses as "the project" the Integration Component Library you have selected. Listed below is a high-level summary of how Business Object Designer Express functions operate on an ICL-based project. More detailed information about performing these tasks is provided in the topics starting with "Launching Business Object Designer Express" on page 42.

- **Editing existing Business object definitions:** To edit a business object definition stored in a project, click the File > Open menu item from the menu bar.
- **Creating new business object definitions:** To create a new business object definition, click File > New or click File > New Using ODA from the menu bar.
- **Saving business object definitions:** To save a new or modified business object definition, click File > Save from the menu bar. The business object is saved to

the business objects folder in the project. To save a new or modified business object definition using a different name, click File > Save As.

- **Deleting business object definitions:** To delete a business object definition, select the business object definition and then select Delete from the menu bar.

When you run Business Object Designer Express without System Manager, you do not have access to the Integration Component Libraries. In this environment, Business Object Designer Express uses a local project as described in "If Business Object Designer Express is running without System Manager" on page 39.

Launching Business Object Designer Express

You can open Business Object Designer Express in any of the ways listed in Table 8. After opening Business Object Designer Express, you can create a business object definition manually or use an Object Discovery Agent to generate a definition for an application-specific business object. For more information, see Chapter 4, "Developing business objects," on page 53.

Table 8. Ways to launch Business Object Designer Express.

From System Manager	• Select the business objects folder in an Integration Component Library, then do either of the following: – Select Business Object Designer Express from the Tools menu. – Click the Business Object Designer Express tool bar icon. • Right-click on the business objects folder in an Integration Component Library. • Double-click on a business object definition.
Using a Windows shortcut	Select Programs > IBM WebSphere InterChange Server Express > IBM WebSphere Business Integration Toolset Express > Development > Business Object Designer Express.
From another development tool	Do either of the following: • From the Tools menu, select Business Object Designer Express. • From the tool bar, double-click the Business Object Designer Express icon.

When you open Business Object Designer Express directly from System Manager, without first selecting a business object definition, the New Business Object dialog opens automatically. If System Manager is not running, Business Object Designer Express opens but the New Business Object dialog does not display.

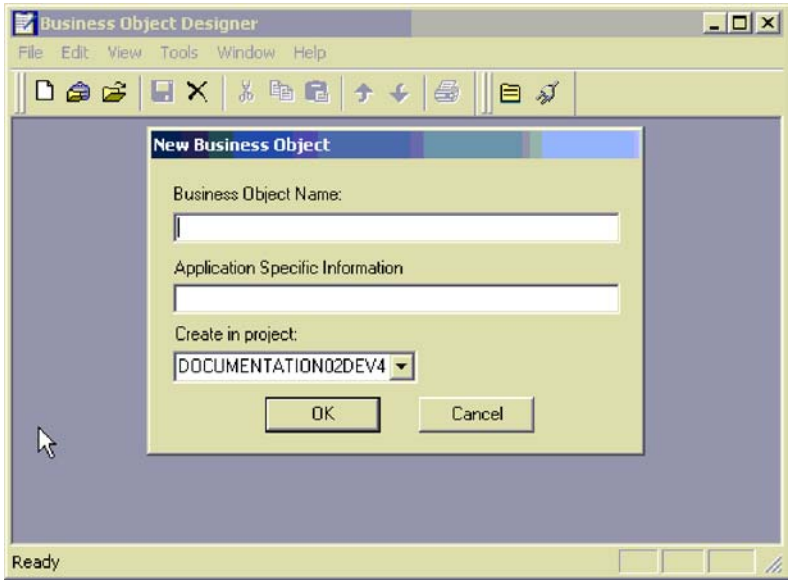


Figure 23. Business Object Designer Express displaying New Business Object dialog

When you open Business Object Designer Express by double-clicking a business object definition, the selected definition is displayed in Business Object Designer Express's work area.

Opening an existing business object definition from Business Object Designer Express

Once you open Business Object Designer Express, you can open object definitions stored in a file. If Business Object Designer Express is running from System Manager, you can also open business object definitions stored in an Integration Component Library.

This section describes:

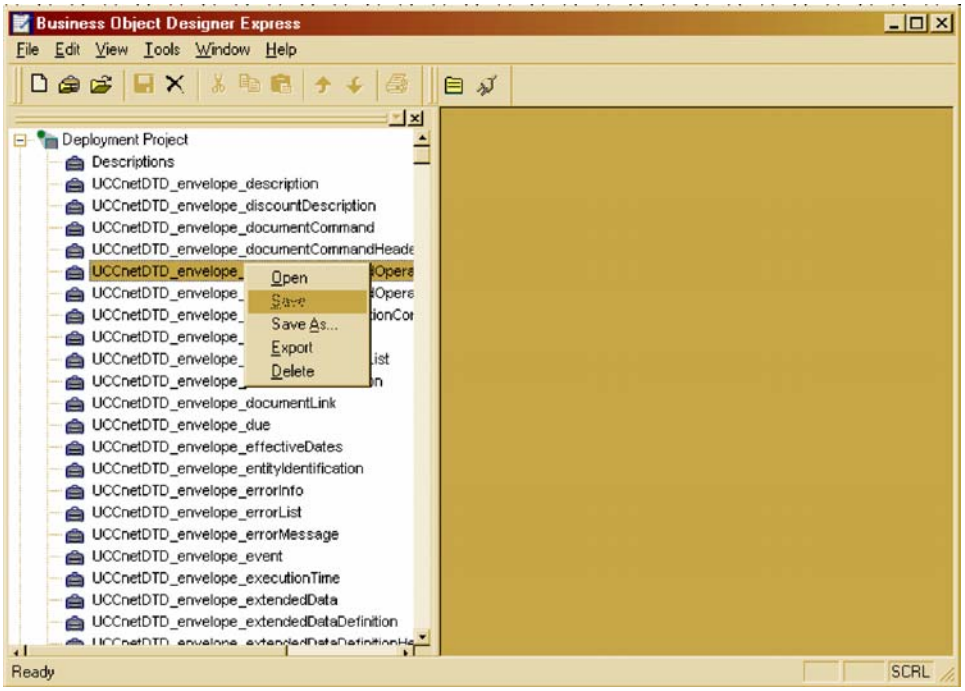
- "Opening a business object definition from a project"
- "Preventing duplicate definition names" on page 45
- "Opening a definition from a file" on page 44

Opening a business object definition from a project

If Business Object Designer Express is already open, you can do the following to open a business object definition from a project.

Note: If Business Object Designer Express is running from System Manager, the project is an ICL-based project. Otherwise, the project is a local project, which contains only business object definitions you have imported into it. See "Working with projects" on page 39 for more information about projects in Business Object Designer Express.

1. From the list of business object definitions in the project, highlight the name of the definition you want to open.



2. To select multiple business object definitions in the project, do one of the following:
   - When selecting consecutive names, select the first name and, while pressing the `Shift` key, click the last name.
   - When selecting non-consecutive names, press the `Ctrl` key and click as you select each name.
3. After selecting the definitions to be opened, right-click and then click Open.

   Business Object Designer Express displays a window for each selected definition.

Opening a definition from a file

To open a business object definition that is stored in a local directory, do the following:

1. From the File menu, select Open From File.

   The Import dialog displays. The dialog defaults to filter files of type XML Schema Definition (with a `.xsd` extension). You can also select a different file type from the Files of type drop-down menu or you can select all file types.
2. In the Import dialog, browse until you locate the file, select it, and click Open. Figure 24 illustrates this dialog.

   Note: If Business Object is not running from System Manager, the To Project field, which lets you specify the ICL-based project to receive the imported business object definition is omitted from the dialog. Instead, the business object definition is imported into your local project.

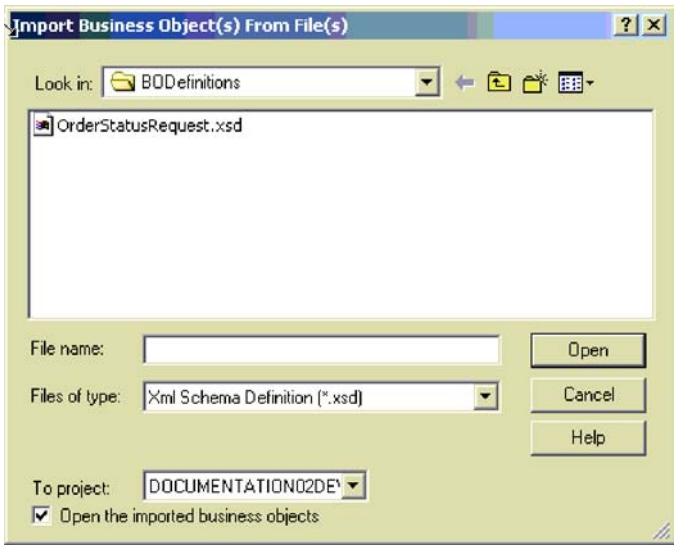


Figure 24. Import Business Objects dialog.

If the Open the imported business objects checkbox is checked then Business Object Designer Express also opens the business object definition for editing. Otherwise, the business object definition is imported into the project but not opened for editing. For more information, see "Working with business object definitions" on page 46.

Preventing duplicate definition names

Business Object Designer Express does not allow you to have two business objects with the same name in the same project, which might occur in either of the following situations:

- You attempt to open a definition from a file that is identical to one you already have in your project.

  For more information, see "Preventing duplicate names when a definition with the same name already exists in your project" on page 45.
- You attempt to create a new definition that is identical to one that already exists in your project.

  In this case, Business Object Designer Express displays an error message with the text: `Business object with this name already exists`.

Preventing duplicate names when a definition with the same name already exists in your project

If you attempt to open a definition from a file and your local or ICL-based project already contains a definition with the same name, Business Object Designer Express displays the Import Results dialog illustrated in Figure 25.

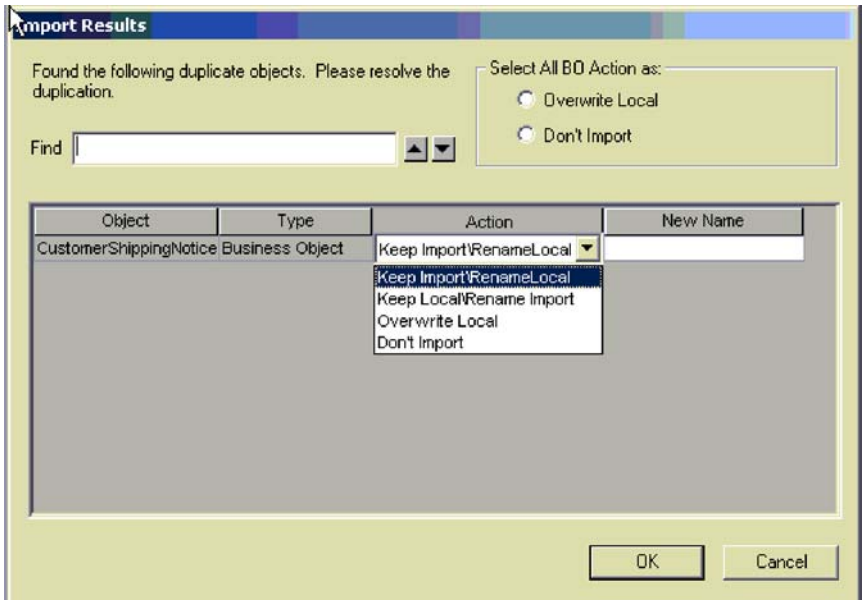


Figure 25. Preventing duplicate names: Keep Local or Import.

In the Import Results dialog, do the following for each business object definition listed as having a duplicate name:

1. Click the Action entry for the definition and select an action from the drop-down list. An explanation of each action is provided below.
2. If you selected Keep Import/Rename Local or Keep Local/Rename Import, enter the new name for the business object definition in the Name column as shown in Figure 25.

Alternatively, you can use "Select All BO Action as:" to specify either of two actions for every business object definition listed as a duplicate name. To overwrite all the business object definitions in your project with the definitions you are importing, select Overwrite Local. To refrain from importing the business object definitions that have duplicate names, select Don't Import.

The Actions drop-down list in the Import Results dialog provides the following options:

- Keep Import\RenameLocal—Allows you to change the name of the definition in your project and leave unchanged the name of the definition in the file.

  To make this change, enter the new name in the New Name column as shown in Figure 25.
- Keep Local\Rename Import—Allows you to change the name of the definition in the file and leave unchanged the name of the definition in your project.

  To make this change, enter the new name in the New Name column as shown in Figure 25.
- Overwrite Local—Overwrites the definition currently stored your project with the definition stored in the file.
- Don't Import—Cancels the action to import the definition stored in the file.

Working with business object definitions

Business Object Designer Express provides a tabbed dialog with two screens for creating and editing a definition:

- General screen — specify or change application-specific information and verbs at the business object-level.
- Attributes screen — specify or change attribute properties.

When you first create or open a definition, the Attributes screen displays.

Figure 26 illustrates the screen for defining and editing attributes.

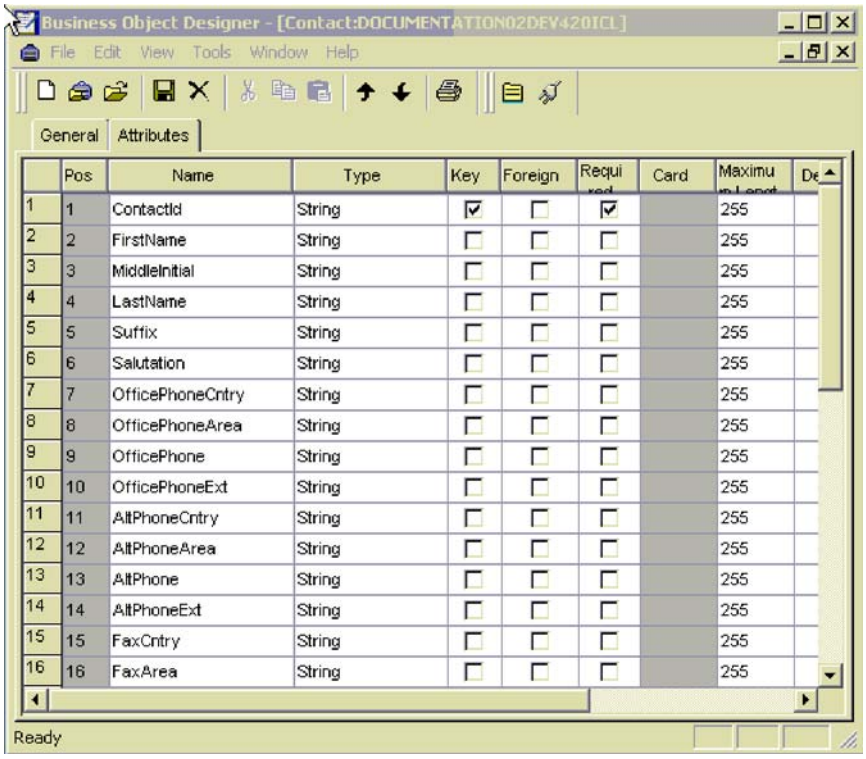


Figure 26. Defining and editing attributes.

For information on using the General and Attributes screens, see "Creating a business object definition" on page 53.

Opening a business object definition and its contained child

Business Object Designer Express allows you to open separate windows to edit definitions for a parent business object definition and its contained child.

Figure 27 illustrates the separate windows for editing parent and child business objects.

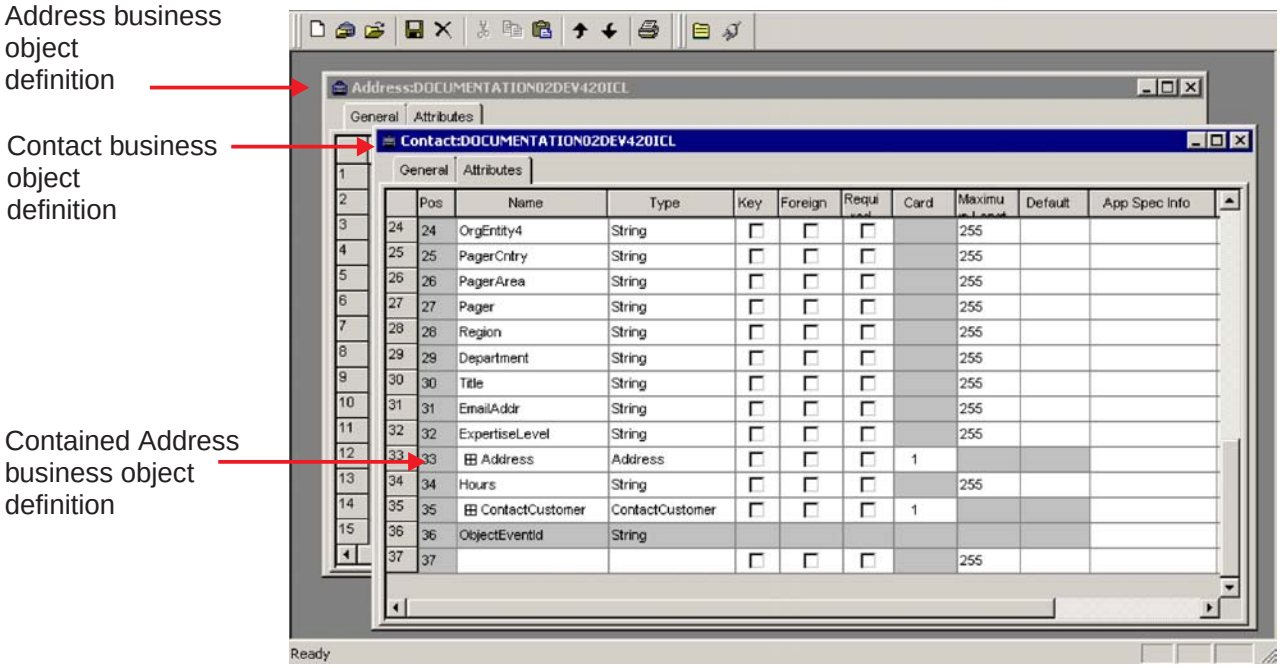


Figure 27. Separate windows for parent and child business objects.

Notice that the Address attribute in the Contact business object is collapsed in Figure 27. You can expand the attribute so that the Contact window displays all attributes of the Address business object, which enables you to edit the child directly from the parent. To prevent you from changing the same definition in two places, however, the tool automatically closes a child's window whenever you expand the child within its parent.

Figure 28 illustrates the tool after Contact's Address attribute has been expanded and the Address window has been closed.

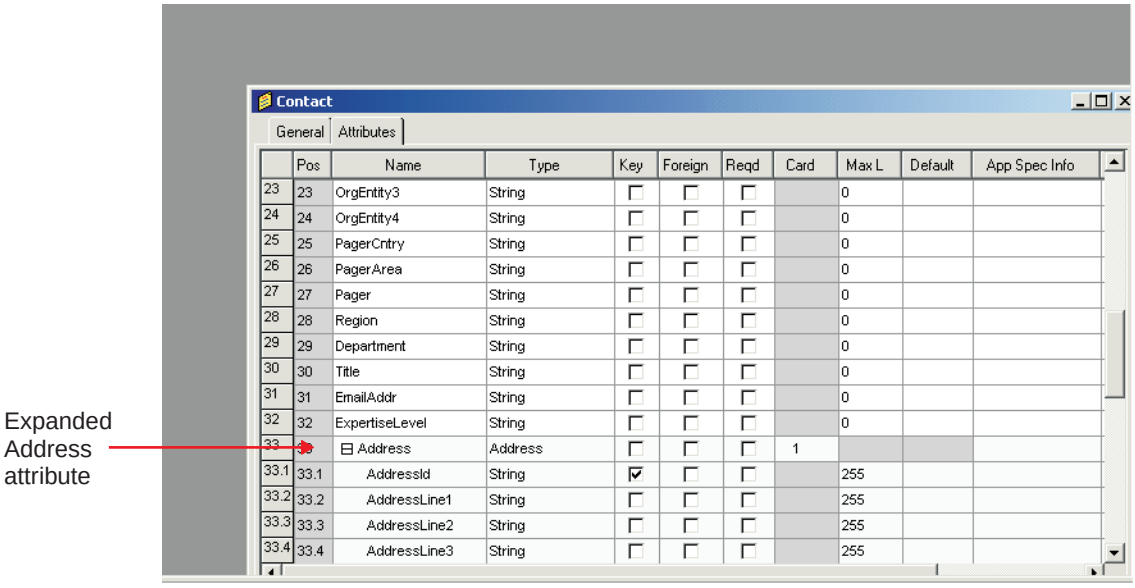


Figure 28. Expanding the parent's attribute that represents the child.

Business Object Designer Express functionality

You can access Business Object Designer Express functions in either of the following ways:

- From the menu bar
- Using toolbar icons.

The following sections provide an overview of Business Object Designer Express menus and menu options.

File menu

The File menu displays the following options:

- New BO — Creates a business object definition manually. For more information, see "Creating a business object definition" on page 53.
- New Using ODA — Presents the Business Object Wizard, which allows you to create a business object definition from an Object Discovery Agent. For more information, see "Using an Object Discovery Agent to create a business object definition" on page 62.
- Open — Opens a business object definition located in the project. If Business Object Designer Express is running from System Manager, the project is an ICL-based project. Otherwise, the project is a local project. See "Working with projects" on page 39 for more information about projects.
- Open From File — Imports and optionally opens a business object definition from a local directory.
- Save — Saves the business object definition as follows to the project.

  If you modified an existing business object definition:

  – If the project is ICL-based, the business object definition is saved in the project where it originated.
  – If the project is local, the business object definition is saved to its existing file.

  If you created a new business object definition:

  – If Business Object Designer Express is running from System Manager, the business object definition is saved to its ICL-based project.
  – Otherwise, you are prompted to specify the local destination directory and file name for the business object definition. If the business object definition is successfully saved, any subsequent Save actions automatically save the business object to the local destination directory and file you specified the first time you saved the file.

  The business object definition can be saved as a file of type:

  .xsd XML Schema Definition. This is the default file type.

  .in or .txt InterChange Server Express

  .xls Spreadsheet
- Save As — Saves the business object definition with a new name. The name of the file containing the business object definition must be unique within the destination project.

  If you modified an existing business object definition:

  – If Business Object Designer Express is running from System Manager, the business object definition is saved to the ICL-based project.

– If the business object definition was opened from a file, the modified business object definition is saved to that file.

If you created a new business object definition:

– If Business Object Designer Express is running from System Manager, you are prompted to select the ICL where the business object definition is to be saved.
– Otherwise, you are prompted to specify a local destination directory and file name for the business object definition.

The business object definition can be saved as a file of type:

.xsd	XML Schema Definition. This is the default file type.
.in or .txt	InterChange Server Express
.xls	Spreadsheet

- Save All—Saves all open business object definitions as described under the Save menu item on page 49.
- Save Copy to File—Exports a copy of the business object definition to a separate file.
- Copy All to One File—Exports all business object definitions in a project as a single file in repos-copy format.
- Close—Closes the active definition. This menu item is disabled if no definitions are open.
- Close All—Closes all open definitions. This menu item is disabled if no definitions are open.
- Delete — Allows you to delete a business object definition from a project.

  Note: Business Object Designer Express only lets you delete business object definitions from the project. If your project is ICL-based, the business object definitions you delete are removed from the specified ICL. If your project is local, the business object definitions you delete are removed from the local project but the files that contain business object definitions in local directories are not affected. To delete local files, use the tools provided by Windows.
- Print Setup — Allows you to specify the printer and printing properties.
- Print Preview — Displays a preview of the definition to be printed. This menu item is disabled if no definitions are open.
- Print — Allows you to print the selected definition. This menu item is disabled if no definitions are open.
- Exit — Allows you to exit Business Object Designer Express.

Edit menu

All options of the Edit menu are disabled if no definitions are open. The Edit menu displays the following options:

- Cut — Deletes an attribute from the definition or text from a column. This menu item is disabled if no text has been selected in a column or no attribute has been selected (by clicking in the left-most column). See Figure 26 on page 47 for an illustration of the window for editing attributes.
- Copy — Copies an attribute in the definition or text in a column. This menu item is disabled if no text has been selected in a column or no attribute has been selected (by clicking in the left-most column). See Figure 26 on page 47 for an illustration of the window for editing attributes.

- Paste — Pastes a cut or copied attribute into the definition, or cut or copied text into the selected column. By default, the tool pastes a buffered attribute at the bottom of the definition. However, if you insert an empty row at a specific location, you can paste the buffered attribute into the empty row.
- Delete Row — Deletes an attribute from the definition. This menu item is disabled if no attribute has been selected (by clicking in the left-most column). See Figure 26 on page 47 for an illustration of the screen for editing attributes.
- Select All — Selects all attributes in the definition.
- Insert Above — Inserts an empty row above the selected attribute.
- Insert Below — Inserts an empty row below the selected attribute.
- Move Up — Moves the selected attribute up one row. This menu item is disabled if no attribute has been selected.
- Move Down — Moves the selected attribute down one row. This menu item is disabled if no attribute has been selected.

Note: You can access the Insert Above, Insert Below, Cut, Copy, Paste, and Delete options from a context menu opened by right-clicking in the left-most column of an attribute.

View menu

The View menu operations are valid when Business Object Designer Express first opens and when the Working Area display pertains to the visual appearance of activity diagrams. Many of these operations can be toggled on or off.

The View menu displays the following options:

- Expand All — Displays all attributes in all child business objects. This menu item is disabled if no definitions are open.
- Collapse All — Closes display of all attributes in all child business objects. This menu item is disabled if no definitions are open.
- Preferences — Displays the Business Object Preferences dialog, which allows you to turn off confirmation of business object deletion.
- Toolbars — Contains a submenu whose options control display of the two toolbars of the Business Object Designer Express. Menu options include:
  - Standard — When you click this menu item, Business Object Designer Express displays the buttons for the Standard toolbar.
  - Programs — When you click this menu item, Business Object Designer Express displays the buttons for accessing other WebSphere Business Integration Toolset Express programs.
- Status Bar — When you click this menu item, Business Object Designer Express displays a single-line status message at the bottom of its main window.

Note: You can access the Expand, Collapse, and Open In Window options from a context menu opened by right-clicking in the left-most column of an attribute that represents a child business object or an array of child business objects.

Tools menu

The Tools menu contains the following options:

- Log Viewer — Launches Log Viewer.
- Connector Configurator Express — Launches Connector Configurator Express.

Window menu

The Window menu operates as it does in a standard environment. Use the menu options to control display features such as tiling, cascading, and activating open windows.

Chapter 4. Developing business objects

This chapter walks you through the basic steps for creating and deleting a business object definition. After you complete this chapter, you will be familiar with the steps for creating a definition both manually and by using an Object Discovery Agent (ODA). Each ODA generates definitions for a specific application.

Although this chapter presents the mechanics for creating business object definitions, you should understand the design concepts before you actually create one. For more information, see Chapter 2, "Designing business objects," on page 13.

The main topics of this chapter are:

- "Creating a business object definition"
- "Deleting a business object definition" on page 60
- "Using an Object Discovery Agent to create a business object definition" on page 62

Creating a business object definition

There are two ways to create a business object definition:

- Manually—Useful when creating a generic business object or a simple business object, or when modifying a definition generated by an Object Discovery Agent. Business Object Designer Express provides a graphic interface for the manual creation of a business object definition. This section provides a tutorial that explains:
  - "Creating a flat business object definition manually"
  - "Creating a hierarchical business object definition manually" on page 59
- Using an Object Discovery Agent—Useful when creating an application-specific business object. The *Object Discovery Agent* examines specified entities in the application, "discovers" the elements of those objects that correspond to business object attributes and the properties of each attribute, and generates the business object definition. For more information, see "Using an Object Discovery Agent to create a business object definition" on page 62.

Creating a flat business object definition manually

This section describes the manual creation of a business object definition named `Hello`. This business object is used by the `SampleHello` collaboration.

Figure 29 illustrates the `Hello` business object definition that you will create and shows the values that its integration broker might expect from its triggering-event business object.

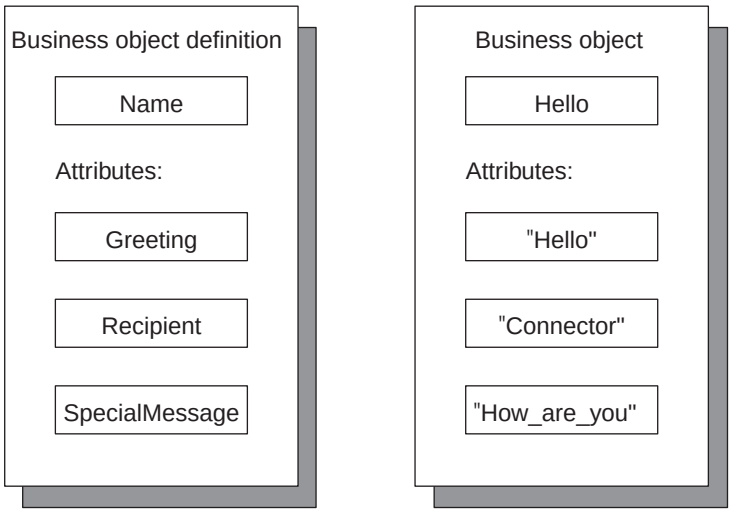


Figure 29. Hello business object.

To create a business object definition manually:

1. Launch Business Object Designer Express; for more information, see "Launching Business Object Designer Express" on page 42.
2. From the File menu, select New.

   Business Object Designer Express displays the New Business Object dialog. Figure 30 shows the version of the New Business Object Dialog you see if you are running Business Object Designer Express from System Manager. If you are not running Business Object Designer Express from System Manager, the Create in Project: field is omitted from the dialog.

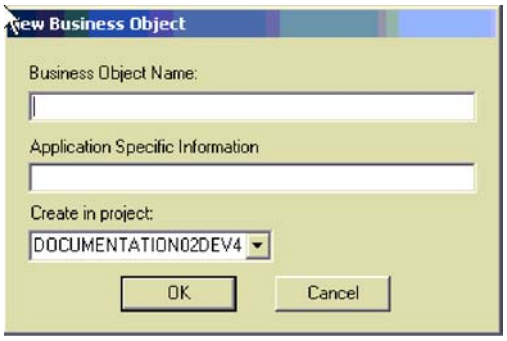


Figure 30. New Business Object dialog.

3. Enter the name `Hello` for the new business object definition.

   Names are generally case-sensitive, so enter the name exactly as shown here.

   Note: The name of a business object definition can contain only alphanumeric characters and underscore (_). This name must use *only* characters defined in the code set associated with the U.S. English locale (`en_US`).
4. Leave the Application Specific Information field empty and click OK.

Business Object Designer Express displays the business object definition dialog, as illustrated in Figure 31.

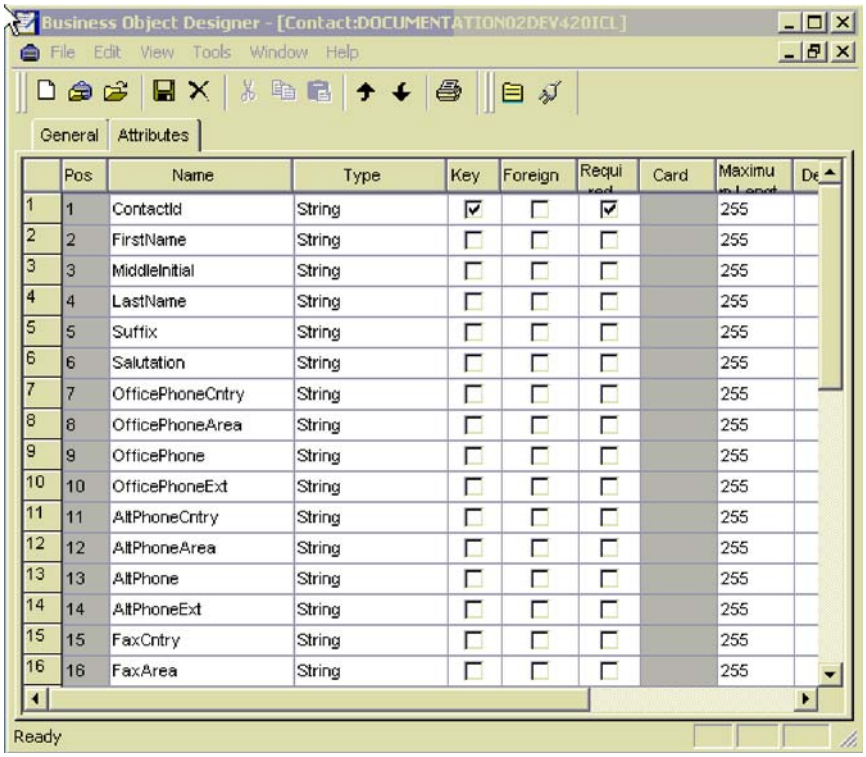


Figure 31. Initial display of a new business object definition.

Adding attributes

Each piece of information in the business object is represented by an *attribute* in the `Hello` business object definition. You must provide the attribute definitions for the `Hello` business object. As illustrated in Figure 31, the tool automatically adds an entry for the required "end of object" marker, `ObjectEventId`.

Important

Do not delete, change, or move the `ObjectEventId` attribute. This attribute is reserved for the WebSphere business integration system's internal use. Business Object Designer automatically moves this attribute when you save the definition.

The row for each attribute defines the attribute's properties. For information on the attribute properties, see "Business object attributes and attribute properties" on page 2.

As Figure 29 on page 54 shows, the `Hello` business object definition has the following attributes: `Greeting`, `Recipient`, and `SpecialMessage`. Define the attributes and their properties, one at a time.

Adding the Greeting attribute: To add the `Greeting` attribute:

1. Enter the attribute name `Greeting` in the Name column of the first available empty row, which is 2 for the first attribute.

   Note: This attribute name must use *only* characters defined in the code set associated with the U.S. English locale (`en_US`).

2. Click in the Type column to display the pull-down list of options and select `String` for the attribute type. The type of an attribute is its data type.

   Tip:

   If you have other business objects opened in Business Object Designer Express, their names appear in the Type pull-down menu. Displaying existing business objects among the choices for Type allows you to create a hierarchical business object with an attribute whose type is another business object.

   If System Manager is running, then every business object definition in the Integration Component Library you are working from is automatically displayed in this list.
3. Skip the Key, Foreign, Reqd (or Required), and Card fields.

   These fields specify whether the current attribute is the business object's primary or foreign key, whether the attribute's value is required, and whether the attribute represents a child business object or objects. For an explanation of these properties, see Chapter 2, "Designing business objects," on page 13.
4. In the Max Length field, retain the default value of `255`.

   This field specifies the maximum number of bytes available for this attribute's value.
5. In the Default value field, enter `Hello`.

   This field specifies the value to use if no other value is supplied for the attribute at runtime.

   You have now defined the following properties for the `Greeting` attribute:

Name:	`Greeting`
Type:	`String`
Maximum length:	`255`
Default value:	`Hello`

6. Ignore all other fields and click in the Name column of the third row.

Adding the Recipient attribute: The second attribute, `Recipient`, is a string.

The `SampleHello` collaboration object uses this attribute as follows:

- The connector sets the value to `Collaboration` when it sends a message to the collaboration.
- The collaboration sets the value to `Connector` when it sends a message to the connector.

At least one attribute in each business object definition must be a *key attribute*. A key attribute contains a value by which the WebSphere business integration system uniquely identifies instances of the business object. Make the `Recipient` attribute the key attribute.

To add the `Recipient` attribute, enter the text `Recipient` in the Name column, and follow the steps for adding the `Greeting` attribute, using the following properties:

Name:	`Recipient`
Type:	`String`

Maximum length:	255
Default value:	Collaboration
Key:	Yes (A checkmark appears in the Key column)

Leave the other fields blank and click in the Name column of the fourth row.

Adding the SpecialMessage attribute: The third attribute, SpecialMessage, is a string.

The SampleHello collaboration expects the value of this attribute to be entered by the system administrator or another person with access to the collaboration configuration properties after the collaboration object has been created. The collaboration dynamically obtains the value of the configuration property and appends it to the message.

To add the SpecialMessage attribute, enter the text SpecialMessage in the Name column, and follow the steps for adding the Greeting attribute, using the following properties

Name:	SpecialMessage
Type:	String
Maximum length:	255

Leave the other fields blank.

The Attributes tab now displays three user-defined attributes: Greeting, Recipient, and SpecialMessage. Figure 32 illustrates the Hello business object's attributes.

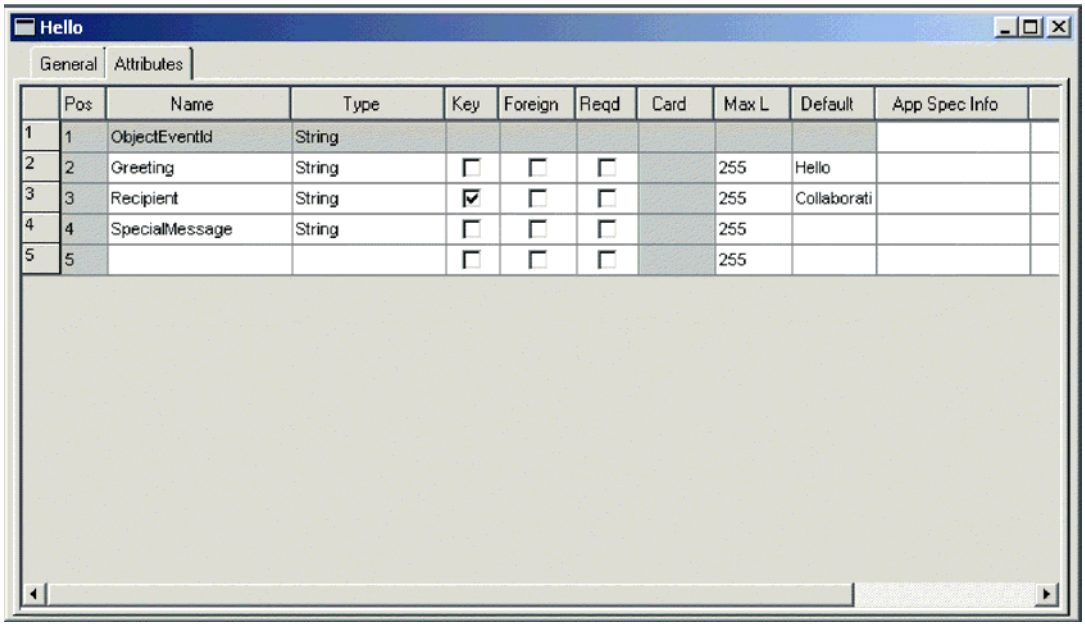


Figure 32. New business object definition with attributes.

Changing attribute order

You can graphically change the sequence order of attributes in the business object definition. For example, to place the key attribute, Recipient, above the Greeting attribute, click in the first (leftmost) column and drag the cursor up one row.

Specifying the supported verbs

You must now specify the verbs that this `Hello` business object will support. These verbs represent the triggering events that the business object sends to the integration broker. Click the General tab of the Hello business object definition dialog to display the screen in which you specify the verbs. Figure 33 illustrates this dialog.

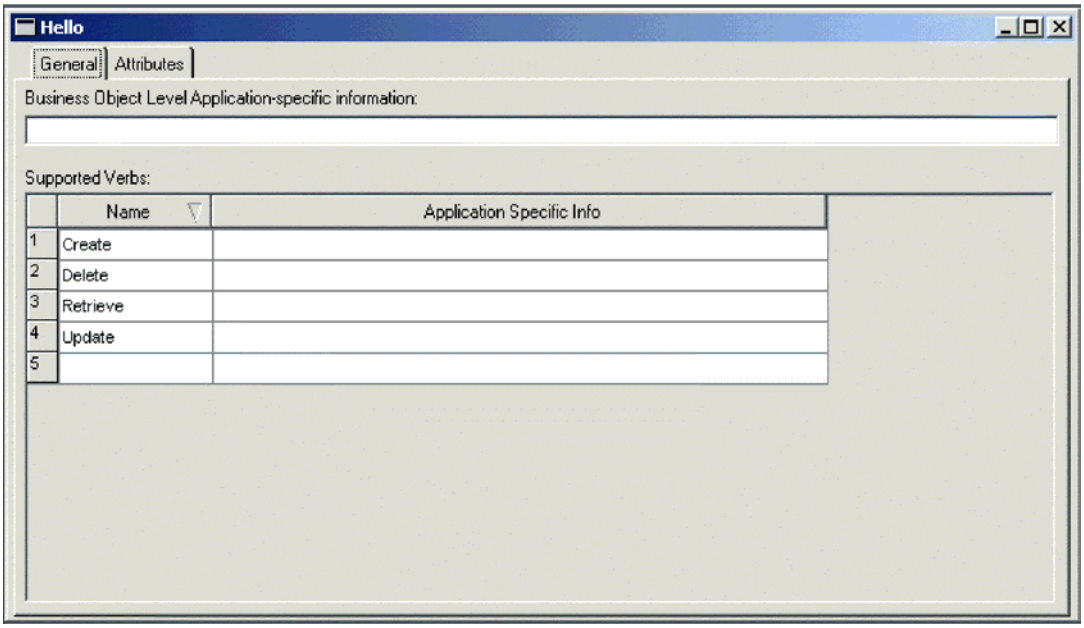


Figure 33. General editing screen

The business object supports the four default verbs—Create, Delete, Retrieve, and Update—and they appear by default. In this example, only one triggering event is supported: Create. Therefore, change the business object definition to support only this verb.

Important: You must specify at least one verb for each business object definition.

Note: The name of a verb can contain only alphanumeric characters and underscore (_). This name must use *only* characters defined in the code set associated with the U.S. English locale (`en_US`).

To indicate that the `Hello` business object supports only the Create verb, you can either delete the remaining verbs simultaneously or individually.

Deleting multiple verbs simultaneously: To delete the Delete, Retrieve, and Update verbs simultaneously:

1. Select the Delete verb and, while pressing the `Shift` key, click the Update verb.
2. Click the Delete key.

Deleting each verb individually: To delete each verb individually:

1. Click the number to the left of the Delete line in the Supported Verbs table.

The row becomes highlighted.

2. Click the Delete key.
3. Repeat steps 1 and 2 for the Retrieve and Update verbs of the Supported Verbs table.
4. Leave blank the field of Application Specific Info for the Create verb.

You have finished the definition for the `Hello` business object. This is a good time to save your changes by selecting File > Save from the menu bar. If you are using an ICL-based project, the definition is saved to the ICL. If you are using a local project, you will be prompted to specify a file name and local directory in which to save the definition.

Creating a hierarchical business object definition manually

This section describes how to create a hierarchical business object definition by defining an attribute that represents a child business object or an array of child business objects.

Because the previous section explains how to define a simple attribute and supported verbs, this section explains only the definition of an attribute that represents a child business object. This example creates a business object named `HierarchicalBO` that has two attributes:

- An attribute named `Key` that serves as the required business object key.
- An attribute named `Addr` that represents the Address business object with cardinality 1.

To manually create a hierarchical business object definition:

1. Open Business Object Designer Express.
2. From the File menu, select New.

   Business Object Designer Express displays the New Business Object dialog, as illustrated in Figure 30 on page 54.
3. Enter the name `HierarchicalBO` for the new business object definition.
4. Leave the Application Specific Information field empty and click OK.

   Business Object Designer Express displays the business object definition dialog, as illustrated in Figure 31 on page 55.
5. Create a key attribute in the first available empty row, which is `2` for the first attribute. Name it `Key`, specify any simple data type, and select the Key column.
6. Create the next attribute in the next available empty row, which is `3`. Name it `Addr`.
7. Click in the Type column to display the pull-down list of options and select `Address` for the attribute type.

   Note: If the child business object does not exist in the list, you can create it now by selecting New business object from the Type pull-down menu. You must save the new child before you can complete this step.
8. Skip the Key, Foreign, and Reqd (or Required) fields. Click in the Card field to display the drop-down list of options. Select 1.
9. Ignore all other fields. Define supported verbs and save the definition.

Deleting a business object definition

You can delete a business object definition using Business Object Designer Express or System Manager. This section describes:

- "Deleting a definition using Business Object Designer Express"
- "Deleting a definition using System Manager" on page 61

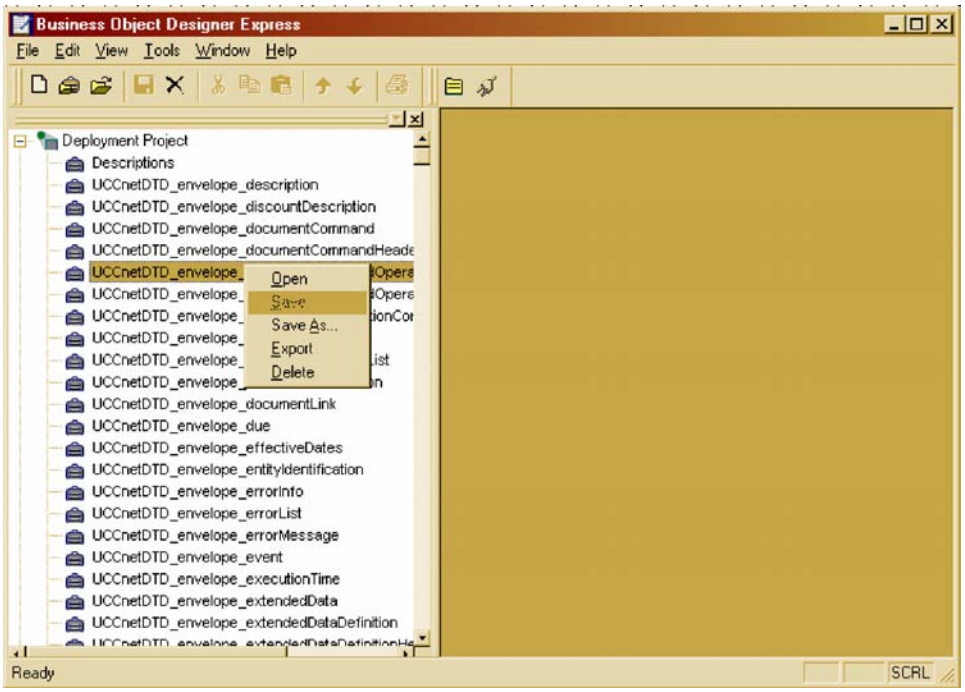
Important

You can delete business object definitions from an integration component library through System Manager (if you are using InterChange Server Express and System Manager is running) or from a project in Business Object Designer Express. You cannot use the Delete function in Business Object Designer Express or in System Manager to delete local files that contain business object definitions. To delete local files, use the tools provided by Windows.

Deleting a definition using Business Object Designer Express

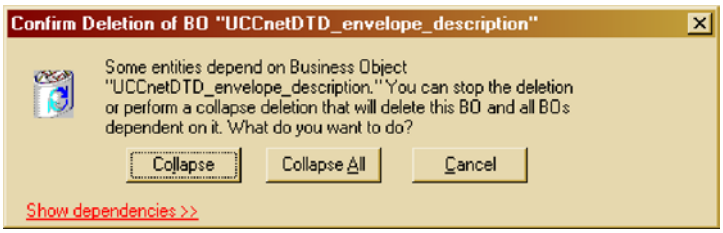
To delete a business object definition from a project using Business Object Designer Express, do the following:

1. Open Business Object Designer Express.
2. From the list of business object definitions in the project, highlight the name of the definition you want to delete.

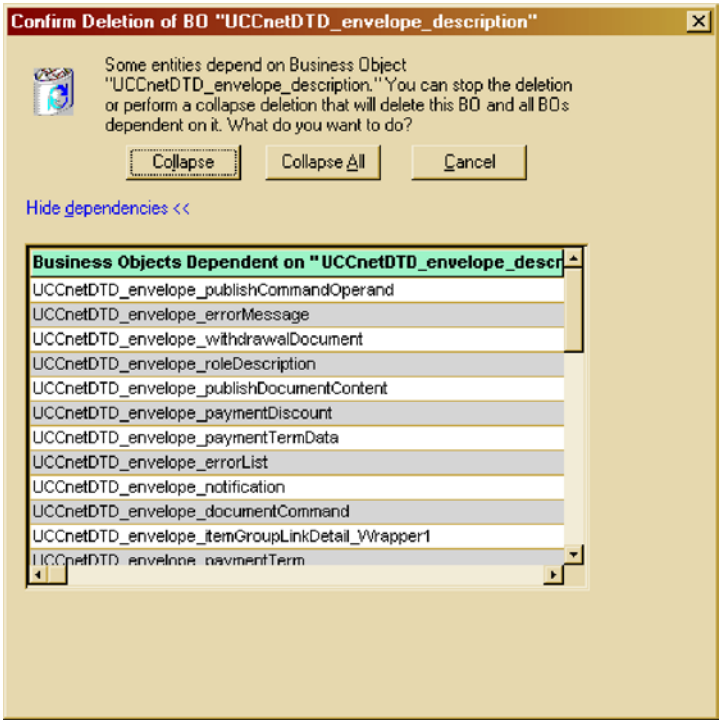


3. To select multiple names, do one of the following:
   - When selecting consecutive names, select the first name and, while pressing the `Shift` key, click the last name.
   - When selecting non-consecutive names, press the `Ctrl` key and click as you select each name.
4. After selecting the definitions to be deleted, right-click and then click Delete.

- Business Object Designer Express displays the ″Deleting business object″ confirmation dialog. Click Yes to delete the business object definition you selected in Step 2 on page 60. Click Yes to all to delete all the business object definitions you selected in Step 3 on page 60.
- If the business object has dependencies, Business Object Designer Express displays a collapse delete confirmation dialog.



5. If dependencies exist, click the ″Show dependencies>>″ link. The list of dependencies for the business object you want to delete appears in the dialog.



6. Do one of the following:
   - Click Collapse to delete the business object definition you selected in Step 2 on page 60 and all of the business objects that depend on it.
   - If you selected multiple business objects in Step 3 on page 60, click Collapse All to delete the business objects you selected and all of the business objects that depend on each of those business objects.
   - Click Cancel to cancel deleting the business object definition and its dependencies.

Deleting a definition using System Manager

To delete a business object definition using System Manager, do the following:

1. Start System Manager.
2. Expand Integration Component Libraries and then expand the integration component library from which you want to delete a business object definition.

3. Open the business objects folder and select the name of the business object definition to delete.
4. Delete the business object definition by doing either of the following:
   - Click the Delete tool bar icon.
   - Right-click the business object definition and select Delete from the menu.
5. When prompted whether you want to delete, click Yes.
6. If the business object has dependencies, System Manager notifies you with an error message. You must use Business Object Designer Express to remove these dependencies before System Manager allows you to delete the business object.

Using an Object Discovery Agent to create a business object definition

This section describes how to use an Object Discovery Agent (ODA) to generate business object definitions for application-specific business objects. An ODA is an optional component of an adapter. When you install a pre-defined adapter that has an ODA, its ODA is installed automatically.

At runtime, execution of an ODA involves the following components:

- Business Object Designer Express provides a graphical interface in the form of a wizard to interact with the ODA. This wizard is called *Business Object Wizard*. The purpose of Business Object Wizard is to display a series of dialogs to the user. These dialogs obtain information from the user that the ODA needs to generate the content.
- The *ODA runtime* is the intermediary component between Business Object Wizard and the ODA. It is the ODA runtime that you launch with the ODA startup script.
- The *ODA* is the component that ″discovers″ source nodes in the data source and generates the content. The ODA receives information that the user provides in the dialogs of Business Object Wizard from the ODA runtime. It then sends information (such as the generated content) to the ODA runtime, which sends it to Business Object Wizard.

Figure 34 on page 63 shows the components of the ODA runtime architecture.

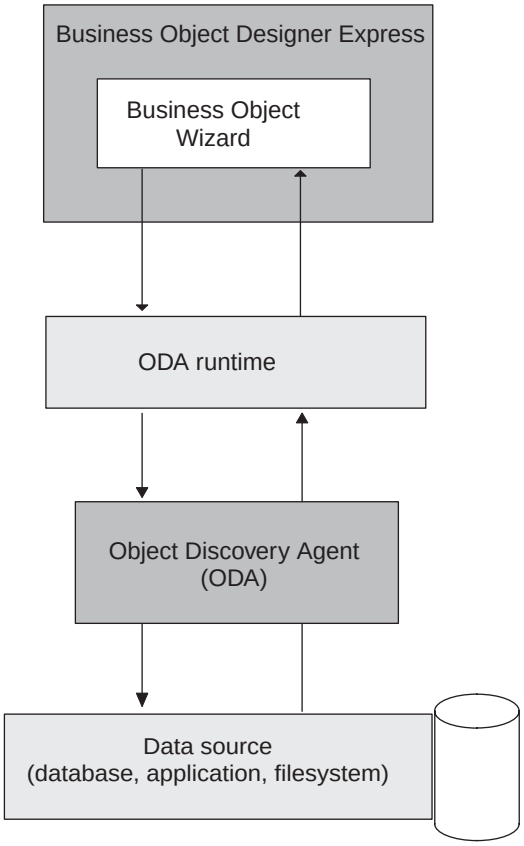


Figure 34. Object Discovery Agent Architecture

To configure and run the ODA, use *Business Object Wizard* within Business Object Designer Express. Business Object Wizard is a graphical user interface to ODAs, as well as managing the discovery and content-generation process. This section provides the following information:

- "Before using an ODA"
- "Using the ODA to create business object definitions" on page 65
- "Entering values and saving a profile" on page 73
- "Setting up logging and tracing" on page 74
- "Moving through the source-node hierarchy" on page 76
- "Providing additional information" on page 80
- "Using multiple ODAs simultaneously" on page 81

Before using an ODA

Before you run an ODA, verify that the following steps have occurred:

- System startup files are available and correct.
- The ODA has been launched.
- Business Object Designer Express has been launched.

System startup files

For the ODA to launch, you need to verify that your system has the required files for the ODA. When you install a pre-defined adapter that has an ODA, these ODA

system startup files should be installed automatically. However, IBM recommends that you confirm that the following files exist and are correct for your ODA:

- ODA startup script
- ODA environment file

ODA startup script: Each ODA requires a *startup script*, which begins execution of the ODA. Before you launch an ODA for the first time, you must make sure that the variables are correctly set within the startup script. Open for editing the batch file (`start_`*`ODAname`*`.bat`) and confirm that the values described in Table 9 are correct.

Table 9. ODA batch file configuration variables

Variable	Explanation	Example
`set AGENTNAME`	Name of the ODA	`set AGENTNAME=`*`ODAname`*
`set AGENT`	Name of the ODA's jar file	`set AGENT = %`*`ProductDir`*`%\ODA\`*`srcDataName`*`\`*`ODAname`*`.jar`
`set AGENTCLASS`	Name of the ODA's Java class	`set AGENTCLASS=com.ibm.oda.`*`srcDataName`*`.`*`ODAname`

ODA environment file: For an ODA's startup script to run, it needs to locate the ODA environment file. This file, `CWODAEnv.bat`, establishes general settings for various software and information that an ODA needs at runtime. It should be installed in the bin subdirectory of your product directory: *`ProductDir`*`\bin`

Launching the ODA

You can launch an ODA with the following startup script:

`start_`*`srcDataName`*`ODA.bat`

You configure and run the ODA using Business Object Wizard within Business Object Designer Express. Business Object Wizard locates each ODA by the name specified in the `AGENTNAME` variable of each script or batch file.

Note: For information on how to launch multiple instances of the ODA, see "Using multiple ODAs simultaneously" on page 81.

Launching Business Object Designer Express

Once you launch the ODA, you must launch Business Object Designer Express to configure and run it. For information on the ways to launch Business Object Designer Express, see "Launching Business Object Designer Express" on page 42. To run an ODA, Business Object Designer Express provides Business Object Wizard, which guides you through each step.

To start Business Object Wizard, do the following:

1. Open Business Object Designer Express using one of the methods listed in Table 8 on page 42.
2. From the File menu, click the ″New Using ODA″ menu item.

Business Object Designer Express invokes the Business Object Wizard to run the ODA. Step 1 of Business Object Wizard displays the first dialog in the wizard, Select Agent, which provides graphical access to all available Object Discovery Agents. From this dialog, the user selects the desired ODA to run. To display the names of all ODAs that are running on the current machine, click the Find Agents button.

Note: An ODA must already be launched for Business Object Wizard to list it as an ODA you can run. For more information, see "Before using an ODA" on page 63.

Table 10 summarizes the steps of Business Object Wizard.

Table 10. Steps of Business Object Wizard

Execution task		Step in Business Object Wizard
1.	Select the desired ODA.	Step 1: Select Agent
2.	Obtain the configuration properties, including those that describe the data source to open.	Step 2: Configure Agent
3.	Obtain the source data for which the ODA generates the content.	Step 3: Select Source
4.	Confirm that the selected source nodes are those desired for content generation.	Step 4: Confirm Source Nodes
5.	Initiate the content-generation process.	Step 5: Generating Business Objects Business Object Properties
6.	Save the business object definitions in a user-specified format.	Step 6: Save Business Objects

Using the ODA to create business object definitions

The steps outlined in this section include sample screens of an Object Discovery Agent that converts Roman-army soldiers (in XML format) to business object definitions.

1. Open Business Object Designer Express using a method listed in Table 8 on page 42.
2. From the File menu, click the ″New Using ODA″ menu item.

   Business Object Designer Express invokes Business Object Wizard, which displays the first dialog in the wizard, Select Agent, shown in Figure 35.

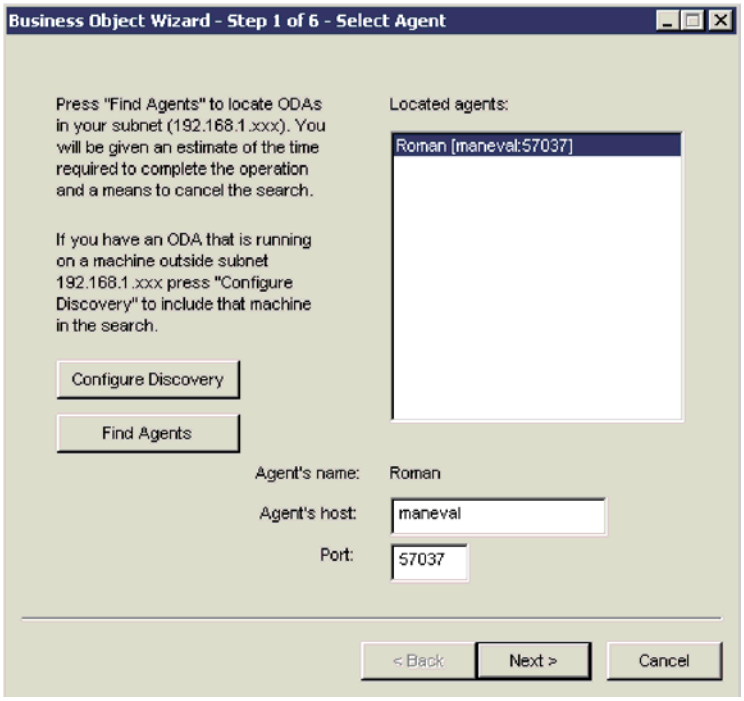


Figure 35. Select Agent dialog.

3. To select the ODA to which Business Object Wizard connects:
   a. Click the Find Agents button to display in the "Located agents" field those ODAs that are currently running (those that have been launched with their startup scripts).

      Note: If Business Object Wizard does *not* locate your desired ODA, check the startup of the ODA.

      Business Object Wizard identifies each running ODA by the name specified for the `AGENTNAME` variable of its startup script or batch file. This sample ODA is named `Roman`.
   b. Select the desired ODA from the "Located agents" list. Business Object Wizard displays your selection in the Agent's name field. Alternatively, you can find the ODA by specifying its host name and port number.
4. Click Next. Business Object Wizard attempts to connect to the specified ODA. If the ODA has been started, Business Object Wizard displays a status window as it connects to the ODA, as Figure 36 shows.

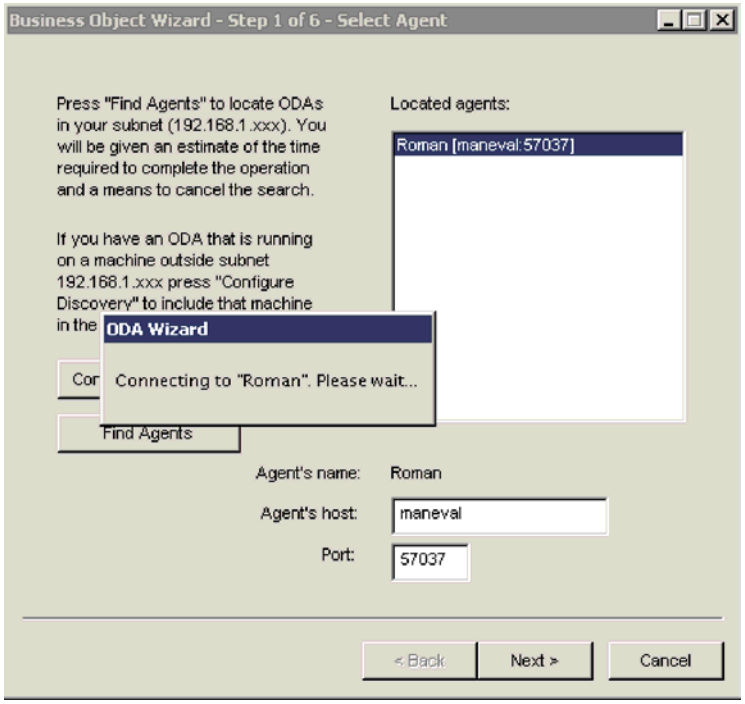


Figure 36. Connecting to an ODA.

5. Once Business Object Wizard is connected to the ODA, it displays the second wizard dialog, Configure Agent, which is shown in Figure 37. This dialog displays the ODA configuration properties required to access the data source and initialize the ODA.

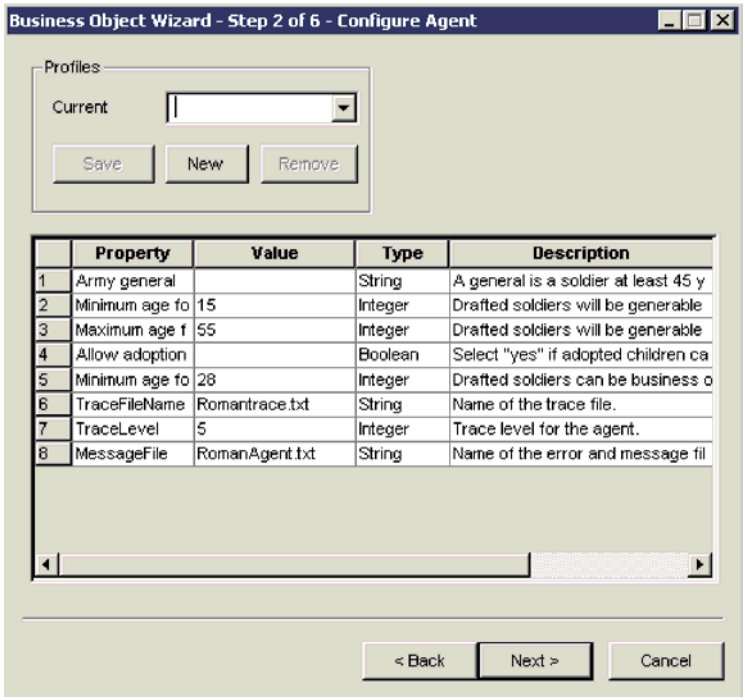


Figure 37. Configure Agent dialog.

6. Specify ODA configuration values or select a profile to display previously saved values. One of the required configuration areas for the ODA is to set up the logging and tracing. For more information, see “Setting up logging and tracing” on page 74.

   The first time you use a particular ODA, you specify values for each of its configuration properties. After doing so, you can save the property values in a named profile by clicking the Save button. The next time you use the same ODA, you can select the saved profile from the “Select profile” box. For more information, see “Entering values and saving a profile” on page 73.
7. Click Next. Business Object Wizard displays the third wizard dialog, Select Source, which is shown in Figure 38. The Select Source dialog displays the *source-node hierarchy*, which is a tree structure with the top-level objects at the top and child objects underneath. In the initial display, the Select Source dialog usually displays only the top-level source nodes.

Important

If the ODA is unable to proceed when you click Next, verify that the ODA message file you have specified for the `MessageFile` configuration property exists in the *`ProgramDir`*`\ODA\messages` directory. For more information, see “Specifying the ODA message file” on page 76.

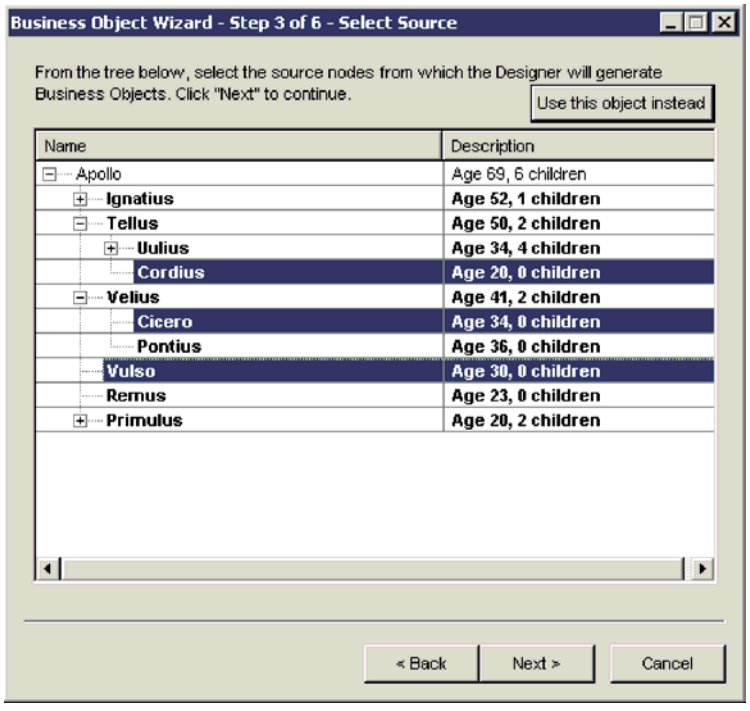


Figure 38. Initial Select Source dialog.

The nodes of the source-node hierarchy can be table names, business object names, schema, or functions, depending on the ODA’s data source. The ODA illustrated in these steps generates nodes from objects within an XML file called `RomanArmy.xml`. Figure 38 shows the single top-level source node for the Roman general specified for the `Army general` configuration property (see Figure 37 on page 67).

8. Select objects in the source-code hierarchy for which you want the ODA to generate business object definitions. To select one source node, click on the node name. To select additional nodes, use Ctrl-click (the Control key and a left-click). In Figure 39, several source nodes have been expanded and three source nodes (which correspond to XML objects) have been selected.

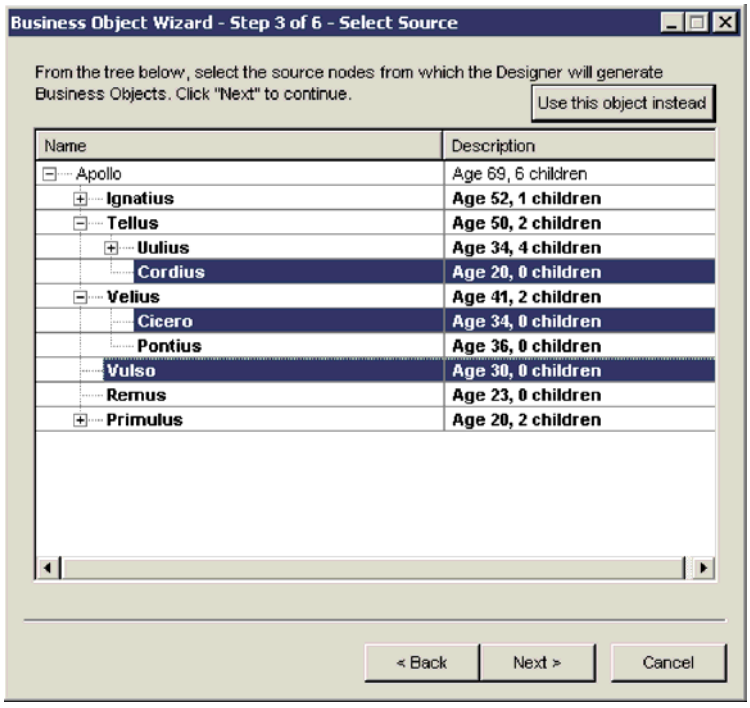


Figure 39. Select Source dialog with source nodes expanded and selected.

To expand a source node to display its children, do either of the following:

- Click the + symbol to the left of its name.
- Right-click on the node's name. Business Object Wizard displays the context menu shown in Figure 40.

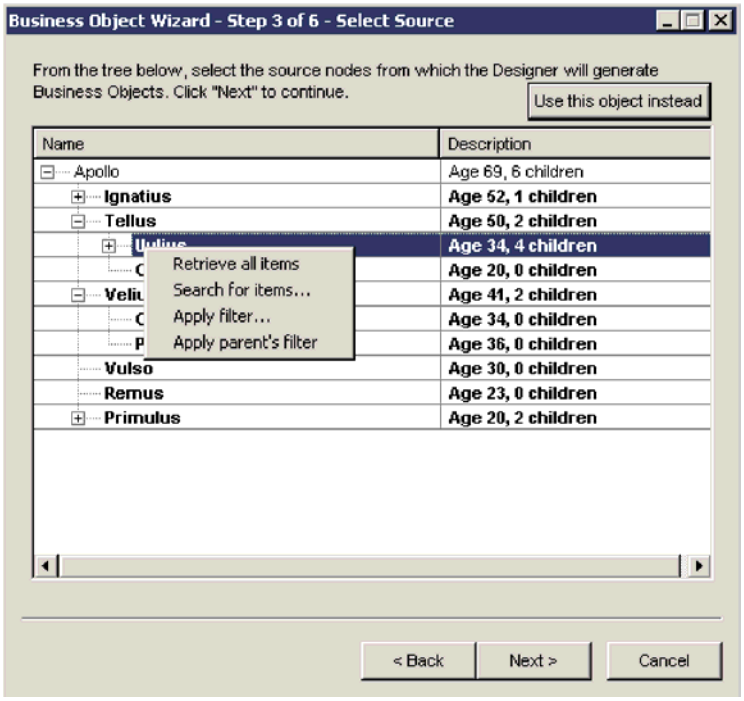


Figure 40. Context Menu for a node.

To expand the selected node, click the "Retrieve all items" menu item. Business Object Wizard displays the next level of source nodes: the child nodes for the expanded parent node. To open lower levels, repeat this process.

Note: Business Object Wizard provides several other mechanisms to move through the nodes of the source-node hierarchy. For more information, see "Moving through the source-node hierarchy" on page 76.

9. After you select the source nodes for which business object definitions are to be generated, click Next. Business Object Wizard displays the fourth wizard dialog, Confirm Source, which is shown in Figure 41. This dialog allows you to confirm your selection of source nodes. Selected source nodes display in a highlighted font. In Figure 41, the source nodes for Cordius, Cicero, and Vulso are highlighted.

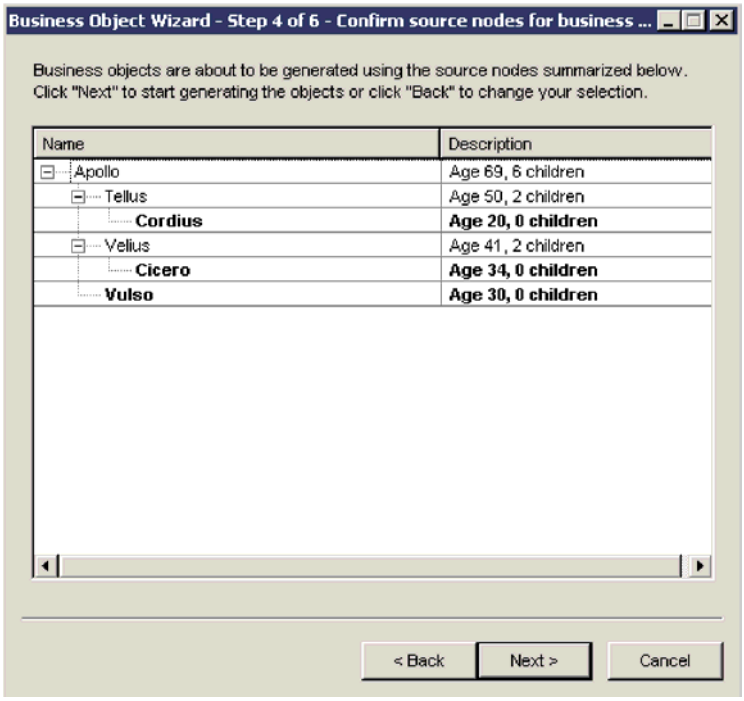


Figure 41. Confirming the objects for which to generate business object definitions.

If your selection is not correct, click Back to return to the previous dialog and make the necessary changes.

10. When your selection is correct, click Next. Business Object Wizard displays the wizard's fifth screen, Generating Business Objects, which is shown in Figure 42. This screen informs you that the ODA is generating the business object definitions.

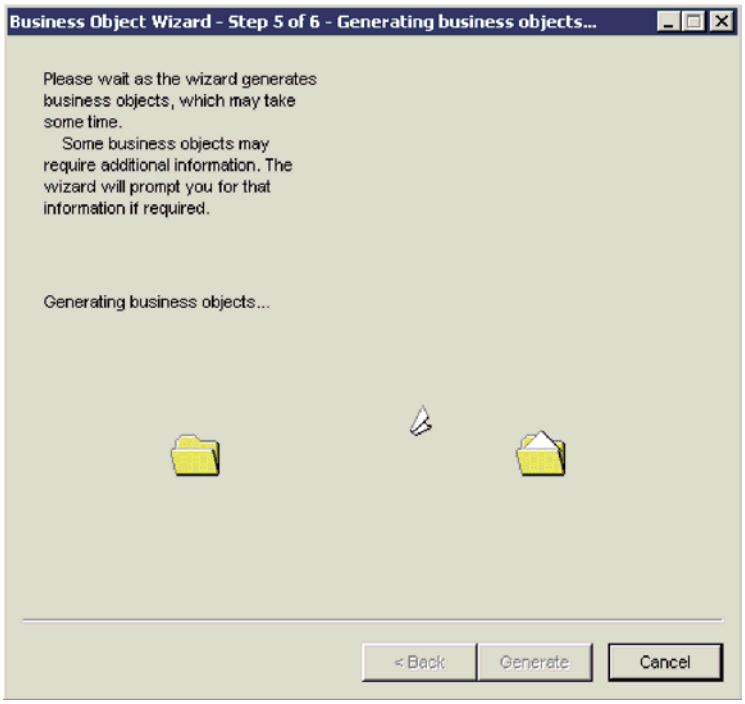


Figure 42. Generating the definitions.

If the ODA needs additional information, Business Object Wizard prompts you for this information by displaying the BO Properties dialog. However, this sample ODA does not require additional information. Therefore, this dialog does not display. For more information about the BO Properties dialog, see "Providing additional information" on page 80.

11. After the ODA completes the generation of business object definitions, Business Object Wizard displays the final dialog in the wizard, Save Business Objects, shown in Figure 43 on page 73. This dialog offers the following options to save the business object definitions that the ODA has generated:
    - Save the business object definitions to an ICL-based project if Business Object Designer Express is running from System Manager.
    - Save the business object definitions to a file.
    - Open the business object definitions for editing in Business Object Designer Express.
    - Shut down the ODA.

Important

If the ODA generates a business object definition from a data-source object that does *not* identify a key element, this business object definition will *not* have a key attribute. Every business object must have *at least one key*. If the ODA might have generated business object definitions that do not include keys, you might want to choose the "Open the new BOs in separate windows" option instead of saving the business object definitions. Within Business Object Designer Express, you can verify that each business object definition has a key attribute, adding one if none exists. Business Object Designer Express does not allow you to save any business object definition that does not include a key.

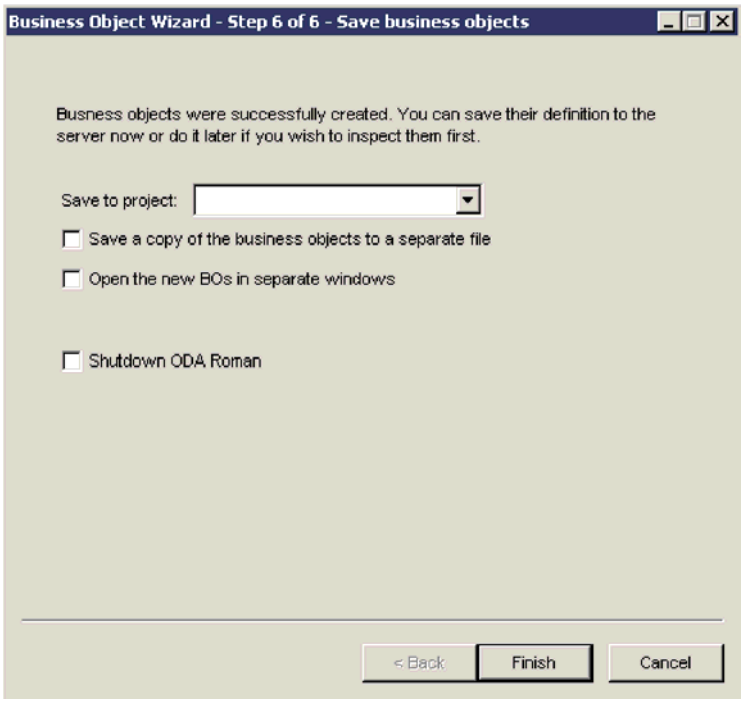


Figure 43. Saving the business object definition.

Click Finish to save the business object definitions or Cancel to exit without saving these definitions. In either case, Business Object Wizard disconnects from the ODA. This dialog also provides the option to have Business Object Wizard shut down the ODA after it disconnects. If you no longer need to use the ODA, click this option.

After you click Finish, if you have told Business Object Wizard to save the business object definitions to a file, Business Object Wizard provides a browse window that allows you to specify the name of this file, where to save it, and what format to use (text file or InterChange Server Express-specific format).

You have now successfully created business object definitions using an Object Discovery Agent.

Entering values and saving a profile

You can save a particular set of ODA configuration values in a profile so that they can be available for future uses of the ODA. To save a profile:

1. Click New, located below the Current field.

   Note: To base a profile on an existing one, locate the desired profile in the profile drop-down list. Do *not* click the New push button.
2. Provide a name for the profile in the Current field (see Figure 37 on page 67 for an illustration).

   Note: If you are basing a profile on an existing one, overwrite the name of the existing profile in the profile drop-down list.
3. Enter the desired configuration values in the Configure Agent table.
4. Click Save, located below the Current field.

Business Object Wizard saves the profile under the following directory:

```
C:\Documents and Settings\All Users\Application Data\CrossWorlds\
   BusObjDesigner\profiles.bod
```

Setting up logging and tracing

As part of the configuration of the ODA, you must set up the logging and tracing. You specify the logging and tracing information for an ODA in the Configure Agent dialog of Business Object Wizard. Business Object Wizard always provides the standard configuration properties (shown in Table 11) for an ODA.

Table 11. Standard ODA configuration properties.

Property name	Property type	Description
`TraceFileName`	String	Specifies the file into which the ODA writes trace information. For more information, see "Specifying the trace file and trace level" on page 74.
`TraceLevel`	Integer	Trace level enabled for the ODA. For more information, see "Specifying the trace file and trace level" on page 74.
`MessageFile`	String	Name of the ODA's error and message file. Use this property to verify or specify an existing file. For more information, see "Specifying the ODA message file" on page 76.

This section provides the following information:

- "Specifying the trace file and trace level"
- "Specifying the ODA message file" on page 76

Specifying the trace file and trace level

Figure 44 shows the Configure Agent dialog in Business Object Wizard, in which you specify the name of the trace file and the trace level.

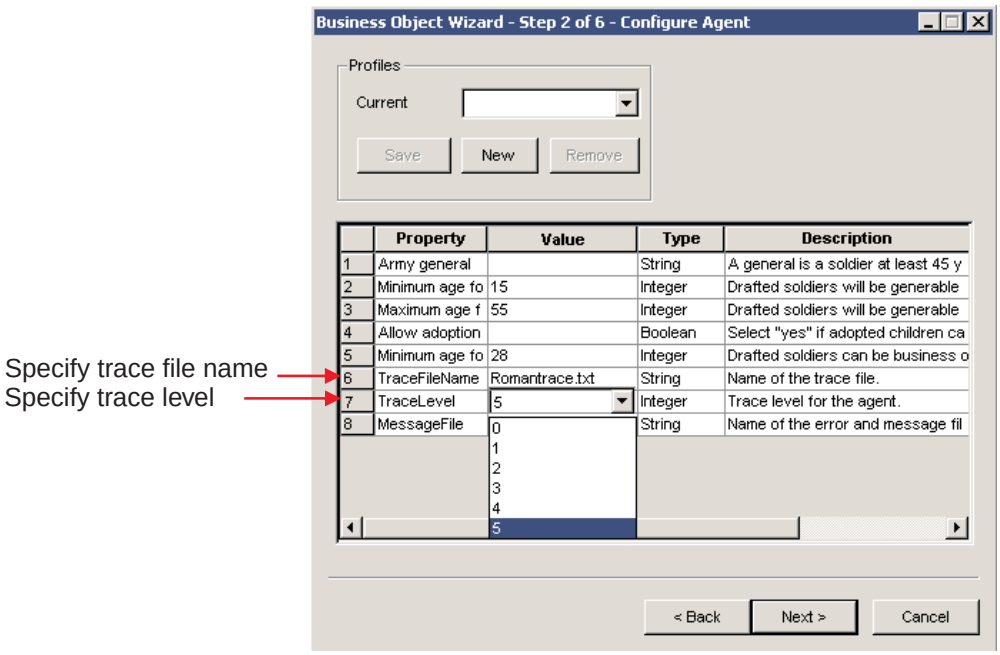


Figure 44. Specifying tracing information.

Specifying a trace file: The `TraceFileName` configuration property specifies the name of the ODA's *trace file*. This file is the destination for *all* trace and error messages that the ODA logs. By default, the ODA runtime names the trace file according to the following naming convention:

ODAname`trace.txt`

In the preceding line, *ODAname* is the name that uniquely identifies the ODA. For example, if the ODA is named `HTMLODA`, it generates a trace file named `HTMLODAtrace.txt`.

Note: An ODA uses only a single file to hold both trace and error messages. Therefore, although this file is called a trace file, it also contains any error messages that the ODA generates.

If the specified trace file does *not* exist, the ODA creates it in the ODA's runtime directory, which is the `ODA\`*`srcDataName`* subdirectory of the product directory. If the specified trace file already exists, the ODA appends to it. When configuring the ODA, you can specify a different name for the trace file by resetting the `TraceFileName` property.

Setting the trace level: The `TraceLevel` configuration property specifies the ODA's *system trace level*. The ODA's trace method sends the specified message to the trace file when the message's trace level is less than or equal to this system trace level. Therefore, the system trace level determines the level of detail that the trace messages provide. Table 12 lists trace levels and their associated behavior.

Table 12. Trace levels

Level	Behavior
0	Writes error messages to the specified trace file.
1	Traces whenever a method is entered—useful for status messages and key information for each business object definition.

Table 12. Trace levels (continued)

Level	Behavior
2	Traces the agent properties and the values received.
3	• Traces the names of the business object. • Traces the business object properties and the values received.
4	• Traces the spawning of all threads. • Traces a message whenever a method is entered and exited.
5	• Indicates the initialization of the Object Discovery Agent and logs the values retrieved for all the Object Discovery Agent properties. • Traces detailed status of each thread spawned by the Object Discovery Agent. • Traces the business object definition dump.

Specifying the ODA message file

The `MessageFile` configuration property specifies the name of the ODA's *message file*. An ODA can store its error and trace messages in this ODA message file. It can then retrieve these messages by message number, instead of creating the message text itself.

By default, the ODA runtime names this message file according to the following naming convention:

ODAname`Agent.txt`

In the preceding line, *ODAname* is the name that uniquely identifies the ODA. For example, if the ODA is named `HTMLODA`, the value of the `MessageFile` property defaults to `HTMLODAAgent.txt`. The message file must reside in the following message-file directory:

ProductDir`\ODA\messages`

Important

If the specified message file does *not* exist or does not exist in the message-file directory, the ODA generates a runtime exception. You must ensure that the message file (which `MessageFile` specifies) exists before you continue with the execution of the ODA.

If the ODA uses a different message file, set the `MessageFile` property to specify a different name for the trace file.

If you create multiple instances of the ODA script or batch file and provide a unique name for each represented ODA, you can have a message file for *each* ODA instance. For more information, see "Using multiple ODAs simultaneously" on page 81.

Moving through the source-node hierarchy

Within the Select Source dialog, Business Object Wizard provides the following mechanisms for moving through the nodes of the source-node hierarchy:

- "Limiting display of child nodes" on page 77
- "Specifying an object path" on page 78
- "Associating an operating-system file" on page 78

Limiting display of child nodes

The ways to expand a source node given in step 8 on page 69 describe how to display *all* children of an expandable node. To limit which objects are displayed, you can use either of the following options from the context menu for a node name (see Figure 40 on page 70):

- Apply filter
- Search for items

Using a filter: The Apply Filter menu item allows you to specify a *filter*, which can limit which of the currently selected source nodes displays. When you click this menu item, Business Object Wizard displays the "Apply filter to node" dialog, as shown in Figure 45.

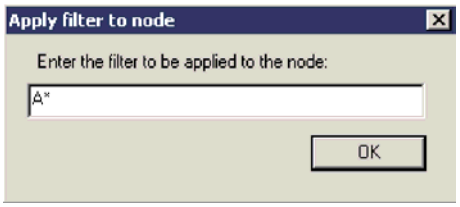


Figure 45. Specifying a filter to limit results.

In the filter text, you can use the asterisk (*) character as a wildcard (to represent zero or more matching characters). This wildcard character can appear in any position and in as many positions as required. For example, `SAP*`, `*SAP`, `*SAP*`, or `*S*AP*`.

When you click OK, Business Object Wizard searches the currently retrieved children of the parent node for those whose names match the filter text. When it expands this parent node, it displays only those children whose names match this text.

Important: When Business Object Wizard receives a filter, it searches for matching children of the parent node in the currently retrieved source node; that is, it does *not* search the data source for matching children. To have Business Object Wizard search the data source, you can specify a search pattern. For more information, see "Specifying a search pattern."

If you specify a filter at the top of a node and then expand the node, you can apply the same filter to child objects by clicking the "Apply parent's filter" menu item from the node's context menu. If you used the "Retrieve all items" menu item, the parent's filter is applied to all elements.

Specifying a search pattern: The "Search for items" menu item allows you to specify a *search pattern*, which can limit which source nodes Business Object Wizard selects from the data source. When you click the "Search for items" menu item, Business Object Wizard displays the "Enter a Search Pattern" dialog. Figure 46 on page 78 illustrates this dialog.

Note: An ODA must support the search-pattern feature for the "Search for items" menu item to be enabled. If this menu item is disabled, the ODA does *not* support search patterns.

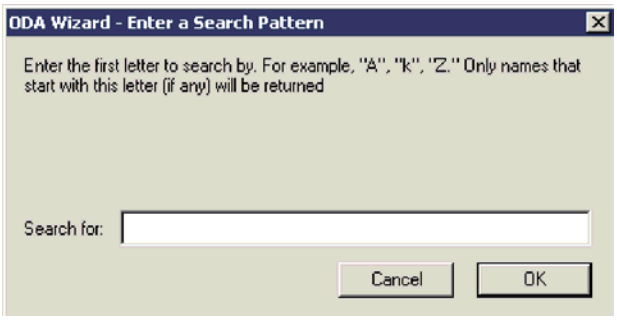


Figure 46. Specifying a search pattern to limit retrieval results.

The Enter a Search Pattern dialog provides a description of the search criteria that your search pattern can use. In Figure 46, the text in this dialog specifies that the search pattern can consist of one letter. The ODA provides a customized description of the search criteria. Make sure that the search pattern you enter follows the described search criteria. Otherwise, the ODA throws an exception.

When you click OK, Business Object Wizard searches the data source for children of the parent node whose names match the search pattern. When it expands this parent node, it displays only those children whose names match this pattern.

Important: When Business Object Wizard receives a search pattern, it searches for matching children of the parent node in the data source; that is, it retrieves a new tree node from the data source. It does *not* simply search the currently retrieved tree node for matching children. To have Business Object Wizard search the currently retrieved tree node, you can specify a filter. For more information, see "Using a filter" on page 77.

Specifying an object path

Instead of moving through the source-node hierarchy, you can specify an exact path for the desired object. To do so, click "Use this object instead", at the upper right of the Select Source dialog. Business Object Wizard displays the Object Path dialog, shown in Figure 47, in which you specify the path.

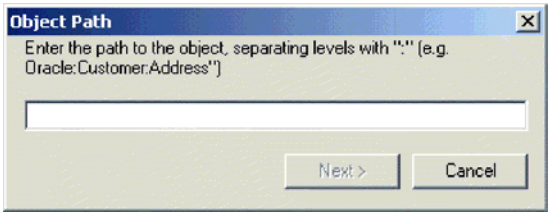


Figure 47. Specifying an object's path.

You specify the object path as the fully qualified path of the source node (from the top-level parent down to the desired node). Node names within this path are separated with a colon (:).

Associating an operating-system file

To associate an operating-system file with the current node of the source-node hierarchy, click the "Associate files" menu item from the context menu for a node name (see Figure 48). When you associate a file with a source node, the ODA uses the file as the source for that source node's data (instead of using the ODA's data source).

Note: An ODA must support the associate-files feature for the "Associate files" menu item to be enabled. If this menu item is disabled, the ODA does *not* support associating files with the current source node.

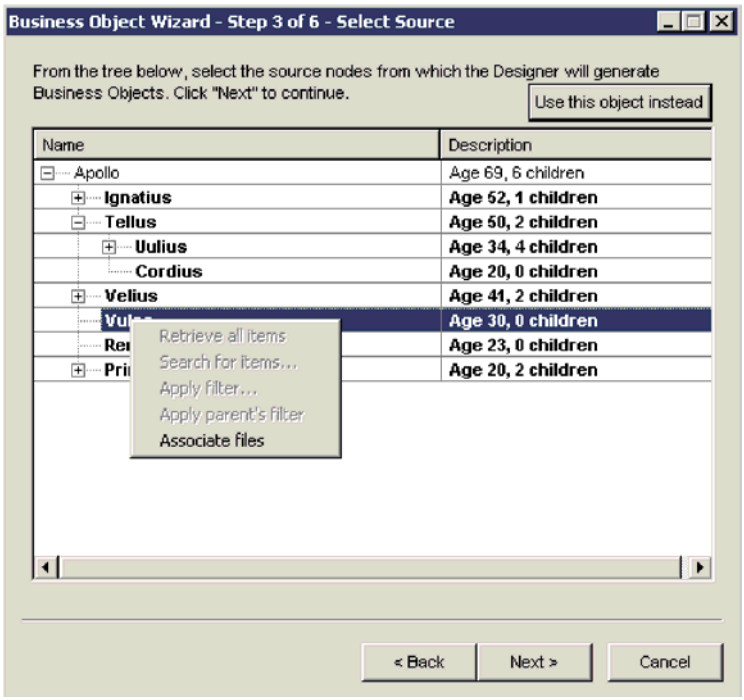


Figure 48. Associating a file with a source node.

When you click the "Associate files" menu item, Business Object Wizard displays the Open window shown in Figure 49. From this window, you can browse the file structure and choose the file to associate with the current node.

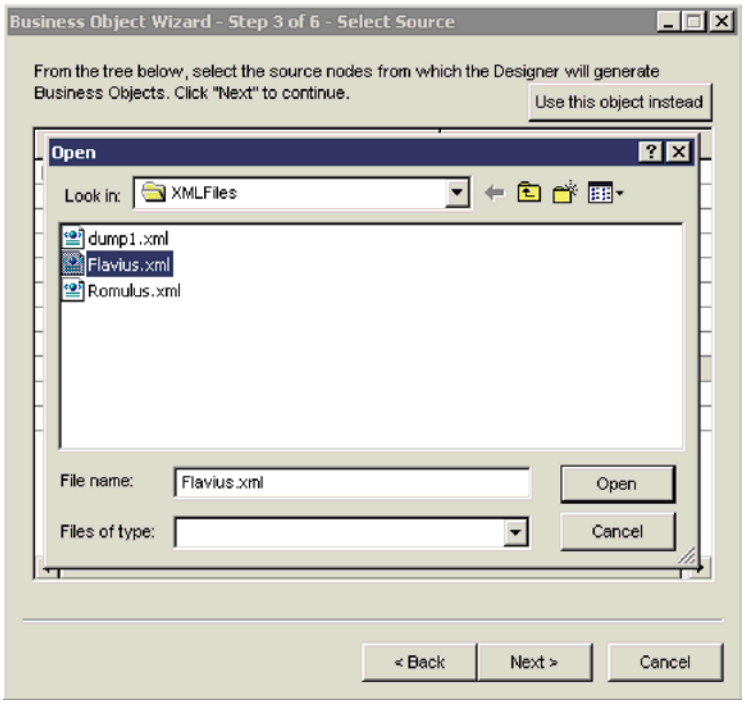


Figure 49. Open window for selecting the file to associate.

Once you have selected the file to associate with the source node, click Open. When Business Object Wizard returns control to the Select Source dialog, the file you selected now displays under the source node with which it is associated, as Figure 50 shows.

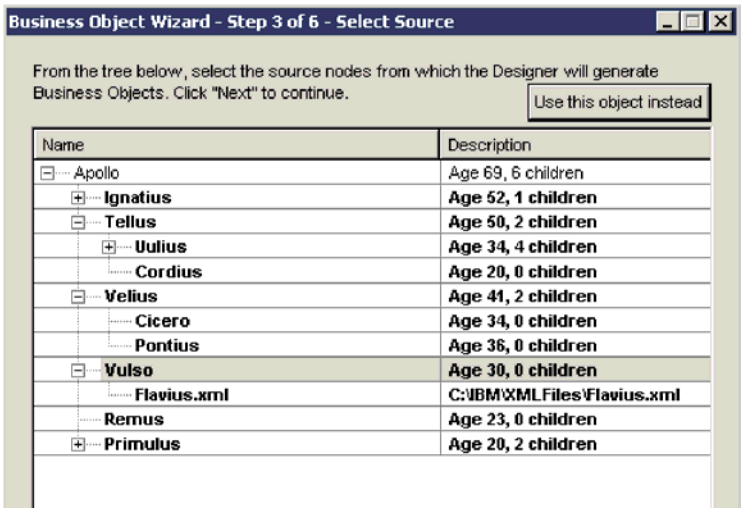


Figure 50. File associated with a source node.

Providing additional information

In Step 5, Generating Business Objects, if the ODA needs additional information, Business Object Wizard prompts you for this information by displaying the BO Properties dialog, as shown in Figure 51.

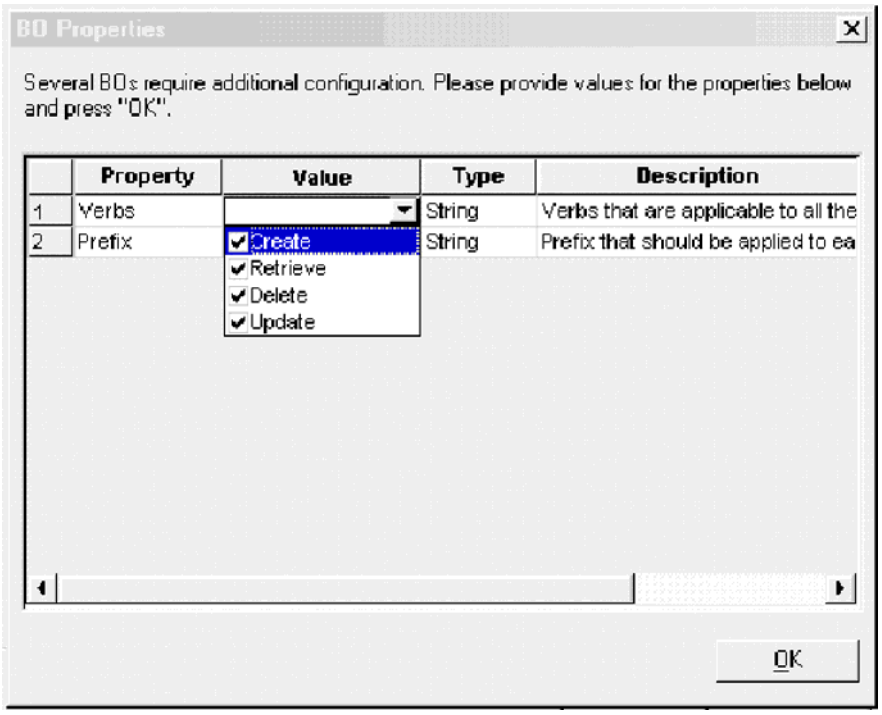


Figure 51. Providing additional information.

Note: If a field in the BO Properties dialog has multiple values, the field appears to be empty when the dialog first displays. Click in the field to display a drop-down list of its values.

After you provide all required information in the BO Properties dialog, click OK. The ODA continues with its generation of business object definitions.

Using multiple ODAs simultaneously

You can run multiple instances of an ODA either on the local host machine or a remote host machine. Each instance runs on a unique port. You can specify this port number when you launch each ODA from within Business Object Wizard.

To run multiple Object Discovery Agents simultaneously in Business Object Designer Express, do the following:

1. Start each Object Discovery Agent by running its `start_`*`ODAname`*`.bat` file.
2. Launch Business Object Designer Express.
3. From the File menu, select the "New Using ODA" menu item.

   Business Object Designer Express invokes Business Object Wizard, which displays the first dialog in the wizard, Select Agent (see Figure 35 on page 66).
4. Click the Find Agents button to display currently running ODAs in the "Located agents" field. You can also find the ODA using its host name and port number.
5. Select the first ODA from the displayed list. Business Object Wizard displays your selection in the Agent's name field.
6. Again, select the "New Using ODA" submenu from the File menu of Business Object Designer Express.

   Business Object Wizard displays a second instance of the Select Agent dialog.

7. Click the Find Agents button to display currently running ODAs in the Located agents field or find the ODA using its host name and port number.
8. Select the second ODA from the displayed list.
9. Proceed with the configuration of each ODA.

If you create multiple instances of the ODA script or batch file and provide a unique name for each represented ODA, you can have a message file for *each* ODA instance. Alternatively, you can have differently named ODAs use the same message file. There are two ways to specify a valid message file:

- If you change the name of an ODA and do *not* create a message file for it, you must change the name of the message file in Business Object Wizard as part of ODA configuration. Business Object Wizard provides a name for the message file but does not actually create the file. If the file displayed as part of ODA configuration does not exist, change the value to point to an existing file.
- You can copy the existing message file for a specific ODA, and modify it as required. Business Object Wizard assumes you name each file according to the naming convention. For example, if the `AGENTNAME` variable (within the ODA startup script) specifies `HTMLODA`, the tool assumes that the name of the associated message file is `HTMLODAAgent.txt`. Therefore, when Business Object Wizard displays the filename for verification as part of ODA configuration, the filename is based on the ODA name. Verify that the default message file is named correctly, and correct it as necessary.

Notices

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

Programming interface information

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

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

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

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

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

IBM
the IBM logo
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Business Object Designer Express includes software developed by the Eclipse Project (http://www.eclipse.org/).

WebSphere Business Integration Express for Item Synchronization V4.3

Index

Printed in USA