IBM WebSphere InterChange Server

Naming Components Guide

Version 4.2.2

IBM WebSphere InterChange Server

Naming Components Guide

Version 4.2.2

Note!
Before using this information and the product it supports, read the information in "Notices" on page 19.

19December2003

This edition of this document applies to IBM WebSphere InterChange Server, version 4.2.2, 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

IBM(R) WebSphere(R) InterChange Server and its associated toolset are used with IBM WebSphere Business Integration Adapters to provide business process integration and connectivity among leading e-business technologies and enterprise applications.

This document guides you through the naming conventions for all components in the IBM InterChange Server system.

Audience

This document is for anyone who develops components for WebSphere InterChange Server system.

Related documents

The complete set of documentation available with this product describes the features and components common to all WebSphere InterChange Server installations, and includes reference material on specific components.

You can install the documentation from the following sites:

- For InterChange Server documentation:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
- For collaborations documentation:
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For WebSphere Business Integration Adapters documentation:
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter

These sites contain simple directions for downloading, installing, and viewing the documentation.

Conventions used in this guide

This document uses these abbreviations and style conventions:

- Abbreviations are used for some words and some variables in syntax statements. These abbreviations are used:

`App` = name of application	`Gen` = generic
`Attr` = name of attribute	`ICS` = InterChange Server
`BO*` = business object (prefix to a BO file)	`ObjectName*` = variable name for business object
`Collab` = collaboration	`Src` = source
`Dest` = destination application	

* `BO` is the prefix to a business object file; the term BO generally refers to the business object definition text files. *`ObjectName`* refers to the variables used to represent business objects.

- The term "camelcase," used in this document, refers to a style of capitalization whereby the spaces between words are eliminated, and the first letter of each

word (or more than one letter if using acronyms) is capitalized. For example, the collaboration template name `DemandForecastVer2` is in camelcase.

In addition, this document uses the following typographical 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.
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.

New in this release

This chapter describes the following new features, which are covered in this document:

New in release 4.2.2

- Add alternative syntax statements for naming maps, submaps, and polymorphic maps.
- Delete the quick-reference chapter from the document.
- Add more field-size restrictions to individual sections.

New in release 4.2.1

No changes were made to the guide for this release.

New in release 4.2

- The ″CrossWorlds″ name is no longer used to describe an entire system or to modify the names of components or tools, which are otherwise mostly the same as before. For example ″CrossWorlds System Manager″ is now ″System Manager,″ and ″IBM CrossWorlds InterChange Server″ is now ″IBM WebSphere InterChange Server.″ For continuity, you might want to keep this abbreviation. this document continues to use abbreviations that include ″Cw″ and ″cw.″
- Sections on tools and data handlers have been added.

New in release 4.1.1

- Add internationalization requirements.
- Add character limits in collaborations and maps sections.

New in release 4.1.0

- Sections on naming Object Discovery Agents (ODA) have been added.
- The document has been revised to reflect the IBM CrossWorlds brand.

New in release 4.0.1

The changes made in IBM CrossWorlds 4.0.1 do not affect the content of this document.

New in release 4.0.0

- The sections have been reorganized in a more logical order.
- Chapter 2, ″Naming components: Quick-reference,″ has been added.
- The section on “Maps” on page 17 has been significantly enhanced.
- Content from Appendix C of the *Map Development Guide*, “Abbreviations and Naming Conventions,” has been incorporated into the sections on naming maps in this guide.

Naming components

This document lists the requirements and guidelines for naming the components in an IBM WebSphere InterChange Server (ICS) system. Following these guidelines assures consistency of component names within an organization. Standardizing component names makes it easier to use the ICS "language"—you can easily tell if a name represents a business object, a collaboration, a connector, and so on.

This chapter includes the following sections:

Although most of the naming conventions in this guide are recommended, some are required. When a naming scheme is required, it is called out as such in the syntax statement.

Important: The information in this document is for new components. For shipped products, retain the existing names, unless otherwise indicated in this document.

General naming conventions

The following general guidelines and requirements apply to naming all ICS components:

- In general, limit component names to combinations of alphanumeric characters and underscores because most punctuation and special characters are not used in an ICS system. See specific sections for exceptions.
- Begin all component names with a letter (not a digit or an underscore). See specific sections for exceptions.
- In property names, do not use apostrophes (') or double quotation marks (").
- Most component names have a maximum length limit. Refer to the sections for the individual components for those restrictions.
- The following components must be composed only of characters in the code set associated with the U.S. English locale, `en_US`: connectors and connector configuration properties, collaboration templates, collaboration objects, collaboration properties, collaboration ports, maps, business object definitions, attributes, attribute types, and verbs.
- In UNIX file names, use underscores (_) instead of spaces between words.

Note: In the interest of backward compatibility, IBM enforces only a few naming requirements. However, because naming is subject to restrictions from

underlying system components, follow the conventions in this guide. For example, Java imposes restrictions on Java class names, and your operating system and database likely have restrictions on the length of and characters used in file names.

Refer to "Conventions used in this guide" on page v for an explanation of the abbreviations and typographical conventions used in this guide.

System objects

This section provides naming conventions for the following system-level objects:

InterChange Server

Required syntax: Unique name in network.

The InterChange Server name:

- Cannot exceed 80 characters.
- Cannot be an internationalized name. For example, you cannot use Japanese characters.
- Must contain only alphanumeric characters, periods (.), or underscores (_)
- Must have a letter, not a numeral, as the first character
- Must not contain spaces

For network authentication, the name must be unique in a network. To avoid the possibility of duplicate names, name the ICS instance after the machine that hosts it. Note that the ICS name is case-sensitive, so `IBM` and `ibm` are different servers.

Examples:

```
IBMDEV
QABOX
QAbox
PRODUCTION
```

Notes:

1. If you have multiple ICS installations on the same machine, append the ICS version to the name, for example, IBMDEV_42 and PRODUCTION_42.
2. If you plan to configure ICS and the adapters as Windows services, make sure that the ICS name is not ″Ignore.″ ICS must be set up as a Windows service dependency for the adapters; the name ″Ignore″ causes Windows services to ignore that dependency.

Integration broker

When you run the ICS installation program, you must name the integration broker. The requirements for naming the broker are:

- The name must be unique in the network.
- The name cannot contain spaces.

WebSphere MQ queue manager

Syntax: *ICSName*.queue.manager

To name the WebSphere MQ queue manager:

1. Start with the name of the ICS machine associated with the WebSphere MQ queue.

   Note: Match the capitalization of the ICS machine.
2. Add .queue.manager.

Do not use hyphens in the name. Each section of the name on either side of a period must be fewer than 42 characters.

Examples:

```
IBMDEV.queue.manager
QABOX.queue.manager
QAbox.queue.manager
PRODUCTION.queue.manager
```

Database

This section includes recommendations for one-database or multi-database environments as well as environments with more than one database. ICS is certified with the DB2, Oracle, and MS SQL databases.

One database

By default, ICS uses one database for its repository, event management, and transactions tables. If the installation includes relationship tables, they also reside, by default, in the ICS database.

Syntax for DB2 database: cwrepos

No specific name is required for the database. However, the name cannot exceed eight characters.

Syntax for Oracle database: cwld

Use cwld for the database name and the Oracle System ID (SID). The SID cannot exceed eight characters. However, you can set a longer database name by setting up a database alias in the tnsnames.ora file. A database alias might include the version of Oracle you are using. For example, if the database SID is cwld, you can define a database alias that further identifies this database as ora817_cwld.

Note: For Oracle versions 8.1.x, the database name cannot exceed four characters.

Syntax for Microsoft SQL database: cwrepos

Multiple databases

Although all database-related activity in ICS defaults to one database (with the default name CwRepos), some production environments partition the database activity.

Syntax:

- Tables of static (more or less) definition information: CwRepos
- Event tables of dynamic work-in-progress activity: CwEvents

- State information for transactional collaborations: `CwTrans`
- Tables of map relationships: `CwRelns`

Notes on database names:

- In early ICS installations, table names typically used `CW` rather than `Cw`. To be consistent with other components that use the `Cw` abbreviation, use the same capitalization style here. This style (`Cw`) was selected for readability.
- For existing installations that use `CWxreferences` for the database partition that contains relationship tables, refer to the *Map Development Guide* for relationship migration information.

Tools

This section covers naming conventions for the following tools:

- "System Manager"
- "Web-based System Monitor"

System Manager

Various components you create through System Manager require that you provide a name: user projects, folders within projects, integration component libraries, database connections, database connection pools, benchmarks, and schedules.

The names of these components:

- Cannot exceed 80 characters
- Must contain only alphanumeric characters, periods (.), or underscores (_)
- Must have a letter, not a numeral, as the first character
- Must not contain spaces

Web-based System Monitor

Web-based System Monitor is made up of two components: a build-time wizard that is launched from System Manager and the Web-based monitoring. You must supply names in both of these components. In addition, at run time, you can create view names.

Build-time wizard

In the build-time wizard, you create monitor names. The names can be 1 to 155 characters long.

Web-based System Monitor installation

During Web-based System Monitor installation, you must provide the following names:

- **Application Server name:** Use ICS Monitor.
- **Enterprise Application name:** For example, InterChange Server Monitor.
- **Context root name:** The name must be ICSMonitor. Enter it in the ″Context root for web module″ field, after the forward slash (/).

Run-time view

At run time, you can create Views in Web-based System Monitor. The View name:

- Cannot exceed 30 characters
- Cannot contain special characters such as hyphens (-), double-quotation marks (″), apostrophes (’), pound signs (#), commas (,), or periods (.)

- Can contain spaces
- Can contain any alphanumeric characters
- Can begin with a numeral

Business objects

The business object naming conventions listed in the following sections are for unmodified business objects. If you customize a business object, you might want to indicate that in its name. For example, you can add an abbreviation for a company or group name. Or you can simply add, for example, _x to indicate that the business object has been modified.

Examples of modified business objects:

```
Customer_x
Item_x
ShipmentsToDate_x
Customer_AE (Acme Enterprises)
SiteAddress_GS (Global Services)
SiteAddress_SC (Star Company)
x_Customer
x_Item
x_ShipmentsToDate
```

Generic business objects

Syntax for top-level business objects: Gen_*GenObjectName*

Syntax for child business objects: Gen_*GenObjectNameChildObjectName*

The generic business object name should accurately identify the type of entity it represents and should not contain spaces. If the entity cannot be described in one word, concatenate all the meaningful words into one, using camelcase style (a capital letter beginning each word).

Examples:

```
Gen_Customer
Gen_Item
Gen_ShipmentsToDate
```

To name child business objects of generic business objects:

1. Start with the name of the parent object.
2. Add the name of the child business object.

For example, to name a child business object of the generic business object `Gen_Customer`, start with `Gen_Customer`, then add the *`ChildObjectName`* name: `Gen_CustomerProfile`.

Application-specific business objects

Syntax for top-level business objects: *App_AppSpecificObjectName*

Syntax for child business objects: *App_ParentObjectNameChildObjectName*

To create the name of an application-specific business object:

1. Start with the name (or abbreviation) of the application.
2. Add an underscore (_).
3. Add concatenated words (in camelcase) to indicate the application entity that the object represents (such as CustomerMaster). For child business objects, include the name of the parent business object.

The name should be as faithful as possible to the application's terminology and differ from it only if the application uses phrasing that is not useful (for example, if the application assigns the name `DataTable1` to the table that contains its customer data, you might consider naming the business object `Customer` instead of `DataTable1`).

Examples of top-level application-specific business objects:

```
SAP_Order
Clarify_BillingRecord
```

Examples of child business objects:

```
SAP_OrderOrgData
Clarify_BillingLine
```

Referenced business objects

Syntax: *Entity*`Ref` or `Related`*Entity*`Ref`

Examples:

```
ContactRef
RelatedVendorRef
```

A business object references another business object in one of two ways:

- A simple attribute of the parent contains another business object's unique identifier (without actually containing the other business object).
- An attribute in the parent contains a child business object or an array of child business objects, each of which contains another business object's unique identifier and may contain other defining information.

In the first case, the parent business object directly references the child. In the second case, the parent business object contains an intermediate business object that references the child.

In both cases, the referenced child is a top-level business object whose name follows the same convention as other top-level business objects. However, the second case requires a naming convention. When the parent contains an intermediate child whose purpose is to reference another business object, name that child in a way that identifies its purpose.

Two naming conventions for business objects reference another business object:

- *Entity*`Ref`—Use when the top-level business object has a multiple-cardinality relationship to the top-level business object it references. For example, the

generic Customer and generic Order business objects contain ContactRef because a Customer or Order can have as many contacts as required.

- Related*Entity*Ref—Use when the top-level business object has a complex, sometimes recursive, relationship to the top-level business object it references. In this case, the referenced business object is often of the same type as the referencing one but serves another role in the relationship. For example, generic Customer and generic Vendor reference other Customer or Vendor business objects that serve different roles. The Customer business object that represents the SoldTo role often contains RelatedCustomerRef, which references Customer business objects in the ShipTo and BillTo roles. The Vendor business object contains RelatedVendorRef for similar reasons.

Note: When applying these conventions to application-specific business objects, follow the standard convention of prefixing the name with the application's name and an underscore.

Business object attributes

Syntax: *AppFieldName* or *AdaptedAppFieldName*

Business object attribute names should be either:

- The exact field name from the application's schema, for example, contact_customer_id
- The field name from the application adapted to the environment's naming style, for example, ContactCustomerId

Note: In business object attribute names, do not use periods (.), left or right brace ([ ]), apostrophes (‘), or double quotation marks (“).

IBM-delivered generic business objects inconsistently name primary key attributes. Some business objects use ObjectId as the name of the primary key attribute. Others use *EntityName*Id. For example, the primary key of generic Contact and generic Order is ContactId and OrderId, respectively, and the primary key of generic Customer and generic Vendor is ObjectId.

Because the primary key of a generic business object might be named ObjectId, there may be a discrepancy in attribute names when one business object references another. In other words, the name of the key attribute of a referenced business object might differ from the name of the attribute that references it. For example, generic Order uses its CustomerId attribute to reference a generic Customer. In this case, Order's CustomerId attribute references Customer's ObjectId attribute. For more information, see “Referenced business objects” on page 6.

Object Discovery Agents (ODA)

Syntax: *HostName+AgentName*

Prefix each ODA name with the name of the host machine on which the ODA runs. This allows multiple copies of the same ODA to run on different machines without causing confusion. *AgentName* is the value of the AGENTNAME parameter (sampleODA in the following example).

Example:

```
HOST2sampleODA
```

Data handlers

When you work with data handlers, you might have to supply top-level meta-object names, data handler names, child meta-object names (with classnames), and corresponding business object attribute names.

Top-level meta-object names

Meta-object names are not enforced. They can be any legal business object name.

Syntax for top-level server meta-object: `MO_Server_DataHandler`

Syntax for top-level connector meta-object: `MO_DataHandler_Default`

Data handler names and classnames

Data handler names and data handler classnames:

- Must begin with a letter, an underscore (_), or a dollar sign ($)
- Can include numerals, but cannot begin with a numeral
- Must not include spaces (tab, word space, linefeed, or carriage return)
- Cannot be a Java keyword

Example:

`SOAP` (data handler name)
`com.ibm.adapters.datahandlers.xml.soap` (classname).

Child meta-object names

Syntax: `MO_DataHandler_Default`*`DatahandlerName`*`Config`

If callers that access a top-level meta-object require more than one behavior of a particular data handler, create a child meta-object with the appropriate configuration information for each data-handler behavior and associate each child meta-object with a unique MIME-type name (in the top-level data-handler meta-object).

Name the child meta-objects and corresponding business object attribute names using the name of the data handler, followed by a unique MIME type/subtype combination. For example, for a data handler named NameValue with a MIME type of text/namevalue, use:

- `MO_DataHandler_DefaultNameValueConfig` as the child meta-object name
- `text_namevalue` as the corresponding business object attribute name.

Note: Because MIME types cannot contain special characters, such as forward slashes (/), you must replace those characters with an underscore (_) in the corresponding attribute names. For example:

MIME type	Attribute name
text/namevalue	text_namevalue
text/xml	text_xml
application/octet-stream	application_octet_stream

Maps

This section provides naming conventions for the following map objects:

Note: The length of any map name may not exceed 76 characters.

Parent-level maps

Syntax: Outbound (application-specific-to-generic) map:
*AppSpecificObjectName*_to_Gen+*GenObjectName*
*AppSpecificObjectName*_Gen+*GenObjectName*

To create the outbound map name:

1. Start with the name of the application-specific business object.
2. Add the string:

   _to_

   or simply use an underscore character.
3. Add Gen to indicate the generic business object. See "Application-specific business objects" on page 6.
4. End with the full generic business object name.

Example:

To map the application-specific business object SAP_CustomerMaster to the generic Customer business object:

SAP_CustomerMaster_to_GenCustomer (or SAP_CustomerMaster_GenCustomer) maps

Syntax: Inbound (generic-to-application-specific) map:
Gen+*GenObjectName*_to_*AppSpecificObjectName*
Gen+*GenObjectName_AppSpecificObjectName*

To create the inbound map name:

1. Start with Gen to indicate the generic business object.
2. Add the full name of the generic business object.
3. Add the string:

   _to_

   or simply use an underscore character.
4. End with the name of the application-specific business object.

Example:

To map the generic Order business object to the Clarify_SFAQuote application-specific business object:

GenOrder_to_Clarify_SFAQuote (or GenOrder_Clarify_SFAQuote)

Note: For parent-level maps, do not use the number 2 to indicate the direction of the transformation, because that might not make sense to an international audience; the left-to-right order of the abbreviated environments indicates the direction. For example:

```
Clarify2IBMCustomer
```

Also, do not use abbreviations that may be popular internally to IBM but unfamiliar outside of it.

Submaps

Syntax: Outbound (application-specific-to-generic) submap:
Sub_*AppSpecificObjectName*_to_Gen+*GenParentObjectName*
Sub_*AppSpecificObjectName*_Gen+*GenParentObjectName*
*AppSpecificObjectName*_Gen+*GenParentObjectName*

To create the outbound submap:

1. Start with the Sub_ prefix. (This step is optional; you can begin the name with the application-specific business object name.)
2. Add the application-specific business object name. See "Application-specific business objects" on page 6.
3. Add the string:

   ```
   _to_
   ```

   or simply use an underscore character.
4. Add Gen.
5. End with the name of the destination generic parent business object.

Examples:

```
Sub_SAP_OrderLineItem_to_GenOrderLineItem
Sub_SAP_OrderLineItem_GenOrderLineItem
SAP_OrderLineItem_GenOrderLineItem
```

Syntax: Inbound (generic-to-application-specific) submap:
Sub_Gen+*GenParentObjectName*_to_*AppSpecificObjectName*
Sub_Gen+*GenParentObjectName_AppSpecificObjectName*
Gen+*GenParentObjectName_AppSpecificObjectName*

To create the inbound submap:

1. Start with the Sub_ prefix. (This step is optional; you can begin the name with the generic business object name.)
2. Add Gen to indicate the generic business object.
3. Add the full name of the generic parent business object.
4. Add the string:

   ```
   _to_
   ```

   or simply use an underscore character.
5. End with the name of the application-specific business object.

Examples:

```
Sub_GenOrderLineItem_to_SAP_OrderLineItem
Sub_GenOrderLineItem_SAP_OrderLineItem
GenOrderLineItem_to_SAP_OrderLineItem
```

Note: For submaps, do not use the number 2 to indicate the direction of the transformation, because that might not make sense to an international audience; the left-to-right order of the abbreviated environments indicates the direction. For example:

```
Sub_IBMOrderLineItem2SAP_OrderLineItem
```

Also, do not use abbreviations that may be popular internally to IBM but unfamiliar outside of it.

Polymorphic maps

Syntax: Outbound polymorphic map:
`Poly_`*`AppSpecificObjectName`*`_to_Gen+`*`GenObjectName`*`_or_Gen+`*`GenObjectName`*
`Poly_`*`AppSpecificObjectName`*`_Gen+`*`GenObjectName`*`_or_Gen+`*`GenObjectName`*
`AppSpecificObjectName``_Gen+`*`GenObjectName`*`_or_Gen+`*`GenObjectName`*

Polymorphic maps can be complex. Not only should their names clearly indicate every generic business object that the application-specific business object maps to, but the names should be consistent within a site. Remember that polymorphic map names have an 80-character limit, so use abbreviations if necessary.

1. Start with `Poly_`. (This step is optional; you can begin the name with the application-specific business object name.)
2. Add the application-specific business object. See "Application-specific business objects" on page 6.
3. Add the string:

   `_to_`

   or simply use an underscore character.
4. Add (alphabetically) the multiple generic objects that might be involved, beginning each with `Gen`, and separating them with the string `_or_`.

Example:

To transform the `Clarify_PartRequest` application-specific business object to either the generic `Order` or generic `Requisition` object, use the following polymorphic map name:

```
Poly_Clarify_PartRequest_GenOrder_or_GenRequisition
```

Syntax: Inbound polymorphic map:
`Poly_Gen+`*`GenObjectName`*`_to_`*`AppSpecificObjectName`*`_or_`*`AppSpecificObjectName`*
`Poly_Gen+`*`GenObjectName_AppSpecificObjectName`*`_or_`*`AppSpecificObjectName`*
`Gen+`*`GenObjectName_AppSpecificObjectName`*`_or_`*`AppSpecificObjectName`*

1. Start with `Poly_`. (This step is optional; you can begin the name with the application-specific business object name.)
2. Add the name of the generic object that serves as a source, prefixing it with the letters `Gen`, just as you would for generic objects in other map naming contexts.
3. Add the string:

   `_to_`

   or simply use an underscore character.
4. Add the names of each application-specific business object that serves as a destination, in alphabetical order and separated from one another by the string:

   _or_

See “Application-specific business objects” on page 6.

Example:

The polymorphic map name
`Poly_GenOrder_Clarify_PartRequest_or_Clarify_SFAQuote`

transforms the generic `Order` object to the application-specific business object `Clarify_PartRequest` or application-specific business object or `Clarify_SFAQuote`.

Note: For polymorphic maps, do not use the number 2 to indicate the direction of the transformation, because that might not make sense to an international audience; the left-to-right order of the abbreviated environments indicates the direction. For example:

`Poly_Clarify_PartRequest2IBMOrder_or_IBMRequisition`

Also, do not use abbreviations that may be popular internally to IBM but unfamiliar outside of it.

Relationships and participants

Identity

Identity relationships associate the unique identifiers of application entities. There is only one unique identifier for any object, and therefore only one relationship instance in an identity relationship for it.

Syntax for identity relationships: Name that reflects the generic business object

Name the relationship so that it reflects the generic business object that the application-specific business objects are transformed to during mapping. Relationship Designer imposes an eight-character limit on identity relationship names: use the full generic object name if it is eight or fewer characters; abbreviate the name or parts of the name if it is more than eight characters.

Examples:

`Customer`—associates objects involved with the generic `Customer` object

`OrdrLine`—associates objects involved with the generic `OrderLineItem` object

Syntax for identity participants: *`AppNameGenObjectName`* or `Gen`*`GenObjectName`*

Relationship Designer imposes an eight-character limit on participant names:

1. Start with up to four characters for the application name. See “Application abbreviations in relationship names” on page 13 for common application abbreviations.

   Note: For generic participants, start with the three-letter abbreviation `Gen`.
2. Add a four-letter abbreviation for the business object name.

Examples:

```
SAPCust
GenCust
ClarQuln
```

Lookup

Lookup relationships associate fields that contain similar data across different applications, but are often represented in different formats.

Syntax for lookup relationships: *GenObjectName*

There may be multiple lookup relationships for any object, so it is inappropriate to name the relationship after the object. Instead, consider naming a lookup relationship after the attribute in the generic business object to which attributes in the application-specific business objects are mapped.

Note: Even though you do not create a participant for the generic object in a lookup relationship, IBM recommends that you use the name of a generic business object as the name of the relationship to indicate the type of data to be referenced.

The objective is to name the relationship in such a way that suggests what the data represents but in a way that is not application-specific. That is typically how a generic object's attribute is named as well. Therefore, you can use the attribute name as the name of the lookup relationship.

Examples:

`State`—associates different application formats for representing state information in an address

`Country`—associates different application formats for representing country information in an address

`Currency`—associates different application formats for representing currency information

Syntax for lookup participants: *AppNameAttrName*

Relationship Designer imposes an eight-character limit on participant names:

1. Start with up to four characters for the application name. See "Application abbreviations in relationship names" for a list of common application abbreviations.
2. Add a four-letter abbreviation for the attribute name in the application-specific business object.

Examples:

`ClarStat`—represents the State field in Clarify

`SAPCtry`—represents the Country field in SAP

`OracCrcy`—represents the Currency field in Oracle

Application abbreviations in relationship names

Use the following abbreviations for applications in participant names:

Application	Abbreviation
Clarify	Clar

Application	Abbreviation
Oracle	Orac
PeopleSoft	Psft
SAP	SAP
Siebel	Sbl

Connector definitions

Use the guidelines in these subsections to name your connector definitions.

Note: The length of the name of a connector cannot exceed 80 characters.

One connector

Syntax: *AppName*`Connector`

To create the name of the connector definition:

1. Start with the name of the application.
2. Add the word `Connector`.
   Make sure there are no spaces between the words.

Examples:

```
OracleConnector
ClarifyConnector
```

Multiple connectors

Syntax: *AppName_VersionNumber*`Connector`

Multiple versions of an application sometimes require multiple connectors. For multiple versions of a connector, name the connector definitions as follows:

1. Start with the name of the application.
2. Add an underscore and an instance number.
3. Add the word `Connector`.

Examples:

`ClarifyConnector`—A connector for Clarify

`Clarify_2Connector`—The second of multiple connectors for Clarify defined for load balancing

`Clarify_3Connector`—The third of multiple connectors for Clarify defined for load balancing

Collaborations

This section provides naming conventions for the following collaboration parts:

- "Collaboration templates" on page 15
- "Collaboration objects" on page 15
- "Collaboration ports" on page 16

Collaboration templates

Syntax: *CollabName*

Describe the collaboration template's function in its name. If the name is more than one word, concatenate the words using camelcase style.

The combined length of the name of a collaboration template and that of its triggering port (internal or external) may not exceed 80 characters.

Note: Make sure the template name is different from the collaboration object it processes. If the names are the same, you cannot access the collaboration object using `repos_copy` with the `-e` (Entity) option because `repos_copy` searches for the collaboration template before searching for a collaboration object.

Examples:

```
CustomerSync
DemandForecastVer2
```

Collaboration objects

Syntax: *CollabName_SrcApp*`_to_`*DestApp* or *CollabName_SrcApp*`To`*DestApp* or *SrcApp*`To`*DestApp_CollabName*

The combined length of the name of a collaboration object and that of its triggering port (internal or external) may not exceed 80 characters.

The collaboration object name should indicate the applications involved, the direction of the flow, and the template on which the collaboration object is based. This naming convention is beneficial particularly for tracing information during troubleshooting. The applications involved is important information. Including the flow direction helps to distinguish problems between two collaborations created for bidirectionality. Including the template on which the object is based helps to distinguish one process from other processes that might exist between identical applications and identical directionality.

Note: Make sure the collaboration object name is different from the template name that processes it. If the names are the same, you cannot access the collaboration business object using `repos_copy` with the `-e` (Entity) option because `repos_copy` searches for the collaboration template before searching for a collaboration object.

To create a collaboration object name:

1. Start with the name of the source application.
2. Add the string `_to_` or `To`.
3. Add the name of the destination application.
4. *(Optional)* Prepend to the source application name the name of the collaboration template on which the object is based, followed by an underscore (_), or append to the destination application an underscore (_) followed by the name of the collaboration template on which the object is based.

Note: As shown in this procedure, the collaboration template name can be either the first or last element of the collaboration object name. Whichever you choose, be sure that the names are consistent throughout your environment.

Examples:

```
SalesOrderProcessing_Clarify_to_SAP
SalesOrderProcessing_ClarifyToSAP
ClarifyToSAP_SalesOrderProcessing
```

Note: This convention works only for two-application collaborations. In other cases, use names that are appropriate for the particular site.

Collaboration ports

Syntax:

`From`	The port that receives the incoming business object and triggers the flow
`To`	The port through which the primary business object of the collaboration is sent to the destination application
`DestApp``Retrieve`	The port through which the collaboration retrieves data in the destination application to compare it to the source business object
`To`*`ReferencedObjectName`*`Wrapper`	The port through which a reference-valued business object is sent for processing by a grouped collaboration object (naming convention for extending a collaboration to delegate an object to a wrapper)
`SrcApp``Retrieve`	The port through which business object data is retrieved from the source application

Note: In complex installations with multiple To and From ports, add a numeral to the end of the port name, or add the application name, name of the system, name of the business object supported by the port, or other text that suggests the port's role.

Repository object files

This section provides naming conventions for the repository files that contain the following objects:

Business objects and children

Syntax: `BO_`*`ObjectName`*`.txt`

Names of business objects cannot exceed 80 characters.

Examples:

```
BO_ClarifySalesOrderStatus.txt
```

Note: There is no underscore after the application name in the business object name.

```
BO_Item.txt
```

For information on naming business objects and their children, see "Business objects" on page 5.

Object Discovery Agents (ODA)

Syntax: *AgentName***Agent.txt**

Example:

```
JDBCODAAgent.txt
```

Maps

Syntax: NM_*MapName***.txt**

Names of maps cannot exceed 76 characters.

Example:

```
NM_SAP_CustomerMaster_to_GenCustomer.txt
```

Note: As of the 4.0.0 release, maps can be saved as XML files. For these XML files, called map definition files, use the same naming conventions as for map repository files. The only difference between the names for these two kinds of files is the file extension—`.cwm` for map definition files; `.txt` for map repository files.

For information on naming maps, see "Maps" on page 9.

Relationships

Syntax: RL_*ObjectName***.txt**

Names of relationships can contain only letters and numbers and must begin with a letter; they cannot exceed eight characters.

Example:

```
RL_Order.txt
```

For information on relationship naming, see "Relationships and participants" on page 12.

Connector objects

Syntax: CN_*ConnectorName***.txt**

Examples:

```
CN_XML.txt
CN_Email.txt
```

For information on naming connector objects, see "Connector definitions" on page 14.

Collaboration templates

Syntax: CT_*CollabName*.**txt**

Example:

`CT_CustomerSync.txt`

Note: As of the 4.0.0 ICS release, collaboration templates can be saved as XML files. For these XML files, called template definition files, use the same naming conventions as for collaboration repository files. The only difference between the names for these two kinds of files is the file extension—`.cwt` for collaboration definition files; `.txt` for collaboration repository files.

For information on naming collaboration templates, see "Collaboration templates" on page 15.

Business object zip files (SAP transport files)

Required syntax: `BO_`*`AppBOName`*`.zip`

When a transport corresponds to a single SAP business object, name the transport file like the business object's repository file, but change the `.txt` extension to `.zip`.

Example:

For `BO_SAP_BOMHeader.txt`:

`BO_SAP_BOMHeader.zip`

When a transport spans multiple objects, use only the text that is shared by the application business objects, then append the word `Objects` before the file extension.

Examples:

- For `BO_SAP_OrderDelvyStatu.txt` and `BO_SAP_ReturnDelvyStatu.txt`:
  `BO_SAP_DelvyStatuObjects.zip`
- For `BO_SAP_SalesOrderRegenVer2.txt`, `BO_SAP_SchedRecRegenVer2.txt`, and `BO_SAP_TransOrdRegenVer2.txt`:
  `BO_SAP_RegenVer2Objects.zip`

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.

Printed in USA