IBM® DB2® Warehouse Manager Standard Edition

Information Catalog Center Administration Guide

Version 8.2

SC27-1125-01

IBM® DB2® Warehouse Manager Standard Edition

Information Catalog Center Administration Guide

Version 8.2

SC27-1125-01

Before using this information and the product it supports, be sure to read the general information under *Notices*.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

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 book

This book can help information catalog administrators adapt and tailor the Information Catalog Center to meet their organization's needs.

Who should use this book

You should read this book if you are responsible for setting up, organizing, populating, customizing, or maintaining an information catalog.

This book assumes that you are familiar with the tasks that users perform within the Information Catalog Center, such as searching and browsing for information. These tasks are described in the *Information Catalog Center Tutorial*, which is available in the DB2 Information Center as an HTML book and on the documentation CD-ROM as a PDF file. The *Information Catalog Center Tutorial* is structured so that you can learn how to use the Information Catalog Center with the aid of sample scenarios that describe a fictitious company. The scenarios show how the users at this company use the information catalog to find the information that they need.

How to use this book

Information catalog administrators need to ensure:

- The descriptive data that users need is available.
- The data is easy to find and use.
- The data is as current as it needs to be.
- The data is protected from unauthorized access.

Unless a specific DB2 Universal Database™ product is named, throughout this book the generic terms "DB2 Universal Database" or "DB2 UDB" are used to denote the DB2 Universal Database that stores your information catalog on your platform of choice.

Administrator task information

The tasks of an administrator are in these categories:

Setting up the information catalog
You probably set up the information catalog when you installed the Information Catalog Center. If you did not set up an information catalog, or if you need to set up a new information catalog, descriptions of these tasks begin on page 1.

Organizing your information resources
You determine what kinds of resources your organization wants to describe in your information catalog. You create object types that describe the characteristics of different kinds of information, and you update and delete these object types as needed. Description of these tasks begins on page 7.

Populating the information catalog
You create objects of various types and place them in your

information catalog. To do this, you translate information into terms users are familiar with. Description of these tasks begins on page 17.

Making the information catalog convenient for users
You create relationships and establish relationships between objects. You also add comments to objects and group objects together in subject areas to make them easier to browse. Additionaly, you associate programs with object types so that users can start programs and retrieve the information they want. Description of these tasks begins on page 23.

Expanding and automating your information catalog
You use the Information Catalog Center tag language to easily work with large amounts of descriptive data at one time. To do this, you import and export tag language files. You might extract descriptive data from your organization's existing database catalogs, modeling tools, and user files. Application programmers can write their own customized extract programs. You combine information catalogs to keep descriptive data current and appropriately synchronized with its sources.

You can keep a log of objects, object types, or relationships that are deleted from your information catalog. You can transfer the log to a tag language file and use it to duplicate the deletions in other information catalogs. For example, you can "shadow" information catalogs in a distributed environment. Description of these tasks begins on page 35.

Maintaining the Information Catalog Center
You might also perform some routine database administration tasks, such as backing up the information catalog, although these tasks are not part of managing the Information Catalog Center. You prevent or solve some of the problems that your users might have with the Information Catalog Center. Description of these tasks begins on page 55.

Performing Information Catalog Center tasks with the user interface or tag language

The Information Catalog Center provides a graphical interface to your information catalog. The Information Catalog Center also provides a tag language, which you can use to perform many of the same tasks. The tag language is more difficult to use because you must learn syntax rules to code a tag language file, but it is especially powerful for performing tasks in bulk.

Throughout this book, Information Catalog Center tasks are described first as you would perform them using the graphical interface. When there is a tag language equivalent for performing the Information Catalog Center task, it is presented following the user interface description.

How to send your comments

Your feedback helps IBM to provide quality information. If you have any comments about this book or other Data Warehouse Center publications, visit the following Web site:

`http://www.ibm.com/software/data/db2/warehouse/`

The Web site has a feedback page where you can enter and submit your comments.

Chapter 1. Setting up an information catalog

This chapter gives you an overview of the tasks that are necessary to set up an information catalog. You probably set up an information catalog when you installed the Information Catalog Center. If you did not set up an information catalog during installation, or if you want to create a new information catalog, you can use the steps in this chapter.

Information Catalog Center

The Information Catalog Center allows you to manage descriptive data (business metadata) through information catalogs. The descriptive data (organized into metadata objects) helps you identify and locate information. You can search for specific objects in the information catalog and view any relationships an object participates in or an object's lineage. You can also create comments for objects. Some users can also define additional objects in the information catalog.

The Information Catalog Center helps administrators organize the metadata objects by requiring that each object be based on an object type and by allowing administrators to define relationship types and additional object types.

The Information Catalog Center provides security at the object level, so your privileges are set for each object, allowing greater control of business information.

You can also use some of the Information Catalog Center functions through the Information Catalog Center for the Web. For more information about installing the Information Catalog Center for the Web, see the *DB2® Warehouse Manager Installation Guide*.

Related concepts:

- "Object-level security" on page 1

Related tasks:

- "Installing the Information Catalog Center" in the *DB2 Warehouse Manager Standard Edition Installation Guide*
- "Getting started with the Information Catalog Center" on page 2
- "Starting the Information Catalog Center" on page 3

Object-level security

In the Information Catalog Center, you can implement security at the object level. For each object in your information catalog, you can select which users and groups have privileges to the object and what kind of access the user or group has. There are three levels of privileges to objects:

- Show—Any user or group with only this privilege can only see that the object exists. A user or group with this access cannot read the properties of the object if they do not have any of the other privileges.
- Read—A user or group with only this privilege can see that the object exists and can also view the object's properties. However, the user or group cannot change any of the object's information unless they also have the Write priviliege.

- Write—A user or group with this access can see that the object exists, view the object's properties, make changes to the object's information, and delete the object's metadata from the information catalog.

You can set security for an object on the Privileges page of the Define Object or Object Properties windows. If you are an administrator, you can also set security for an object by viewing the properties of a user or group and setting the security access to specific objects. (You can perform those tasks on the Privileges page of the Users or Groups notebook.)

When you define an object, certain groups have default privileges to the new object:

- Administrators—Have show, read, and write privileges to all objects by default. You cannot change the privileges for this group.
- Default power user group—Has show and read privileges to all objects by default. This group can also define objects and if the group has been given write privileges to an object, the group can also delete, update, or copy the object.
- Default user group—Has show and read privileges to all objects by default. This group can create comments for any object that they have read privileges to.

Related concepts:

- "User, user ID and group naming rules" in the *Administration Guide: Implementation*
- "Objects" on page 17

Related tasks:

- "Setting the default user and power user groups: Information Catalog Center help"

Getting started with the Information Catalog Center

As soon as you have a Version 8 information catalog, you can start using the Information Catalog Center to organize your information resources. You probably prepared a new information catalog or migrated your Version 7 metadata to a Version 8 information catalog during the Information Catalog Center installation process. If you did not prepare or migrate an information catalog, you can do it from the Manage Information Catalog wizard.

Prerequisites:

If you did not do it during the installation process, prepare a Version 8 information catalog, or migrate your Version 7 metadata to a Version 8 information catalog.

Procedure:

To begin working with the Information Catalog Center, start by setting the object-level security so that only authorized users can access objects.

Related concepts:

- "How partner applications can work with the Data Warehouse Center and the Information" in the *Data Warehouse Center Application Integration Guide*
- "Information Catalog Center" on page 1
- "Object-level security" on page 1

Related tasks:

- "Preparing an information catalog" on page 3
- "Migrating metadata from an existing information catalog" on page 4
- "Starting the Information Catalog Center" on page 3

Starting the Information Catalog Center

If you are using Windows, you can start the Information Catalog Center from the Start menu. If you are using a UNIX supported platform, you can start the Information Catalog Center from a command line.

Prerequisites:

The Information Catalog Center must be installed and configured.

Procedure:

To start the Information Catalog Center, use one of the following methods depending on your operating system:

- From the Windows desktop, click **Start → IBM DB2 → Business Intelligence Tools → Information Catalog Center**
- At a UNIX command prompt, enter `db2icc`

Related concepts:

- "Information Catalog Center" on page 1

Related tasks:

- "Getting started with the Information Catalog Center" on page 2

Preparing an information catalog

Information catalogs store the metadata for your organization's information. You probably created an information catalog during Information Catalog Center installation. You can use these instructions to prepare an additional information catalog. When you prepare an information catalog, you are creating an information catalog and initializing it for use.

Prerequisites:

- The Information Catalog Center must be installed and configured.
- The DB2 Warehouse Manager must be installed.
- There must be a DB2 Universal Database cataloged on the local workstation.
- You must be a database administrator for DB2 Universal Database.

Procedure:

To prepare an information catalog:

1. Use one of the following methods to start the Manage Information Catalog wizard:
   - From the Windows desktop, click **Start → IBM DB2 → Set-up Tools → Manage Information Catalog wizard**.
   - At a command prompt, enter `db2iccwz`

The Manage Information Catalog wizard opens.

2. Select the **Prepare an information catalog** option.
3. Enter the required information on each page of the wizard and click **Finish**.

   The information catalog is created and initialized. You are now ready to use your information catalog.

Related tasks:

- "Preparing an information catalog from a command line" on page 119
- "Migrating metadata from an existing information catalog" on page 4

Migrating metadata from an existing information catalog

You can migrate metadata from a DB2 Version 7 information catalog to the DB2 Version 8 metadata format. When you migrate, your Version 7 metadata is not modified. The contents of your Version 7 metadata are retrieved and used to create the same metadata in the Version 8 format.

Prerequisites:

The Information Catalog Center must be installed and configured.

You must be a database administrator for DB2 Universal Database.

Procedure:

To migrate metadata from an existing information catalog:

1. Use one of the following methods to start the Manage Information Catalog wizard:
   - From the Windows desktop, click **Start** → **IBM DB2** → **Set-up Tools** → **Manage Information Catalog wizard**.
   - At a command prompt, enter `db2iccwz`

   The Manage Information Catalog wizard opens.
2. Select the **Migrate metadata from an existing information catalog** option.
3. Enter the required information on each page of the wizard and click **Finish**.

   The Version 7 information catalog metadata is migrated to a Version 8 information catalog. You are now ready to use your information catalog.

Related tasks:

- "Preparing an information catalog" on page 3

Creating DB2 Version 7 views for a DB2 Version 8 information catalog

Create DB2 Version 7 views if you have applications that use SQL to access Version 7 information catalog metadata tables. For example, you might use views if you wrote reports on the metadata in Version 7. You must create DB2 Version 7 views for each DB2 Version 8 information catalog that your applications access. Use of DB2 Version 7 views compromises DB2 Version 8 object-level security because Version 7 views have no security enablement. Applications that access the DB2 Version 7 views will be able to access any object, regardless of security restrictions. DB2 Version 7 views are read-only.

Prerequisites:

You must have a DB2 Version 8 information catalog.

You must be a database administrator for DB2 Universal Database.

There must not be a DB2 Version 7 information catalog in the same database.

Procedure:

To create DB2 Version 7 views for a DB2 Version 8 information catalog:

1. Use one of the following methods to start the Manage Information Catalog wizard:
   - From the Windows desktop, Click **Start** → **IBM DB2** → **Set-up Tools** → **Manage Information Catalog wizard**
   - At a command prompt, enter `db2iccwz`

   The Manage Information Catalog wizard opens.
2. Select the **Maintain DB2 Version 7 views for a DB2 Version 8 information catalog** option.
3. Select the **Create Views** option.
4. Enter the required information on each page of the wizard and click **Finish**.

   The DB2 Version 7 views are created.

Related tasks:

- "Deleting DB2 Version 7 views" on page 5

Deleting DB2 Version 7 views

You can delete any DB2 Version 7 views you have created. You must identify the Version 8 information catalog that contains the views that you want to delete. Applications that use SQL based on the Version 7 data model, will no longer be able to access your Version 8 information catalog.

Prerequisites:

You must have created DB2 Version 7 views for a DB2 Version 8 information catalog.

You must be a database administrator for the database that contains the Information Catalog that you want to delete views from.

Procedure:

To delete DB2 Version 7 views:

1. Use one of the following methods to start the Manage Information Catalog wizard:
   - From the Windows desktop, Click **Start** → **IBM DB2** → **Set-up Tools** → **Manage Information Catalog wizard**
   - At a command prompt, enter `db2iccwz`

   The Manage Information Catalog wizard opens.
2. Select the **Maintain DB2 Version 7 views for a DB2 Version 8 information catalog** option.

3. Select the **Delete Views** option.
4. Enter the required information on each page of the wizard and click **Finish**.
   The DB2 Version 7 views are deleted.

Related tasks:

- "Creating DB2 Version 7 views for a DB2 Version 8 information catalog" on page 4

Registering a server node using the DB2 Control Center

An information catalog can either be local (stored on your workstation) or remote. If an information catalog is remote, you must register the server where the catalog resides. You must register the server on each workstation that accesses the information catalog. If you previously connected to a database that resides on the server on which your remote information catalog resides, you can skip this task.

Procedure:

Use the DB2 Control Center to register a server node. Information catalog administrators or remote database administrators can use the DB2 Control Center to complete the following tasks.

- Add a system
- Add an instance
- Add a database

Chapter 2. Organizing your information resources

After you create an information catalog, you must complete some preparatory steps so that you can populate your information catalog with descriptive data about your business information.

Start by organizing the information that you want to include. For example, you can plan to include descriptive data about your organization's personnel records, financial spreadsheets, building plans, and digital images from advertising campaigns. Each of these items is a different type of information resource.

After you categorize the types of information that you want to include in your information catalog, you identify the type of information in your information catalog.

Object types

To organize your information resources in the information catalog, you use object types. An *object type* is a classification for objects that is used to reflect a type of business information, such as a table, report, or image. For example, you might create an object type called Image, which describes a set of objects that are digital bitmap images. For each object type, you define a set of *properties* that describe the characteristics of the object type. For an object type that is called Image, you might define properties such as Resolution, Size, and Color.

Selected properties of an object type make up the definition for the unique identifier.

Information Catalog Center has a set of predefined object types that are included with the product and ready for you to use in your organization. You can also create your own object types.

Every object type has a set of possible relationship types. A relationship type determines what roles objects can play in a relationship. For example, the Table object type with a relationship type of Contains could play the role of parent or child in a hierarchical relationship.

Object type prototypes

When you create your own object types, create a prototype for each object type that you need first. Then create one or two sample objects. Try entering different values for each property to be sure that you have the right data types and sizes. You might want to consult with your database administrator and some of your users to ensure that the properties you specify meet your work group's needs.

If you are not satisfied with your prototype, delete the object type and your sample objects and start over. You define properties for an object type when you create it. After you create an object type, you can add additional properties to it if they are not required properties. However, the only way to change or delete properties on an object type is to delete the object type and all objects of that type. You must then create a new object type.

Related concepts:

- "Unique identifiers" on page 8
- "Relationship types" on page 23

Related tasks:
- "Defining an object type" on page 9
- "Deleting an object type" on page 15
- "Updating an object type" on page 12

Related reference:
- "Information Catalog Center predefined object types" on page 95

Unique identifiers

All object types must have at least one property that is part of the *unique identifier*. The unique identifier is a set of properties that help you distinguish objects. The unique identifier enables you to import the contents of one information catalog into another.

For example, in an information catalog for your manufacturing division, an object named Product List shows all products that are manufactured by the division. The sales division's information catalog might also have an object named Product List that shows all products sold by the sales division.

Because these objects share a name, without a way to uniquely identify them you risk overwriting the metadata when you combine information catalogs.

The Information Catalog Center prevents overwriting by having you define the unique identifier. You do not have to create unique names on your own or know what every object in another information catalog is called.

You choose up to sixteen properties in an object type. The values for each of these properties, in the order you give them, become the unique identifier for any object of that type.

When you import an object into your information catalog, if there is MERGE tag in the file you are importing, the Information Catalog Center compares the values of the unique identifier properties to see if they match those of an existing object. If all the unique identifier properties have the same value in both objects, the Information Catalog Center treats the two as the same object. It updates the values in the existing object's non-unique identifier properties. If the unique identifier's properties have different values, the Information Catalog Center adds the incoming object to the information catalog.

You can also use the unique identifier along with the object type to find a specific object through the Information Catalog Manager APIs.

In Information Catalog Manager Version 7, the unique identifier was called a universal unique identifier, or UUI.

Related concepts:
- "Object types" on page 7

Related tasks:
- "Defining an object type" on page 9

Defining an object type

You can define an object type by using the Information Catalog Center windows or tag language. You can define object types that meet the needs of your organization. When you define an object type, you give the object type a name and icon. You also provide the following information for the object type:

- Assign properties and permitted values for properties.
- Specify which properties make the unique identifier.
- Specify in what types of relationships objects of this type can participate, as well as what roles they can play in these relationship types..
- Associate programs for objects of this type.

Prerequisites:

Before you perform this task, you should have created and opened an information catalog.

You must have administrator privileges in Information Catalog to perform this task.

Procedure:

To define an object type using the Information Catalog Center:

1. Expand the **Administration** folder in the main Information Catalog window.
2. Right-click the **Object types** folder.
3. Click **Define –> Object Type using Wizard** to use a wizard to define an object type, or click **Define –> Object Type** to use a notebook to define an object type.

   The Define Object Type wizard or notebook opens.

To define an object type using the Information Catalog Center tag language:

1. Enter the following lines in your tag language file:

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE(object_type_short_name)
     EXTNAME(object_type_name)
     DESCRIPTION(object_type_description)
     ICWFILE(Icon_filename)
```

   After each keyword, type an appropriate value within the parentheses:

   Keyword
   Value

   TYPE The short name of the object type. The rules for object type short names are:

   - Case-sensitive.
   - 18 character (SBCS) maximum.
   - First character must be uppercase or lowercase English alphabetic, @, or #
   - Subsequent characters must be uppercase or lowercase English alphanumeric, @, # or _. No $, leading or embedded blanks.
   - It must be unique to the information catalog.

EXTNAME
An extended descriptive name of the object type. The rules for the name are:

- 200 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

DESCRIPTION
A description of the object type. The rules for the description are:

- 254 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

ICWFILE
The name of the .gif icon file, including its extension. You give the drive and path information where the icon file exists as part of the IMPORT command when you import your tag language file.

2. Type a line for each property you want to give your object type:

```
:PROPERTY.SHRTNAME(short_name) DT(data_type) DL(size)
     UUISEQ(position_in_UUI) NULLS(y_or_n) EXTNAME(property_name)
```

Keyword
Value

SHRTNAME
The property short name. The rules for property short names are:

- Case sensitive.
- 18 character (SBCS) maximum.
- First character must be uppercase or lowercase English alphabetic, @, or #
- Subsequent characters must be uppercase or lowercase English alphanumeric, @, # or _. No $, leading or embedded blanks.
- It must not be an SQL reserved word.
- It must be unique.

DT The data type:

I (INTEGER)
4 bytes

S (SMALLINT)
2 bytes

G (BIGINT)
8 bytes

E (DECIMAL)
16 bytes

U (DOUBLE)
8 bytes

R (REAL)
4 bytes

B (BLOB)
0 bytes to 2 gigabytes of bytes

O (CLOB)
0 bytes to 2 gigabytes of characters

C (CHAR)
Up to 254 characters

V (VARCHAR)
Up to 4 000 characters

L (LONG VARCHAR)
Up to 32 700 characters

T (TIMESTAMP)
26 characters, in this format:

yyyy-mm-dd-hh.mm.ss.nnnnnn

M (TIME)
15-character time in the following format:

hh.mm.ss.nnnnnn

D (DATE)
10-character date in the following format:

yyyy-mm-dd

DL The size for the property.

UUISEQ
The position this property has in the UI: **1**— **16**. Include this keyword only if you want the property to be part of the UI.

NULLS
Entry required?

N Entry is required

Y Entry is not required

EXTNAME
The name of the property type. The rules for the name are:

- 200 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

If you want to make the NAME property part of the unique identifier for this object type, you can use only the keywords SHRTNAME and UUISEQ for the property. The Information Catalog Center defines values for other keywords, so you do not specify them or their values here.

After you add all properties for your object type, the tag language file looks similar to the following example.

```
:COMMENT.------------------------------------------------------------
:COMMENT.Generating the report object definitions.
:COMMENT.------------------------------------------------------------
:ACTION.OBJTYPE(MERGE)
:OBJECT.TYPE(REPORT) CATEGORY(ELEMENTAL)
    EXTNAME(Text based reports) ICWFILE(flgnyrep.gif)
    DESCRIPTION(No word processor based reports)
:PROPERTY. SHRTNAME(NAME)                          UUISEQ(0)
:PROPERTY. SHRTNAME(SHRTDESC)   DT(V)   DL(250)    UUISEQ(0)   NULLS(Y)
    EXTNAME(Short description)
```

```
:PROPERTY. SHRTNAME(LONGDESC)   DT(L)   DL(32700)  UUISEQ(0)   NULLS(Y)
    EXTNAME(Long description)
:PROPERTY. SHRTNAME(ACTIONS)    DT(V)   DL(254)    UUISEQ(0)   NULLS(Y)
    EXTNAME(Actions)
:PROPERTY. SHRTNAME(TITLE)      DT(V)   DL(254)    UUISEQ(0)   NULLS(N)
    EXTNAME(Report title)
:PROPERTY. SHRTNAME(RPRTDATE)   DT(C)   DL(26)     UUISEQ(0)   NULLS(Y)
    EXTNAME(Report publication date)
:PROPERTY. SHRTNAME(RPRTFRMT)   DT(V)   DL(80)     UUISEQ(0)   NULLS(Y)
    EXTNAME(Report presentation format)
:PROPERTY. SHRTNAME(DBPRESNT)   DT(V)   DL(254)    UUISEQ(0)   NULLS(Y)
    EXTNAME(Report presentation requirements)
:PROPERTY. SHRTNAME(OWNER)      DT(V)   DL(80)     UUISEQ(0)   NULLS(Y)
    EXTNAME(Report owner)
:PROPERTY. SHRTNAME(FILENAME)   DT(V)   DL(254)    UUISEQ(1)   NULLS(N)
    EXTNAME(Report filename)
:PROPERTY. SHRTNAME(TYPE)       DT(V)   DL(80)     UUISEQ(2)   NULLS(N)
    EXTNAME(Report class or type)
:PROPERTY. SHRTNAME(URL)        DT(V)   DL(254)    UUISEQ(0)   NULLS(Y)
    EXTNAME(URL to access data)
```

Related concepts:

- "Unique identifiers" on page 8

Related tasks:

- "Deleting an object type" on page 15
- "Updating an object type" on page 12

Related reference:

- "OBJECT" on page 81
- "Information Catalog Center predefined object types" on page 95
- "ACTION.OBJTYPE" on page 69
- "PROPERTY" on page 86

Updating an object type

You can update an object type using the Information Catalog Center windows or tag language. You can change the name, icon, and value list for a property of the object type. You can also add properties to the object type, but any properties you add can only be optional; they cannot be required. The Update Object Type window also allows you to associate and disassociate programs.

You cannot remove a constraint entry from the list if it is being used as the value of a property by an object. You must first change the value of all objects with properties of that value to enable the desired constraint change. Search the object type for objects of that value to find objects that need to be changed.

Prerequisites:

You must have defined some object types in your information catalog.

You must have administrator privileges to the information catalog.

Procedure:

To update an object type using the Information Catalog Center windows:

1. Expand the **Administration** folder in the main Information Catalog window.

2. Click the **Object types** folder and list the object types.
3. Right-click the object type that you wish to update.
4. Click **Properties**.

   The Update Object Type window opens.

To update an object type using the Information Catalog Center tag language:

1. Enter the following lines in your tag language file:

   ```
   :ACTION.OBJTYPE(UPDATE)
   :OBJECT.TYPE(object_type_short_name)
   ```

2. To change the object type name, add the following line:

   ```
   EXTNAME(new_object_type_name)
   ```

3. To change the object type description, add the following line:

   ```
   DESCRIPTION(new_object_type_description)
   ```

4. To change the object type's icon, add the following line:

   ```
   ICWFILE(new_icon_filename)
   ```

   After each keyword, type an appropriate value within the parentheses:

   Keyword
   Value

   TYPE The short name of the object type you are updating.

   EXTNAME
   The new name of the object type. The rules for the name are:
   - 200 character maximum.
   - It must not contain null characters.
   - It must not be all blank characters.

   DESCRIPTION
   The new description of the object type. The rules for the description are:
   - 254 character maximum.
   - It must not contain null characters.
   - It must not be all blank characters.

   ICWFILE
   The name of the new .gif icon file, including its extension. You give the drive and path information where the icon file exists as part of the IMPORT command when you import your tag language file.

5. To add an optional property, type the following lines in your tag language file:

   ```
   :ACTION.OBJTYPE(APPEND)
   :OBJECT.TYPE(object_type_short_name)
   :PROPERTY.SHRTNAME(new_property_short_name) DT(data_type) DL(size)
       UUISEQ(0) NULLS(y) EXTNAME(new_property_name)
   ```

   After each keyword, type an appropriate value within the parentheses.

   Any property you add to an object type after you create it must be an optional property. The value for UUISEQ must be 0, and the value for NULLS must be Y.

   Keyword
   Value

   TYPE The short name of the object type you are updating.

   SHRTNAME
   The property short name. The rules for property short names are:

- Case sensitive.
- 18 character (SBCS) maximum.
- First character must be uppercase or lowercase English alphabetic, @, or #
- Subsequent characters must be uppercase or lowercase English alphanumeric, @, # or _. No $, leading or embedded blanks.
- It must not be an SQL reserved word.
- It must be unique; if you type a name that already exists in this object type, the Information Catalog Center asks you for another name.

DT The data type:

I (INTEGER)
4 bytes

S (SMALLINT)
2 bytes

G (BIGINT)
8 bytes

E (DECIMAL)
16 bytes

U (DOUBLE)
8 bytes

R (REAL)
4 bytes

B (BLOB)
0 bytes to 2 gigabytes of bytes

O (CLOB)
0 bytes to 2 gigabytes of characters

C (CHAR)
Up to 254 characters

V (VARCHAR)
Up to 4 000 characters

L (LONG VARCHAR)
Up to 32 700 characters

T (TIMESTAMP)
26 characters, in this format:

yyyy-mm-dd-hh.mm.ss.nnnnnn

M (TIME)
15-character time in the following format:

hh.mm.ss.nnnnnn

D (DATE)
10-character date in the following format:

yyyy-mm-dd

DL The size for the property.

UUISEQ
The position this property has in the UI: **1**— **16**. Include this keyword only if you want the property to be part of the UI.

NULLS
Entry required?

N Entry is required

Y Entry is not required

EXTNAME
An extended descriptive name of the property. The rules for the name are:

- 200 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

Related tasks:

- "Defining an object type" on page 9
- "Deleting an object type" on page 15

Related reference:

- "OBJECT" on page 81
- "Information Catalog Center predefined object types" on page 95
- "ACTION.OBJTYPE" on page 69
- "PROPERTY" on page 86

Deleting an object type

You can delete an object type by using the Information Catalog Center windows or tag language. Make sure that you select the correct object type to delete because all objects of that type will be deleted also.

Prerequisites:

You must have defined object types in your information catalog.

Restrictions:

You cannot delete the predefined Information Catalog Center object types Comments or Programs.

Procedure:

To delete an object type using the Information Catalog Center windows:

1. Optional: Search for objects of the object type you want to delete to make sure that you do not want to keep any of them.
2. Expand the **Administration** folder in the main Information Catalog window.
3. Click the **Object types** folder and list the object types.
4. Right-click the icon of the object type that you want to delete.
5. Click **Delete**.
   The Confirm Delete window opens.

To delete an object type using the Information Catalog Center tag language:

Type the following lines in your tag language file:

```
:ACTION.OBJTYPE(DELETE_EXT)
:OBJECT.TYPE(object_type_short_name)
```

After each keyword, type an appropriate value within the parentheses:

Keyword
Value

TYPE The short name of the object type you are deleting.

Related tasks:

- "Defining an object type" on page 9
- "Updating an object type" on page 12

Related reference:

- "OBJECT" on page 81
- "Information Catalog Center predefined object types" on page 95
- "ACTION.OBJTYPE" on page 69
- "PROPERTY" on page 86

Chapter 3. Populating the catalog with information

After you define the object types for your information catalog, you populate, or fill, the information catalog with objects.

Objects

An object is an item that represents a unit or distinct grouping of information. Every object is associated with an object type. For example, an object type ″Image″ might include an object that is called My_DBA, which describes a bitmap photograph of the database administrator.

You define objects of various types to represent the actual information available in your organization. The properties an object can have are determined by its object type. The values that you assign to the properties that are defined as the Unique Identifier for this object type makes it unique. No two objects, based on the same type, can have the same values for the properties of its Unique ID.

In the Information Catalog Center, you can add relationships to an object while you define it. The object type determines the types of relationships the object can participate in.

Privileges are determined at the object level. When you define an object, you can specify which users and groups can have show, read, or write authority to the object. Anyone who has update authority can delete the object from the information catalog.

Related concepts:

- “Object types” on page 7
- “Unique identifiers” on page 8
- “Relationships” on page 28

Related tasks:

- “Defining an object type” on page 9

Copying an object

You can create a new object that has the values of an existing object. Copying an object makes it easier to define a new object that will be similar to an existing object.

Prerequisites:

You must have read access or higher access to the object that you want to copy.

Steps:

To copy an object:

1. Find the object that you want to copy in the Information Catalog Center main window.

2. Right-click on the object that you want to copy.
3. Click **Copy**.

   The Copy Object notebook opens.
4. On the Properties / Values page, change any of the properties to make the new object.

   You must change at least one unique identifier value for your new object to be unique.
5. Optional: On the Relationships page, add relationships that you want the new object to have. Remove any relationships in which you do not want the new object to participate.
6. Optional: On the Privileges page, change the privileges for the new object.
7. Click **OK**. The new object is created.

Related concepts:

• "Objects" on page 17

Defining an object

You can define an object using the Information Catalog Center windows or tag language. (You can create many objects at the same time by using the Information Catalog Center tag language.) Objects that you define in the Information Catalog Center are the metadata that represent real business objects. When you define an object, you assign values to a list of properties (which is determined by the object type on which this object is based). The values of certain properties combine to become the unique identifier for the object. While you are defining an object, you assign access privileges to the object. You can also create relationships between the object and other existing objects.

Prerequisites:

You must create an information catalog. You might need to define a new object type if the predefined object types do not suit your needs.

You must have administrator access to the information catalog or be designated as a power user to the information catalog.

Procedure:

To define an object using the Information Catalog Center windows:

1. Expand the **All objects by type** folder in the Information Catalog Center Catalog window.
2. Right-click the icon of the object type for which you want to create an object.
3. Click **Define**.

   The Define Object notebook opens.
4. Enter the information about the object you are creating and click **OK**.

   The object is created.

To define an object using the Information Catalog Center tag language:

1. Type the following lines in your tag language file, using as many `name(value_for_property)` lines as necessary to identify all the object type properties.

```
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(object_type_short_name)
:INSTANCE.short_name(value_for_property)
    short_name(value_for_property)
    short_name(value_for_property)
```

Note: You can use the same tag language file that you used to define an object type. Make sure that the tags for defining an object follow the object type definition. The properties can be typed in any order, and you can omit properties for which you do not have a value if the property is not required.

2. After each keyword, type an appropriate value within the parentheses:

   Keyword
   Value

   TYPE The short name of the object type for which you are creating an object.

   short_name
   The short name of the object type property.

3. For each object, type the name of each object type property, followed by a value for the property in parentheses. Your tag language file should look similar to the following example:

```
:COMMENT.-----------------------------------------------------
:COMMENT. Creating objects of object type
:COMMENT. "Relational tables and views"
:COMMENT.-----------------------------------------------------
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(TABLES)
:INSTANCE.NAME(Customer)
    SHRTDESC(Customer information table)
    LONGDESC(Customer number, name,
        CelDial rep, customer contact information.)
    ACTIONS(Click on 'Start Program...' to invoke Visualizer TableViewer.)
    REMARKS(DB2 table) DBNAME(DGWDATA) OWNER(USERID) TABLE(CUSTOMER)
    URL(http://webserver/cgi-bin/tableviewer)
    SOURCE(DB2 SYSTEM CATALOGS)
```

Related concepts:

- "Object definition for the Data Warehouse Center" in the *Data Warehouse Center Application Integration Guide*
- "Objects" on page 17

Related reference:

- "Default properties for all Information Catalog Center objects" in the *Data Warehouse Center Application Integration Guide*
- "OBJECT" on page 81
- "INSTANCE" on page 76
- "ACTION.OBJINST" on page 65

Updating an object

You can update an object by using the Information Catalog Center windows or tag language. You can update an object's name, description, properties, relationships, and privileges.

Prerequisites:

You must define objects in your information catalog.

If you have user privileges to the information catalog, you can update objects that you defined or that an administrator has given you privileges to update.

Procedure:

To update an object using Information Catalog Center windows:

1. Find the object you want to update in the Information Catalog Center main window.
2. Right-click the object that you want to update.
3. Click **Properties**.

   The Properties notebook opens.
4. Enter updated information into the appropriate fields and click **OK**.

   The Properties notebook closes, and the object is updated.

To update an object using the Information Catalog Center tag language:

1. Type the following lines in your tag language file:

   ```
   :ACTION.OBJINST(UPDATE)
   :OBJECT.TYPE(object_type_short_name)
   ```
2. Type the following lines, specifying the unique identifier properties and property values of the object you want to change:

   ```
   :INSTANCE.SOURCEKEY(UI_short_name(value_for_property)
        UI_short_name(value_for_property)
        UI_short_name(value_for_property))
   ```

   After each keyword, type an appropriate value within the parentheses:

Keyword	Value
UI_short_name	The name of a unique identifier property of the object type.

   Properties and values that are specified after the SOURCEKEY keyword are the unique identifier. When you created the object type, you defined up to sixteen properties in a certain order to make up the unique identifier. When you type those properties and values, the Information Catalog Center checks the values in the order that is defined in the object type to locate a particular object.

   Completely enclose in parentheses all the properties and values after the SOURCEKEY keyword.
3. Type the name of each object property that you want to update, followed by the new value in parentheses.

   ```
   short_name(new_value_for_property)
   ```

   You do not need to include all the object's properties. Any properties that you omit will not be updated.

The following is an example of tag language to update an object. In this example, the value in SHRTDESC is updated.

```
:ACTION.OBJINST(UPDATE)
:OBJECT.TYPE(TABLES)
:INSTANCE.SOURCEKEY(DBNAME(DGWDATA) OWNER(USERID)
     TABLE(CUSTOMER))
     SHRTDESC(Mobile phone customer information table)
```

Related concepts:

- "Objects" on page 17

Related reference:

- "OBJECT" on page 81
- "INSTANCE" on page 76
- "ACTION.OBJINST" on page 65

Deleting an object

You can delete an object by using the Information Catalog Center windows or tag language. When you delete an object, it is deleted from the Information Catalog Center and any comments to the object and any other relationships the object participated in are removed.

Prerequisites:

You must define objects in your information catalog.

You can delete any objects you have authority to update.

Procedure:

To delete an object using the Information Catalog Center windows:

1. Find the object that you want to delete in the Information Catalog Center main window.
2. Right-click on the object that you want to delete.
3. Click **Delete**. The Confirm Delete window opens.

To delete an object using the Information Catalog Center tag language:

1. Type the following line in your tag language file:

   ```
   :ACTION.OBJINST(DELETE)
   ```

2. Type the following line, specifying the object type of the object you want to delete:

   ```
   :OBJECT.TYPE(object_type_short_name)
   ```

3. Enter the following lines, specifying the unique identifier properties and property values of the object you want to delete:

   ```
   :INSTANCE.SOURCEKEY(UI_short_name(value_for_property)
        UI_short_name(value_for_property)
        UI_short_name(value_for_property))
   ```

   After each keyword, type an appropriate value within the parentheses:

Keyword	Value
TYPE	The name of the object type for which you are deleting an object.
UI_short_name	The name of a unique identifier property that belongs to the object type for which you are deleting an object.

   Properties and values that are specified after the SOURCEKEY keyword are the unique identifier. When you created the object type, you defined certain properties in a certain order to make up the unique identifier. When you enter those properties and values, the Information Catalog Center checks the values in the order that is defined in the object type to locate an object.

   Completely enclose in parentheses all the properties and values after the SOURCEKEY keyword.

The following is an example of a tag language file to delete an object.

```
:ACTION.OBJINST(DELETE)
:OBJECT.TYPE(TABLES)
:INSTANCE.SOURCEKEY(DBNAME(DGWDATA) OWNER(USERID)
     TABLE(CUSTOMER))
```

In this example, the table object identified by `DBNAME(DGWDATA) OWNER(USERID) TABLE(CUSTOMER)` is deleted.

Related concepts:

- "Objects" on page 17

Related reference:

- "OBJECT" on page 81
- "INSTANCE" on page 76
- "ACTION.OBJINST" on page 65

Chapter 4. Making the information catalog convenient for users

This chapter describes some of the ways that you can make the information catalog more convenient for users:

- Adding relationships between objects
- Adding comments to objects
- Creating subject areas
- Associating programs with object types

Relationship types

Whenever you want to form a relationship between two objects, you must use a relationship type. A relationship type defines the roles that an object can play in a relationship. For example, if you placed a contains relationship type between two objects, that creates a hierarchical relationship between the two objects. One object would have to play a child role, and the other object would have to play a parent role.

The Information Catalog Center contains a set of predefined relationship types that are ready for you to use in your organization. These relationship types are already associated with the predefined object types in Information Catalog Center.

Each relationship type is based on a category, which determines the roles that object types can play in it. You can create your own relationship types, but you must select a predefined category which will determine the roles that are used within each new relationship type.

There are four predefined relationship categories:

hierarchical
: Relationship types that are used to connect objects that have a hierarchical relationship.

peer to peer
: Relationship types that are used to connect object that have a peer relationship.

support
: Relationship types that connect supporting objects to another object (for example, you can connect a News object to a Spreadsheet object).

precedence
: Relationship types that connect precedence objects to data resources (for example, you can connect a Precedence object to a File object). Objects that are connected with this category of relationship are displayed in the Information Catalog Center Show Lineage Tree window.

There are eight predefined relationship types:

Attachment
: Attaches objects to other objects. Comment objects can only be attached to other objects as a support object.

Cascade
Connects two precedence objects.

Contact
Identifies a reference for more information about an object. More information might include the person who created the information that the object represents or the department responsible for maintaining the information.

Contains
Identifies Information Catalog Center objects that contain other objects. For example, use contains to denote objects in a hierarchical relationship, where one object is the parent and the other is the child.

Dictionary
Associates a glossary entry object type with another object. You can use a glossary entry object type to define terminology that is associated with the object.

Input Connects objects that transform to their input data resource.

Linked
Connects two or more objects in an information catalog. Objects in a linked relationship are peers, rather than a hierarchical relationship.

Output
Connects objects that transform to their output data resource.

Supported
Provides additional information about your information catalog or enterprise.

The following table summarizes the relationship types and their related categories and roles.

Table 1. Relationship types, categories, and roles

Relationship type	Relationship category	Relationship roles
Attachment	Support	Object, support object (only for comment relationships)
Cascade	Precedence	Preceding object, succeeding object
Contact		
Contains	Hierarchical	Parent, child
Dictionary		
Linked	Peer to peer	Object
Input	Precedence	Preceding object, succeeding object
Output	Precedence	Preceding object, succeeding object
Supported		

Related tasks:
- "Defining a relationship type" on page 25
- "Deleting a relationship type" on page 28
- "Updating a relationship type" on page 26

Related reference:

- "Mapping Version 7 object type categories to Version 8 relationship types, categories, and roles" on page 123
- "Predefined relationship type models" on page 100

Defining a relationship type

When you define a relationship type, you determine the relationship roles that an object can play in a relationship of that type. Some relationship types allow you to select more than one role per category. Relationship types are assigned to object types. Each object type has a list of relationship types and roles that they can use when building a relationship. The roles available are determined by what category you select. When you create an object, the object type it is based on has the list of relationship types that you can use when you are actually adding your relationship.

Prerequisites:

You must create an information catalog.

You must have administrator privileges to the information catalog.

Procedure:

To define a relationship type using the Define Relationship Type window:

1. Expand the **Administration** folder in the main Information Catalog window.
2. Right-click the **Relationship Types** folder.
3. Click **Define**.

   The Define Relationship Type notebook opens.
4. Type a name for the relationship type in the **Name** field.
5. Select the category that you want the relationship type to be a part of in the **Category** field.
6. On the **Object type constraints** page, select the object type and role that you want to associate with the relationship type.
7. Click **>** to move your selection to the Selected object types list.

   The constraints that you select determine the possibilities of behavior between any objects to which you assign this relationship .
8. Click **OK** to define the relationship type.

To define a relationship type using the Information Catalog Center tag language:

1. Type the following lines in your tag language file:

```
:ACTION.RELTYPE(ADD)
:RELATIONTYPE.TYPE(relationship_type_short_name)
     CATEGORY(relationship_type_category)
     EXTNAME(relationship_type_name)
     DESCRIPTION(relationship_type_description)
```

   After each keyword, type an appropriate value within the parentheses:

   Keyword
   Value

   TYPE The short name of the relationship type. The rules for a relationship type short name are:
   - Case-sensitive.

- 18 character (SBCS) maximum.
- First character must be uppercase or lowercase English alphabetic, @, or #
- Subsequent characters must be uppercase or lowercase English alphanumeric, @, # or _. No $, leading or embedded blanks.
- It must be unique to the information catalog.

CATEGORY
Use `Support`, `Hierarchical`, `Precedence` , or `Peer to Peer`.

EXTNAME
An extended, descriptive name of the relationship type. The rules for the relationship type name are:
- 200 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

DESCRIPTION
A description of the relationship type. The rules for the relationship type description are:
- 254 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

Related concepts:
- "Relationship types" on page 23

Related tasks:
- "Deleting a relationship type" on page 28
- "Updating a relationship type" on page 26

Related reference:
- "ACTION.RELTYPE" on page 74
- "RELATIONTYPE" on page 89

Updating a relationship type

You can change some of the properties of a relationship type.

Prerequisites:

You must create a relationship type.

You must have administrator privileges to the information catalog.

Procedure:

To update a relationship type using the Relationship Type Properties window:
1. Expand the **Administration** folder in the main Information Catalog window.
2. Expand the **Relationship Types** folder.
3. Right-click the relationship that you want to update.
4. Click **Properties**.

The Relationship Type Properties window opens.

5. Change the name or description, or click the **Object Type Constraints** page to add object types.
6. Click **OK** to update the relationship type.

To update a relationship type using the Information Catalog Center tag language:

1. Type the following lines in your tag language file:

```
:ACTION.RELTYPE(ADD)
:RELATIONTYPE.TYPE(relationship_type_short_name)
```

2. To change the relationship type name, add the following line:

```
EXTNAME(new_relationship_type_name)
```

3. To change the relationship type description, add the following line:

```
DESCRIPTION(new_relationship_type_description)
```

After each keyword, type an appropriate value within the parentheses:

Keyword
Value

TYPE The short name of the relationship type.

CATEGORY
Use `Support`, `Hierarchical`, `Precedence`, or `Peer to Peer`.

EXTNAME
The name of the relationship type. The rules for the relationship type name are:

- 200 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

DESCRIPTION
A description of the relationship type. The rules for the relationship type description are:

- 254 character maximum.
- It must not contain null characters.
- It must not be all blank characters.

Related concepts:

- "Relationship types" on page 23

Related tasks:

- "Defining a relationship type" on page 25
- "Deleting a relationship type" on page 28

Related reference:

- "ACTION.RELTYPE" on page 74
- "RELATIONTYPE" on page 89

Deleting a relationship type

When you delete a relationship type, any relationships that were already created based on that relationship type are deleted.

Prerequisites:

You must create a relationship type.

You must have administrator privileges to the information catalog.

Procedure:

To delete a relationship type using the Information Catalog Center windows:

1. Expand the **Administration** folder in the main Information Catalog window.
2. Expand the **Relationship Types** folder.
3. Right-click the relationship that you want to delete.
4. Click **Delete**.
   The Confirm Delete window opens.
5. Click **OK** to delete the relationship type.
   Note: Deleting a relationship type also deletes all relationships of that particular type.

To delete an relationship type using the Information Catalog Center tag language

1. Type the following lines in your tag language file:
   ```
   :ACTION.RELTYPE(DELETE)
   :RELATIONTYPE.TYPE(relationship_type_short_name)
   ```
   After each keyword, type an appropriate value within the parentheses:

Keyword	Value
TYPE	The short name of the relationship type.

Related concepts:

- "Relationship types" on page 23

Related tasks:

- "Defining a relationship type" on page 25
- "Updating a relationship type" on page 26

Related reference:

- "ACTION.RELTYPE" on page 74
- "RELATIONTYPE" on page 89

Relationships

You can establish a relationship between objects. The relationship role determines the behavior that the objects in the relationship participate in. For example, in the support relationship, if the ″object″ role object is deleted, then the ″support″ role object and the relationship instance between the two are deleted as well. However, if the ″support″ role object is deleted, then the relationship instance is deleted but

the ″object″ role object remains. All relationships are established while defining or updating an object. The exception is the relationship created between an object and its comment. When you create a comment, the relationship is automatically created for you.

Contacts, glossary entries, and supports

To create contacts, glossary entries, and supports, you begin by defining objects from specific object types. Then you use the Add Relationship window to create the relationship. You must have specified that you want pre-defined object types in the information catalog when you initialized it. The following list explains which object type and relationship type to choose in order to create these relationships:

- Contacts—Define an object based on the People to contact object type. Give the contact information in this object. When you add a relationship, select the relationship type ″Contact (object)″ from the **Relationship type (role)** list.
- Glossary entries—Define an object based on the Glossary entries object type. Give the term and definition in this object. When you add a relationship, select the relationship type ″Dictionary (object)″ from the **Relationship type (role)** list.
- Support—Define an object based on the News object type. Give the support information in this information. When you add a relationship, select the relationship type ″Supported (object)″ from the **Relationship type (role)** list.

Note: Glossary entries and supports replace the Information Catalog Manager Version 7 Dictionary facilities and Support facilities.

Related concepts:

- “Relationship types” on page 23

Related tasks:

- “Adding a relationship between objects” on page 29
- “Removing a relationship between objects” on page 30

Related reference:

- “Predefined relationship type models” on page 100

Adding a relationship between objects

You can add relationships between two objects based on the object type constraints and the roles for the relationship type. The types of relationships you can add between two objects is specified at the object type level. When you add a relationship, you specify the relationship type and roles that the two objects will play in the relationship.

Prerequisites:

You must define objects. The objects should also be able to play the roles of the relationship you are adding.

Restrictions:

You must have administrator or power user privileges. If you are a user, you must have write authority to the objects that you want to add a relationship between.

Procedure:

To add a relationship between objects:

1. Find the objects that you want to add a relationship between in the main Information Catalog Center window.
2. Right-click one of the objects.
3. Click **Properties**.

   The Object Properties notebook opens.
4. Click the **Relationships** tab.

   The Relationships page opens.
5. Click **Add**.

   The Add Relationship window opens.
6. In the Available objects list, select the object that you want to participate in the relationship. (You can select more than one object.)
7. Click the > button.

   The object you selected moves to the Selected objects list.
8. Select a relationship type and role from the **Relationship type (role)** list.
9. Click **Apply**.

   The relationship is added. The Add Relationship window remains open. You can continue to add relationships.
10. Click **OK** when you are finished adding relationships.

    The Add Relationship window closes. The relationships page of the object is updated.
11. Click **OK**.

    The Object Properties window closes.

Related concepts:

- "Relationships" on page 28
- "Relationship types" on page 23

Related tasks:

- "Removing a relationship between objects" on page 30

Related reference:

- "ACTION.RELATION" on page 72

Removing a relationship between objects

You can remove the relationship between objects.

Prerequisites:

You must have administrator or power user privileges. If you are a user, you must have write authority to the objects that you want to remove a relationship from.

Procedure:

To remove a relationship between objects:

1. Find the objects that you want to remove the relationship from in the main Information Catalog Center window.
2. Right-click one of the objects.

3. Click **Properties**.
   The Object Properties notebook opens.
4. Click the **Relationships** tab.
   The Relationships page opens.
5. Select the relationship that you want to remove.
6. Click **Remove**.
7. Click **OK**.
   The relationship is removed.

Related concepts:

- "Relationships" on page 28
- "Relationship types" on page 23

Related tasks:

- "Adding a relationship between objects" on page 29

Related reference:

- "ACTION.RELATION" on page 72

Comments

Use comments to provide additional information about an object. A relationship is automatically placed between the comment and the object it is annotating.

Creating a comment

You can attach a comment to any object in the information catalog except another comment object.

Procedure:

To create a comment:

1. Right-click the object that you want to attach a comment to.
2. Click **Create comment**.
   The Create Comment notebook opens.
3. Type a name for the comment in the Name field.
4. Optional: Specify other information for your comment.
5. Optional: On the Privileges page, specify the users and groups you want to have access to this comment.
6. Click **OK** to create the comment and attach it to the specified object.

Related concepts:

- "Relationships" on page 28

Related tasks:

- "Deleting a comment" on page 32
- "Updating a comment" on page 32

Updating a comment

You can update a comment's properties and privileges.

Prerequisites:

You must have already created the comment and the object the comment annotates.

You must have administrator privileges to the information catalog or be the creator of the comment.

Steps:

To update a comment:

1. Right-click the comment you want to update.
2. Click **Properties**.
   The Update Comment window opens.
3. Change at least one of the following values:
   - Name
   - Actions
   - Status
   - Description
4. To change privileges, click the **Privileges** page and update the desired fields.
5. Click **OK**.

Related concepts:

- "Relationships" on page 28

Related tasks:

- "Creating a comment" on page 31
- "Deleting a comment" on page 32

Deleting a comment

When you delete a comment from the information catalog, both the comment and the relationship to the object it annotates are deleted.

Prerequisites:

You must have created a comment and the object it annotates.

Restrictions:

You must have administrator privileges to the information catalog. If you have user privileges, you must be the user that created the comment, or have update authority on the comment.

Steps:

To delete a comment:

1. Right-click the comment you want to delete.

2. Click **Delete**.
   The Confirm Delete window opens.
3. Make sure that the comment listed is the one you want to delete.
4. Click **Delete** to delete the comment.
   The comment is deleted from the information catalog.

Related concepts:

- "Relationships" on page 28

Related tasks:

- "Creating a comment" on page 31
- "Updating a comment" on page 32

Creating a subject area

You can designate instances of an object type as potential subject area objects. You can designate an object type as a possible subject area while defining or updating an existing object type. An object will only appear at the top level in the Subjects folder in the main Information Catalog Center window if the following conditions are met:

- The object is based on an object type that was selected to be a subject area.
- The object must have a Contains-parent hierarchal relationship.
- The object must not have any parents, but it must be allowed to contain children.
- The object must not be contained in another subject area.

Prerequisites:

You must define an object type that will also be a subject area.

Restrictions:

You must have administrator privileges to the information catalog.

Procedure:

To create a subject area while defining an object type, follow the instructions for defining an object type. Make sure to select the **Make a subject area** checkbox on the **Relationships** page of the Define Object Type window.

To create a subject area in an existing object type:

1. Expand the **Administration** folder in the main Information Catalog Center window.
2. Expand the **Object Types** folder.
3. Right-click the object type that you want to make a subject area.
4. Click **Properties**.
   The Object Type Properties window opens.
5. Click the **Relationships** tab.
   The Relationships page opens.
6. Select the **Make a subject area** box.

7. Click **OK**.

   The object type becomes a subject area, and the Object Type Properties window closes.

Related tasks:

- "Defining an object type" on page 9
- "Defining an object type" on page 9

Associating a program with an object type

You can associate a program with an object type when you define an object type or when you update an object type. Associating a program with an object type allows users to open objects of that type with the associated program.

Restrictions:

You must have administrator privileges in the Information Catalog Center to perform this task.

Procedure:

To associate a program with an object type while defining an object type, follow the instructions for defining an object type (LINK HERE).

To associate a program with an existing object type:

1. Expand the **Administration** folder in the main Information Catalog Center window.
2. Expand the **Object Types** folder.
3. Right-click the object type that you want to associate a program with.
4. Click **Properties**.

   The Object Type Properties window opens.
5. Click the **Programs** tab.

   The Programs page opens.
6. Specify the program name, platform, executable, parameter list, and description in the table.
7. Click **OK**.

   The program is associated with the object type, and the Object Type Properties window closes.

Related tasks:

- "Defining an object type" on page 9

Related reference:

- "Predefined program objects" on page 98

Chapter 5. Expanding and automating your information catalog

The easiest way to fill your information catalog with descriptive data is to use exisiting descriptions of your organization's information. You can use an extract program to populate your your information catalog with exisisting data. You can also import a tag language file that contains descriptive data.

Extracting descriptive data

Extract

Many databases and desktop applications already contain valuable descriptive data about your organization's information. The Information Catalog Center allows you to extract these existing descriptions into your information catalog. Extracting descriptive data also makes updating or refreshing your information catalog easier.

Information Catalog Center provides a sample program that can extract descriptive data from any database that is accessible using the JDBC or ODBC interfaces. This program will import the data directly into the Information Catalog Center. If the program is unable to import the data, a tag language file is created that can be used to import the data into the Information Catalog Center. The sample extracter is installed with the ICM Samples component of the Information Catalog Center.

The source code for the sample program is also included. This allows you to write your own extract program or modify the existing code to fit your needs.

Using the extract programs and the Information Catalog Center tag language, you can perform the following tasks:
- Extract the descriptive data
- Change the descriptive data
- Add to the descriptive data (if necessary) to meet your work group's needs
- Import descriptive data into your information catalog

The JDBC extractor and source code files are located in the SQLLIB\SAMPLES\ICMJDBC directory. This directory also includes a README file that provides instructions for setting up the control file and using the command line interface.

You can edit the tag language files produced by extract programs by using any word processing program that can import and export ASCII text files.

Things to consider when writing a customized extract program

Keep the following considerations in mind when writing a customized extract program:
- When the Information Catalog Center imports a tag language file, it ignores any characters with a hexadecimal value less than X'20'.
- If your extract program generates tag language for values that contain parentheses, it must enclose the parentheses with single quotation marks. Otherwise, the parentheses are considered delimiters. For example, if the value

is a telephone number (800) 555-1234, then your extract program needs to represent this value as `PHONENUM('('800')' 555-1234)`.

- When your extract program generates values for the properties, the Information Catalog Center does not remove leading blanks. For example, if your program generates `TABNAME( EMPLOYEE)` instead of `TABNAME(EMPLOYEE)` for a property defined to contain only 8 bytes, the Information Catalog Center returns an error message because the value is 10 bytes long instead of the 8 bytes defined for the TABNAME property in the object type definition.
- If your output tag language file is long, use the :COMMIT tag at regular intervals in the file to commit periodic changes to the database.

Related tasks:

- "Creating object types and objects with a customized extract program" on page 38
- "Merging duplicate object types and objects with a customized extract program" on page 39
- "Extracting descriptive data with the Information Catalog Center extract program" on page 36

Related reference:

- "Valid tags in a tag language file created by a customized extract program" on page 37

Extracting descriptive data with the Information Catalog Center extract program

The Information Catalog Center comes with a sample program that can extract descriptive data from any JDBC or ODBC compliant database. The sample program and source code are installed with the Information Catalog Center in the SQLLIB\SAMPLES\ICMJDBC directory. A README file is also located in this directory to provide instructions for setting up the control file and using the command line interface.

Prerequisites:

The JDBC drivers for the source database must be installed on the system.

Restrictions:

You must be an administrator for the source database.

Procedure:

1. Identify descriptive data for extraction.

   The Information Catalog Center extract program for relational databases extracts descriptive data for Table and Column types of objects. The relational catalogs of the databases do not contain all the properties that these objects require. Therefore, the extract programs produce only the descriptive data for the properties that the catalogs contain.
2. Decide if you are going to use the sample extract program to automatically import the data.

   If you need to customize the data, you might prefer to use the sample extract program to generate a tag file. You can then edit the tag file as needed to fill in any missing information and import the tag file into the Information Catalog

Center. Be sure to read the README file in the SQLLIB/SAMPLES/ICMJDBC directory before using the sample extract program.

Related concepts:

- "Extract" on page 35

Related reference:

- "Tag language" on page 59

Extracting descriptive data with a customized extract program

This section covers some of the considerations you need to keep in mind when you write a customized program to extract descriptive data.

Valid tags in a tag language file created by a customized extract program

An output tag language file derived from a customized extract program can contain some or all of the following tags:

:ACTION.
Specifies that an action (add, update, delete, append, or merge) occurs involving an object type, object, or relationship

:OBJECT.
Identifies an object type and its properties

:PROPERTY.
Identifies a property for the object type that you are defining

:INSTANCE.
Identifies an object or relationship

:RELTYPE.
Identifies the type of relationship that you are adding or deleting

:COMMIT.
Identifies an î database commit point

:COMMENT.
Allows you to add comments to the tag language file

:NL. Allows you to include multi-line property values (for non-UI properties)

:TAB. Allows you to insert tabs in non-UI property values

Related concepts:

- "Extract" on page 35

Related tasks:

- "Creating object types and objects with a customized extract program" on page 38
- "Merging duplicate object types and objects with a customized extract program" on page 39

Related reference:

- "Tag language file content for the Information Catalog Center" on page 63
- "Tag language" on page 59
- "Rules for writing tag language files" on page 60

Creating object types and objects with a customized extract program

The Information Catalog Center provides an API that allows you to write a customized extract program. You can use the API directly to create user-defined object types, or you can write a program that generates a tag language file that can be imported into the Information Catalog Center.

Prerequisites:

To use the Information Catalog Center API, the JDBC drivers for the source database must be installed on the system.

Procedure:

To create user-defined object types from a customize extract program:

1. Use the API or tag language to define an object type.
2. Make sure that the tag language specifies at least one property that has the UUISEQ option with a value of 1.

   You can specify up to fifteen additional properties that have the UUISEQ option with a value of 2, 3, 4, or 5. UUISEQ specifies the position of the property in the unique identifier that uniquely identifies an object to an information catalog.

   For example, the database catalog of your database might describe several tables in the database. This catalog contains the following properties that you want to store in your information catalog:
   - 8-character identifier of the source
   - 10-character name of the table
   - 80-character variable-length description of the table
   - 8-character owner of table

   To produce an object type that contains these properties, your extract program must produce a tag language file that contains the following tags:

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE(MYTABLE)
        CATEGORY(GROUPING)
        EXTNAME(The tables on my data source)
:PROPERTY.EXTNAME(Data source name)
          DT(C) DL(8) SHRTNAME(DSNAME) UUISEQ(2) NULLS(N)
:PROPERTY.EXTNAME(Name of the table)
          DT(C) DL(10) SHRTNAME(TABNAME) UUISEQ(1) NULLS(N)
:PROPERTY.EXTNAME(Description of table)
          DT(V) DL(80) SHRTNAME(TABDESC) NULLS(Y)
:PROPERTY.EXTNAME(Owner of table)
          DT(V) DL(8) SHRTNAME(TABOWNER) NULLS(Y)
```

   If you are using the Information Catalog Center API instead of generating tag language, use the API to create an object type named MYTABLE with a list of properties. The display name (EXTNAME) is a required value.
3. Use the API or tag language to create object instances.

   For example, a database catalog might have descriptive data for three tables that you want to store in your information catalog. Your extract program can read the descriptive data for these three tables from your database catalog. You extract program then writes the tag language file to generate three objects of the MYTABLE object type created in step 2.

Suppose that your tables have the following properties:

Source name	Table name	Table description	Owner
MYDATA	EMPLOYEE	Personnel information about company employees	LONGO
MYDATA	SALES	Data about 2000 sales-to-date	VALDEZ
MYDATA	CUSTOMER	Shipping information about customers	MARSH

Your extract program must produce the tags that are shown below. You must insert these tags in the tag language file following the tags that define the object type.

```
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(MYTABLE)
:INSTANCE.NAME(Personnel information)
          DSNAME(MYDATA)
          TABNAME(EMPLOYEE)
          TABDESC(Personnel information about company employees)
          TABOWNER(LONGO)
:INSTANCE.NAME(Annual sales information)
          DSNAME(MYDATA)
          TABNAME(SALES)
          TABDESC(Data about 1997 sales-to-date)
          TABOWNER(VALDEZ)
:INSTANCE.NAME(Customer shipping information)
          DSNAME(MYDATA)
          TABNAME(CUSTOMER)
          TABDESC(Shipping information about customers)
          TABOWNER(MARSH)
```

If you are using the Information Catalog Center API instead of generating tag language, use the API to create each object instance, specifying the appropriate property values.

Related concepts:

- "Extract" on page 35

Related tasks:

- "Merging duplicate object types and objects with a customized extract program" on page 39
- "Extracting descriptive data with the Information Catalog Center extract program" on page 36

Related reference:

- "Valid tags in a tag language file created by a customized extract program" on page 37

Merging duplicate object types and objects with a customized extract program

The JDBC extract program that is shipped with the Information Catalog Center creates database objects, table objects and column objects. Refer to *The Data Warehouse Center: Application Integration Guide* for more information about the default object types and their property definitions. To modify the default object types, modify the source code for the sample extract program or create a tag file using the extract program. The tag file must be edited to add user defined object

types or modify the default object types. Once the tag file is complete it can be imported into the Information Catalog Center.

Prerequisites:

The JDBC drivers for the source database must be installed on the system.

Restrictions:

The following restrictions apply:

- You cannot merge the Programs or the Comments object types.
- User-defined object types must be created using the Information Catalog Manager API or the ACTION(MERGE) tag before defining object instances.
- Use the Information Catalog Manager API or the ACTION(APPEND) tag to add properties to the default object type definitions.

Procedure:

Use one of the following methods to create a customized extract program that merges duplicate objects and duplicate object types:

- Use the Information Catalog Manager API to write customized source code to create user defined object types or modify default object type definitions. For more information on using the Information Catalog Manager API, refer to the following website:

  `http://www.ibm.com/software/data/db2/warehouse/`
- Write customized source code to create a tag file to create user defined object types or modify default object type definitions.

Related concepts:

- "Extract" on page 35

Related tasks:

- "Creating object types and objects with a customized extract program" on page 38
- "Extracting descriptive data with the Information Catalog Center extract program" on page 36

Related reference:

- "Valid tags in a tag language file created by a customized extract program" on page 37

Importing tag language files

You can import Information Catalog Center tag language files that contain descriptive data into your information catalog.

Prerequisites:

Before you import a tag language file:

- You must have created an information catalog
- You must have created a tag language file

Restrictions:

You must be a database administrator for DB2 Universal Database.

Procedure:

To import a tag language file into your information catalog, start from your Information Catalog window.

1. Select your information catalog in the main Information Catalog window.
2. Click **Selected** –> **Import**.

   The Import window opens.
3. Type the directory path and file name of the tag language file you want to import in the **Import file** field.
4. Type the directory path from which you want to import icon files in the **Icon path** field.
5. Optional: Type the name of the destination file to which import messages will be written in the **Log file** field.

   If you do not specify a log file, the default destination file is the file name you specified in the **Import file** field with a .log extension instead of the original file extension. By default, the log file is placed in the same directory as the import file.
6. Indicate the appropriate radio button to indicate where you want to begin importing the tag language file from.
   - Click **Beginning of the file** to start importing from the beginning of the file.
   - Click **Checkpoint in the file** to start importing from the last point at which the Information Catalog Center successfully committed changes to the information catalog.
7. Click **OK** to begin importing the specified tag language file.

   The Import window remains open with a progress indicator. A message indicates when import is complete.

   To close the window without importing a tag language file, click **Cancel**.

Related tasks:

- "Importing tag language files from the command line" on page 120
- "Exporting tag language files" on page 42
- "Logging deletions from your information catalog" on page 47
- "Reading the import log file" on page 47

Related reference:

- "Metadata import from a tag language file" in the *Data Warehouse Center Application Integration Guide*
- "Metadata export from the Information Catalog Manager" in the *Data Warehouse Center Application Integration Guide*
- "Tag language" on page 59

Exporting tag language files

You can export Information Catalog Center tag language files that contain descriptive data to other applications.

Prerequisites:

Before you export a tag language file:

- You must have created an information catalog
- You must have created a tag language file

Restrictions:

You must be a database administrator for DB2 Universal Database.

Procedure:

To export a tag language file from your information catalog, start from your Information Catalog window.

1. Select your information catalog in the main Information Catalog window.
2. Click **Selected** –> **Export**.

   The Export notebook opens.
3. Type the directory path and file name of the tag language file you want to export to in the **Export file** field. The existing file will be overwritten. The directory path must exist.

   All available objects that can be exported are shown by object type in the **Available Objects** list. Select the objects that you want to export by highlighting them in the **Available Objects** list and clicking the **>** button. The selected objects are moved to the **Selected Objects** list.
4. Click the **Options** tab.

   Select the **Export BLOB/CLOB** checkbox to include BLOB and CLOB with the selected objects. You must enter a valid path in the **Path** field.

   Select the **Export icon file** checkbox to include an icon file with the export. You must enter a valid path to which the icon file can be exported in the **Path** field.

   Select which relationship types you want to export with the selected objects by clicking the appropriate checkboxes in the **Export all related objects** field.

   Enter a valid log file path and file name in the **Log file** field to specify where to output the export log.

   Check the **Export in Information Catalog Manager Version 7 format** checkbox to export the selected objects to a Version 7 tag language format.
5. Click **OK**. The selected objects are exported from the information catalog to the selected export file.

Related tasks:

- "Importing tag language files" on page 40

Related reference:

- "Metadata import from a tag language file" in the *Data Warehouse Center Application Integration Guide*
- "Metadata export from the Information Catalog Manager" in the *Data Warehouse Center Application Integration Guide*

- "Tag language file content for the Information Catalog Center" on page 63
- "Tag language" on page 59
- "How the Information Catalog Center reads tag language files" on page 61

More on tag language files

Information Catalog Center import behavior for missing or empty values

A message is written to the log file if you import tag language to your information catalog that does not contain a complete unique ID for an object instance in a relationship. Table 2 shows the default unique IDs for RECORD and COLUMN object types. You can modify the object types so they have different unique IDs.

Table 2. Unique IDs for default RECORD and COLUMN object types

RECORD	COLUMN
SERVER	DBNAME
DBNAME	OWNER
OWNER	TABLES
RECNAME	COLUMNS
	FILENAME

You can import DB2 Version 8 tag files that have empty string values for object instance properties, even if the empty property is part of the unique ID. Use the following tag language to create an instance with the values of the OWNER and the URL properties set to the empty string. The ADD action creates the instance if it does not exist and issues an error message if another instance exists with the same unique ID values. The MERGE action creates the instance if it does not exist. If the instance exists, the MERGE action updates properties that are not part of the unique ID with values specified in the tag file.

```
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(RECORD)
:INSTANCE.NAME(<01> - CCD-REC)
SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER()
RECNAME(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
TYPE(RECORD) URL()
:COMMIT.CHKPID(30)
```

If a required property is missing from a DB2 Version 8 tag file, a default value is assigned for the property when the instance is created. The following tag code creates a RECORD object instance with a default value of "-" (dash) assigned to the missing OWNER property, where OWNER is part of the unique ID for the RECORD object type. Any missing properties such as SHRTDESC and URL that are not part of the unique ID and are not required are assigned null values in the database.

```
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
```

```
:OBJECT.TYPE(RECORD)
:INSTANCE.NAME(<01> - CCD-REC)
SERVER(stl11ffh)
DBNAME(Cobol Files)
RECNAME(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
TYPE(RECORD)
:COMMIT.CHKPID(40)
```

After an instance is created, the values of the unique ID properties cannot be modified. Other property values can be modified using either UPDATE or MERGE. If unique ID properties are missing from a MERGE or UPDATE, default values are substituted for the missing properties when searching for the instance. The UPDATE action searches for the specified unique ID and replaces property values for the specified instance. If the instance does not exist, an error message is issued. The MERGE action replaces property values if the specified instance exists. If the instance does not exist, the MERGE action creates the instance. In the following example, a default value is substituted for the missing unique ID property OWNER. If an instance is found, the value of the URL property is set to the empty string. Properties that are not specified are not be modified.

```
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(UPDATE)
:OBJECT.TYPE(RECORD)
:INSTANCE.SOURCEKEY(DBNAME(Cobol Files)
SERVER(stl11ffh)
RECNAME(CCD-REC))
URL()
:COMMIT.CHKPID(80)
```

When you create a relationship between two instances, the unique ID properties in the SOURCEKEY and TARGETKEY lists must match the values that were used when the instance was created. In the following example, the relationship is not created because the COLUMNS property is missing from the instance definition of the COLUMN object. A default value of "-" (dash) is assigned for the missing property value when the instance is created. The relationship definition specifies an empty string in the TARGETKEY list for the COLUMNS property. The empty string is considered a value, which results in an *Object Instance not found* message in the log file and the relationship is not created.

```
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(RECORD)
:INSTANCE.NAME(<01> - CCD-REC)
SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER(labriejj)
RECNAME(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
TYPE(RECORD)
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.NAME(<> CCD-REC.)
POSNO(35)
SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER(labriejj)
```

```
TABLES(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
:COMMENT.============================================================
:COMMENT.RELATIONSHIP
:COMMENT.============================================================
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(CONTAIN)
SOURCETYPE(RECORD)
TARGETYPE(COLUMN)
:INSTANCE.SOURCEKEY(SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER(labriejj)
RECNAME(CCD-REC))
TARGETKEY(DBNAME(Cobol Files)
OWNER(labriejj)
TABLES(CCD-REC) COLUMNS() )
:COMMIT.CHKPID(50)
```

To fix the relationship described above, add the COLUMNS() attribute to the object instance definition for the COLUMN object. Because the COLUMNS attribute is identical in the object instance definition and the TARGETKEY list, the relationship is created successfully:

```
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.NAME(<> CCD-REC.)
POSNO(35)
SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER(labriejj)
TABLES(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
COLUMNS()
:COMMIT.CHKPID(60)
```

In the following example, the first instance is missing the COLUMNS and FILENAME attributes. The instance is created with a default value for both attributes. The relationship TARGETKEY list is also missing the COLUMNS and FILENAME attributes. Default values are substituted for the missing unique ID properties when the relationship is created. Therefore, the relationship is created using the <>CCD-REC. COLUMN instance.

```
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(RECORD)
:INSTANCE.NAME(<01> - CCD-REC)
SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER(labriejj)
RECNAME(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
TYPE(RECORD)
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.NAME(<> CCD-REC.)
POSNO(20)
SERVER(stl11ffh)
DBNAME(Cobol Files)
```

```
OWNER(labriejj)
TABLES(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.NAME(AM00-ACCOUNT-KEY)
POSNO(35)
SERVER(VWNT90)
DBNAME(Cobol Files)
OWNER(labriejj)
TABLES(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
COLUMNS(AM00-CURR-BAL-FIN-CHG)
:COMMENT.============================================================
:COMMENT.RELATIONSHIP
:COMMENT.============================================================
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(CONTAIN)
SOURCETYPE(RECORD)
TARGETYPE(COLUMN)
:INSTANCE.SOURCEKEY(SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER(labriejj)
RECNAME(CCD-REC))
TARGETKEY(DBNAME(Cobol Files)
OWNER(labriejj)
TABLES(CCD-REC))
:COMMIT.CHKPID(70)
```

When you delete an object instance, the same substitution rules apply. If the SOURCEKEY list is missing one or more unique ID properties, default values are assigned for the missing properties. In the following example, the SOURCEKEY list for the DELETE is missing both the FILENAME and COLUMNS properties. A default value is assigned for both properties, and the <>CCD-REC. object instance is deleted.

```
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.NAME(<> CCD-REC.)
POSNO(20)
SERVER(stl11ffh)
DBNAME(Cobol Files)
OWNER(labriejj)
TABLES(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
:COMMENT.============================================================
:COMMENT.OBJECT INSTANCE
:COMMENT.============================================================
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.NAME(AM00-ACCOUNT-KEY)
POSNO(35)
SERVER(VWNT90)
DBNAME(Cobol Files)
OWNER(labriejj)
TABLES(CCD-REC)
CRTTIME(2000-06-16.11.44.10.440000)
COLUMNS(AM00-CURR-BAL-FIN-CHG)
:COMMENT.============================================================
:COMMENT.DELETE OBJECT INSTANCE
```

```
:COMMENT.============================================================
:ACTION.OBJINST(DELETE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.SOURCEKEY(DBNAME(Cobol Files)
OWNER(labriejj)
TABLES(CCD-REC))
:COMMIT.CHKPID(A1)
```

Logging deletions from your information catalog

You can keep a log of objects, object types, relationships, or relationship types that are deleted from your information catalog. You can transfer the log to a tag language file and use it to duplicate the deletions in other information catalogs, for example, to shadow information catalogs in a distributed environment.

To protect against erroneous deletions in other information catalogs, examine the contents of a delete history tag language file before importing it to any other information catalog.

Restrictions:

You must be a database administrator for DB2 Universal Database.

Steps:

To work with the delete history, start from your Information Catalog window.

1. Select your information catalog.
2. Click **Selected** → **History of Deleted objects** → **Start Recording**
3. Optional: To stop recording deletions, click **Selected** → **History of Deleted objects** → **Stop recording**
4. Optional: To copy the existing log of deletions to a tag language file:
   a. Click **Selected** → **History of Deleted objects** → **Transfer to file...**
      The Find File window opens
   b. Specify the directory path and name of a new or existing file to which you want to copy the log. Any information in an existing file is overwritten.
5. Optional: To erase the current log of deletions, click **Selected** → **History of Deleted** objects → **Clear recorded history**.

Related tasks:

- "Importing tag language files" on page 40
- "Reading the import log file" on page 47

Reading the import log file

The import log file includes the times and dates when the import process started and stopped. It also includes any error messages for problems that occurred during the process. By default, the import log file is placed in the same directory as the imported tag file. Unless you specified a new name during import, the name of the log file is the name of your tag language file plus an extension of .LOG.

Prerequisites:

You must have imported a tag language file into your information catalog.

Steps:

To read the import log file, locate the log file and open it. Check the log file for any errors.

48 shows an example of a log file.

```
START Import XML data to ICM database:
   Database name = ICMDB
   Catalog name  = ICM
   User ID       = db2admin
   Start Time    = Thu Nov 29 12:23:08 CST 2001
ADD object type = ZDATABASE
Generated checkpoint 20
ICMTAG-0515 Update object type = ZDATABASE.ICMTAG-0501
   Create object type = ZDATABASE.ICMTAG-88510 ICMSQLException.
```

In this example, the Information Catalog Center could not create the object type ZDATABASE. Refer to the DB2 Universal Database Message Reference for explanations of error messages.

Related tasks:

- "Importing tag language files" on page 40
- "Logging deletions from your information catalog" on page 47

Chapter 6. Exchanging metadata with other products

You can publish and exchange metadata with other IBM and non-IBM products. This chapter describes the publication and exchange metadata processes for the following products:

- The Data Warehouse Center
- The DB2 OLAP Server
- The DB2 OLAP Integragtion Server
- The Hyperion Essbase Server

This chapter refers to metadata from DB2 OLAP Server, Hyperion Essbase Server, and the DB2 OLAP Integration Server metadata as OLAP server metadata unless it is necessary to differentiate among the three servers.

Publish and synchronize metadata

This section describes the process of publishing metadata to an information catalog and updating the information catalog metadata when the metadata changes in the Data Warehouse Center or in the DB2® OLAP Server or Hyperion Essbase Server. You use the Data Warehouse Center user interface to publish Data Warehouse Center, DB2 OLAP Server, or Hyperion Essbase Server metadata.

To publish and synchronize metadata in the information catalog, you must complete the following tasks:

1. Identify objects whose metadata you want to publish to the information catalog.
2. Publish the metadata.
3. Create a schedule to run the publication on a regular basis.

After metadata is published in the information catalog, you can automate updates of the metadata. This process is called *metadata synchronization*. When you first publish metadata using the Data Warehouse Center user interface, a publication object is created.

When you synchronize metadata, the metadata for an object that is registered in the information catalog is updated either when you run the publication or based on a schedule that you create for the publication. Metadata is not updated in the information catalog in the following situations:

- When a previously published object is deleted in the Data Warehouse Center or the Essbase outline.
- When the name of an object that you previously published to the information catalog changes (If you select to ″Keep existing object names and descriptions″ while publishing, the names will not be changed. If you select ″Overlay existing object name and descriptions″ while publishing, the names are replaced.)

Related tasks:

- “Preparing to publish OLAP server metadata” on page 50
- “Preparing to publish Data Warehouse Center metadata” on page 51

Publishing OLAP server metadata

Preparing to publish OLAP server metadata

This section describes how to exchange metadata from DB2 OLAP Server to Information Catalog Center.

Prerequisites:

- Both the Information Catalog Center and Essbase client must be installed on the default agent site where the Data Warehouse Center is installed.
- The Windows NT/Windows 2000 warehouse agent site must have access to the Essbase APIs and the information catalog APIs.
- You must verify that the OLAP server environment variable entries are system variables, not user variables. You can check the value of your system and user variables on the Environment page of the System notebook accessible from the Windows Control Panel.

Restrictions:

You can use the Data Warehouse Center user interface to publish DB2 OLAP Server or Hyperion Essbase Server metadata. To use the Data Warehouse Center, see the online help for the Publish OLAP Server Metadata notebook.

Steps:

To publish metadata, you must first identify the metadata that you want to publish and then set up the synchronized updates. Use the instruction below to identify metadata objects and register them for synchronization.

Table 3 provides the mapping between OLAP server and information catalog object types when objects are published to the information catalog.

Table 3. Mapping between object types

OLAP server object type	Information catalog object type
Outline	Multi-dimensional databases
Dimensions in an outline	Dimensions within a mult-dimensional database
Members in a dimension	Members within a multi-dimensional database

Use the **Define OLAP Publication** function of the Data Warehouse Center to identify and synchronize metadata objects.

If the publish program is not completing successfully, you might need to adjust the DB2 database configuration values for the application heap size and the log file size. If either of these values is too small for the amount of information that you plan to publish, one of the following errors will appear in the publish log file:

- SQL0964C The transaction log file for the database is full.
- SQL0954C Not enough storage is available in the application heap to process the statement.

Related concepts:

- "Regular updates of DB2 OLAP Server or Hyperion Essbase Server metadata" on page 51

Related reference:

- "Metadata mappings between the Information Catalog Center and OLAP server" on page 113

Regular updates of DB2 OLAP Server or Hyperion Essbase Server metadata

To synchronize DB2® OLAP Server or Hyperion Essbase Server metadata with metadata that you previously published to the information catalog, use the Data Warehouse Center user interface (which includes the schedule function). This function is only available from the Data Warehouse Center on the Windows® platform and it can only be scheduled to run on the default agent on the Windows platform. This is different from the DWC Publish function, which can be accessed at DWC on all platforms, and scheduled to run on Windows, AIX®, Unix, or Linux agents. You can create a schedule for the publication to run on a regular basis.

After the publication runs, the objects that you identified are checked for updates since metadata was last exchanged with the information catalog. If there were updates, the updated metadata is copied to the information catalog.

The processing log file that shows the results of the metadata synchronization is located in the directory that is specified by the VWS_LOGGING environment variable. The default value of the VWS_LOGGING variable for Windows NT® is `\SQLLIB\LOGGING`. View the file \SQLLIB\LOGGING\ICMOLAP.OUT (located on the drive where you installed the DB2 Universal Database) to check the results. When there is new processing status, the status is appended to the existing log file.

Related tasks:

- "Preparing to publish OLAP server metadata" on page 50

Publishing Data Warehouse Center metadata

Preparing to publish Data Warehouse Center metadata

This section describes how to exchange metadata from DB2 Data Warehouse Center to Information Catalog Center

Prerequisites:

- Ensure that you installed and configured the necessary warehouse components on the correct workstations.
- Ensure that both the administrator user IDs for the Information Catalog Center and Data Warehouse Center have Windows administrator privileges.

Restrictions:

- The Information Catalog Center administrator function must be installed both on the warehouse server site and on the Data Warehouse Center administrative interface component if they are on different workstations.

Steps:

To publish Data Warehouse Center metadata, you must first identify the metadata that you want to publish and then set up the synchronized exchange.

Table 4 provides the mapping between object types in the Data Warehouse Center and information catalog. Data Warehouse Center uses this mapping when you export the metadata to the information catalog.

Table 4. Mapping between Data Warehouse Center and information catalog object types

Data Warehouse Center object type	Information catalog object type
Step	Transformation (at the table or column level)
Column or field	Columns or fields
Warehouse source or warehouse target	Databases, IMS™ database definitions
Subject	Business subject areas
Table, file or segment	IMS segments, relational tables, views
Warehouse schema	Star Schema
Map	Column mapping

For detailed task information on publishing metadata to the information catalog, see the Data Warehouse Center online help for the Publish Data Warehouse Center Metadata notebook.

Related concepts:

- "How Data Warehouse Center metadata is displayed in the information catalog" on page 52
- "Maintenance for published objects in the Data Warehouse Center" on page 53
- "Regular updates to Data Warehouse Center metadata" on page 53

Related reference:

- "Metadata Mappings between the Information Catalog Center and the Data Warehouse Center" on page 105

How Data Warehouse Center metadata is displayed in the information catalog

In the Data Warehouse Center, users begin to work with a data source. Users can then create steps (for example, using SQL logic) to transform the data. The resulting data can be a warehouse target table or file. Because an end-user works with data in its transformed state, the Information Catalog Center displays Data Warehouse Center metadata beginning with the end-result of a transformation (for example, a table or a file). You can expand the Show Lineage Tree view in the information catalog of the metadata to determine all the data sources that were input to a transformation. If you expand the Tree view, you can follow the path from target to transformation to data source.

Related tasks:

- "Preparing to publish Data Warehouse Center metadata" on page 51

Maintenance for published objects in the Data Warehouse Center

The action of publishing metadata from Data Warehouse Center to Information Catalog will create a copy of the metadata in the Information Catalog. To keep this metadata up to date the publish steps should be scheduled in Data Warehouse Center to run periodically to update the metadata. The Information Catalog userid that is entered on the Define window for a Publish step must have either Information Catalog Administrator or Power User authority. This is needed because the Publish action will be creating, deleting, and updating objects and relationships.

Deletions are not automatically propagated from Data Warehouse Center to information catalog; you must delete these items manually. If the option **Delete existing objects and recreate** is selected on the publish window then all underlying objects are deleted and recreated with each run of publish. This will result in the deletion of any old objects that no long exist in Data Warehouse Center.

If you change the name of a warehouse object that you previously published to the information catalog, you must publish the object again to update the information catalog with the option **Overwrite exiting object names and descriptions** selected on the publish window. If you would like to preserve the object names and descriptions that are already in the information catalog then you should publish with the option **Keep existing object names and descriptions**. Keep in mind that unique identifier properties should never be changed in the Information Catalog Center.

If the source to target mapping option **Table level** is selected on the publish window then Data warehouse center sources and targets will be published only the tables not the columns in the tables. If the source to target mapping option **Column level** is selected on the publish window then Data Warehouse Center sources and targets will be published with the columns in the tables. When published Steps have column mapping, and the **Column level** option is selected, the column mapping is also published.

Related tasks:

• "Preparing to publish Data Warehouse Center metadata" on page 51

Regular updates to Data Warehouse Center metadata

To synchronize Data Warehouse Center metadata with metadata that was previously published to the information catalog, you must use the Data Warehouse Center administrative interface to create a schedule for the publication to run.

After running, the log files are located in the directory that is specified by the VWS_LOGGING environment variable. The default value of the VWS_LOGGING variable for Windows® is \SQLLIB\LOGGING, (located on the drive where you installed the DB2® Universal Database). The file will be named pubdwcyyyyMMddhhmmss.log where the ending is the date and time run.

Related tasks:

• "Preparing to publish Data Warehouse Center metadata" on page 51

Chapter 7. Maintaining the Information Catalog Center

This chapter provides information you can use to maintain the information catalog center. It also provides basic problem solving tips.

Maintenance

You can keep the Information Catalog Center running smoothly by:
- Monitoring available disk space.
- Ensuring your LAN configuration provides enough resources for the Information Catalog Center.
- Ensuring users can access the information catalog concurrently.

Your LAN or database administrator can help with most of these tasks, or you can refer to your database documentation for more information.

To maintain good performance of the database, it is a good idea to use the DB2 Universal Database RUNSTATS and REORG utilities. The RUNSTATS utility updates statistics in the DB2® UDB system catalog tables to help with the query optimization process. Without those statistics, the database manager might make a decision that can adversely affect the performance of an SQL statement. Use the REORG utility to help arrange the data in tables and indexes more efficiently. See the administration guide for your DB2 database system for the more information.

Related concepts:
- "Backup" on page 56
- "Problem solving" on page 57

Related tasks:
- "Backing up information catalog databases" on page 56
- "Monitoring available disk space" on page 55

Monitoring available disk space

Regularly monitor the space available on the drive that contains the information catalog database, so that your organization doesn't run out of space as the information catalog grows. If this happens, the Information Catalog Center can fail, and users will not be able to access the information catalog. Also monitor the drive on the user's workstation that contains theWindows paging file. On Windows NT, you can view or edit this file:

Procedure:

To view or edit the Windows paging file on Windows NT:
1. Open the Control Panel.
2. Double-click **System** to open the System Properties notebook.
3. On the Performance page, click **Virtual Memory**.
4. Edit the **Total paging file size** field.
5. Click **OK** to close the System Properties notebook.

If your Information catalog is stored in a DB2 Universal Database: See the online help for the DB2 Universal Database Control Center for information about changing the size of the log files.

Related concepts:

- "Maintenance" on page 55

Backup

To avoid losing your data in case of a hardware or software failure, establish a routine for backing up your information catalog databases, configuration information, and supporting software.

How frequently you back up these components depends on how frequently you make changes to your information catalog, and on your organization's policies for backups.

Your routine should include the following tasks:

- Backing up your LAN server system.
- Backing up each information catalog database.

  Making sure you back up each table space associated with the catalog.
- Backing up your data to tape, to a separate physical or LAN drive, or to diskettes.
- Backing up your data before making major changes to it.
- Backing up your data after you import tag language files that contain major changes to the information catalog.
- Backing up your data on a weekly basis if you make frequent changes to it.

Work with your LAN or database administrator to carry out your backup routine.

Related tasks:

- "Monitoring available disk space" on page 55
- "Recovering Information Catalog Center components and data" on page 57

Backing up information catalog databases

Backing up information catalog databases is crucial to ensure that you can recover your descriptive data if your databases become inconsistent or corrupt.

Procedure:

Use the DB2 BACKUP utility to back up the information catalog database. See the administration guide for your DB2 database system for more information.

Related concepts:

- "Backup" on page 56

Related tasks:

- "Recovering Information Catalog Center components and data" on page 57

Problem solving

The Information Catalog Center gives you some resources to help you solve problems. These resources are:

- Online information and messages.

  The Information Catalog Center provides extensive online information and messages to help you solve problems. When you or your users receive a message, use the online help first to resolve the problem.

  You can find help for Information Catalog Center messages and explanations in the *Message Reference*.

  You can also look up message help from a DB2® command line by typing

  ```
  db2 ? ICMnnnn
  ```

  where **nnnn** represents the id number of the message.
- Information Catalog Center trace file.

Related concepts:

- "Maintenance" on page 55

Related tasks:

- "Recovering Information Catalog Center components and data" on page 57

Recovering Information Catalog Center components and data

If you experience a hardware or software failure, you can lose your information catalog database, your descriptive data, and parts of the component. If you backed up the necessary components and data, you can restore your system, the Information Catalog Center, and data.

Steps:

If a system failure occurs, perform the following steps after the database server's hard disk is restored and before your users access the information catalog:

1. Recover your database management system and reinstall the Information Catalog Center, as necessary.
2. Restore the information catalog databases by using your backup files.

Related concepts:

- "Backup" on page 56

Related tasks:

- "Monitoring available disk space" on page 55

Chapter 8. Performing Information Catalog Center tasks with the tag language

The Information Catalog Center provides a tag language. You can use the tag language to perform many of the tasks that you can perform from the graphical interface. The tag language is more difficult to use because you must learn syntax rules to code a tag language file. However, it is especially powerful for performing tasks in bulk.

Tag language

The Information Catalog Center tag language allows you to format your metadata so that you can import it into your information catalog. The tag language tells the Information Catalog Center what to do with the metadata that it imports.

By formatting metadata with the tag language, you can move metadata from one information catalog to another and define Information Catalog Center object types and objects. You can also write and use extract programs to extract metadata from other sources, such as a relational database catalog, that you can import to your information catalog The following table shows the tags in the tag language and the actions that these tags perform.

Table 5. Information Catalog Center tags

Task	Tag names
Identify action to be taken on input data	ACTION.OBJINST ACTION.OBJTYPE ACTION.RELATION ACTION.RELTYPE
Describe data to the information catalog	OBJECT PROPERTY INSTANCE RELTYPE RELATIONTYPE CONREL
Identify when changes are committed and where check point occurs	COMMIT
Identify user comments	COMMENT
Format data	NL TAB

How to read the examples of tag language syntax in the topics

Code the tags and keywords exactly as they are shown in the text. The tags and keywords are represented like this:

```
:tagname.keyword() keyword()
```

Valid values that you can substitute for variables are described in the keyword list. The values are represented like this: *variable*

In tag descriptions, a vertical bar in each pair of keywords or values indicates that you must include one of the pair with the tag. For example, the syntax for the PROPERTY tag includes the NULLS keyword values NULLS(Y|N). You must code either NULLS(Y) or NULLS(N).

Related reference:

- "NL" on page 81
- "OBJECT" on page 81
- "TAB" on page 93
- "COMMIT" on page 75
- "COMMENT" on page 75
- "INSTANCE" on page 76
- "ACTION.OBJINST" on page 65
- "ACTION.OBJTYPE" on page 69
- "PROPERTY" on page 86
- "ACTION.RELATION" on page 72
- "RELTYPE" on page 91
- "ACTION.RELTYPE" on page 74
- "RELATIONTYPE" on page 89

Rules for writing tag language files

The rules explained in this section apply to all tag language files.

- Each tag name must start with a colon and end with a period. Do not put spaces between the colon and the tag name, or between the tag name and the period. For example:

  ```
  :ACTION.OBJINST.
  ```

  The tag name must be one of the tag names that are listed in Tag Language.
- Include at least one keyword with all tags except COMMENT, NL, or TAB.
- Write the keyword and its value like this:

  ```
  keyword(value)
  ```
- Specify keywords in any order. The only exception is that the SOURCEKEY keyword of the INSTANCE tag must be the first keyword.
- Use a blank to separate keywords.
- Enclose in parentheses the value of a keyword. If the value contains a parenthesis, enclose the parenthesis in a pair of apostrophes; for example:

  ```
  keyword(value'('1')')
  ```
- Do not use the first four characters `ICM$` in your property short names (*short_name*) with the PROPERTY tags or the INSTANCE tags. This character prefix is reserved by Information Catalog Center.
- The property name `NAME` is reserved by Information Catalog Center:

  You can specify NAME as the *short_name* on the PROPERTY tag if you identify NAME as a unique identifier property for an object type when using ACTION.OBJTYPE(ADD) or ACTION.OBJTYPE(MERGE), as shown:

  ```
  :PROPERTY.SHRTNAME(NAME) UUISEQ(1)
  ```

Related reference:

- "Tag language" on page 59

How the Information Catalog Center reads tag language files

When you code a tag language file, consider how the Information Catalog Center:

- Reads the entire tag language file as a continuous data stream.
- Treats any character with a hexadecimal value under X'20' (except for tab and new line character tags that are specified in property values) as a control character and ignores that character.
- Considers a tag complete when it encounters the next tag in the tag language file.
- Does not translate tags and keywords into national languages.
- Only recognizes the values for the keywords in the following table to be enabled for double-byte character set (DBCS) support.

Table 6. Keyword values enabled for DBCS

Tag name	Keywords	Variable value
OBJECT	EXTNAME DESCRIPTION ICWFILE	*name* *description* *GIF_file_name*
PROPERTY	EXTNAME DESCRIPTION	*name* *description*
COMMIT	CHKPID	*checkpt_id*
INSTANCE	*UI_name* *or* *name*	*UI_property_value* *or* *property_value*
RELATIONTYPE	EXTNAME DESCRIPTION	*name* *description*

  All user-defined property values can use DBCS characters.
- Accepts DBCS blanks only in the keyword values that are shown in the following table. If DBCS blanks appear anywhere else in the tag language file, errors can occur.

Table 7. Keyword values enabled for DBCS blank characters

Tag name	Keywords
ACTION	OBJTYPE OBJINST RELATION RELTYPE
OBJECT	All keywords
PROPERTY	All keywords
RELTYPE	All keywords
RELATIONTYPE	All keywords
COMMIT	CHKPID
INSTANCE	*UI_name* *or* *name*

Related reference:

- "Tag language" on page 59

Valid data types for Information Catalog Center descriptive data

The following table shows the valid data types for Information Catalog Center descriptive data.

Table 8. Valid data types for Information Catalog Center descriptive data

Data type	Description
INTEGER (I)	A integer is a four byte integer with a precision of 10 digits. The range of integers is -2 147 483 648 to +2 147 483 647.
SMALLINT (S)	A small integer is a two byte integer with a precision of 5 digits. The range of small integers is -32 768 to 32 767.
BIGINT (G)	A big integer is an eight byte integer with a precision of 19 digits. The range of big integers is -9 223 372 036 854 775 808 to +9 223 372 036 854 775 807.
DECIMAL (E)	A decimal value is a packed decimal number with an implicit decimal point. The position of the decimal point is determined by the precision and the scale of the number. The scale, which is the number of digits in the fractional part of the number, cannot be negative or greater than the precision. The maximum precision is 31 digits
DOUBLE (U)	A double-precision floating-point number is a 64 bit approximation of a real number. The number can be zero or can range from -1.79769E+308 to -2.225E-307, or from 2.225E-307 to 1.79769E+308.
REAL (R)	A single-precision floating-point number is a 32 bit approximation of a real number. The number can be zero or can range from -3.402E+38 to -1.175E-37, or from 1.175E-37 to 3.402E+38.
BLOB (B)	Binary large object. A sequence of bytes with a size ranging from 0 bytes to 2 gigabytes, less 1 byte. You cannot specify a property with a data type of BLOB as a unique identifier property.
CLOB (O)	Character large object. A sequence of characters (single-byte, multibyte, or both) with a size ranging from 0 bytes to 2 gigabytes, less 1 byte. You cannot specify a property with a data type of CLOB as a unique identifier property.
CHAR (C)	Fixed-length character string between 1 and 254 bytes long. Pad the value on the right with trailing blanks if the value is shorter than the defined data length for the property.
TIMESTAMP (T)	26-character timestamp in the following format: `yyyy-mm-dd-hh.mm.ss.nnnnnn`
TIME (M)	15-character time in the following format: `hh.mm.ss.nnnnnn`
DATE (D)	10-character date in the following format: `yyyy-mm-dd`
LONG VARCHAR (L)	Long varying-length character string between 1 and 32 700 bytes long. You cannot specify a property with a data type of LONG VARCHAR as a unique identifier property.
VARCHAR (V)	Varying-length character string between 1 and 32 672 bytes long.

A LOB property (BLOB or CLOB) can have a length up to the limit supported by the database server where the catalog is stored. When writing information in a BLOB or CLOB property, the information is buffered in memory by the database client. To store more than 200-300 kilobytes in a LOB property, the Java Virtual Machine (JVM) running the Information Catalog Center on the client platform must have enough space for all of the BLOB or CLOB information to fit. The physical memory and paging space of the client platform must also be sufficient to execute the large JVM. The size of the JVM can be adjusted in the script that starts the Information Catalog Center. The physical memory and paging space limits are configured by the client operating system. Consult the operating system documentation to modify the physical memory and paging space limits.

When importing a tag language file, the Information Catalog Center automatically removes the trailing blanks from variable values and adjusts their length accordingly before validating and accepting the request.

A required value must be specified; otherwise an error will occur.

Related concepts:

- "Object definition for the Data Warehouse Center" in the *Data Warehouse Center Application Integration Guide*

Related reference:

- "Tag language" on page 59

Tag language file content for the Information Catalog Center

You can use the tags to add, delete, and update object types and objects. Information Catalog Center tags are contextual; you specify tags in different combinations depending on what you want to do.

Define your additions, changes, and deletions

You use the tag language to define actions and the objects of those actions.

Defining what you want to do

The ACTION tag tells Information Catalog Center what you want to do. The keyword tells the Information Catalog Center what kind of information you want to maintain. The option tells the Information Catalog Center what task you want to perform.

:ACTION.OBJINST(*option***)**
Maintaining objects.

:ACTION.OBJTYPE(*option***)**
Maintaining object types.

:ACTION.RELATION(*option***)**
Maintaining object relationships.

:ACTION.RELTYPE(*option***)**
Maintaining relationship types.

Defining the information

After you have specified what you want to do, you need to define precisely what information you are adding, changing, or deleting.

To define:	Use these tags:
Existing object type	OBJECT
Object type to be merged	OBJECT and PROPERTY
New object type	OBJECT and PROPERTY
New properties for an object type	OBJECT and PROPERTY
New or existing object	OBJECT and INSTANCE
New or existing object relationship	RELTYPE and INSTANCE
New relationship type	RELATIONTYPE

Putting it all together

The keywords and values that are required for OBJECT, INSTANCE, PROPERTY and RELATIONTYPE tags are different depending on what they are identifying to add, change, or delete. The sequence of tags within each ACTION tag is:

:ACTION.OBJINST(*option*)

```
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name() ...

:ACTION.OBJINST(DELETE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)

:ACTION.OBJINST(DELETE_TREE_ALL)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)

:ACTION.OBJINST(DELETE_TREE_REL)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)

:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(shortname)
:INSTANCE. short_name() ...

:ACTION.OBJINST(UPDATE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...) short_name()
```

:ACTION.OBJTYPE(*option*)

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE(shortname) EXTNAME() DESCRIPTION()  ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()

:ACTION.OBJTYPE(APPEND)
:OBJECT.TYPE(shortname)
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()

:ACTION.OBJTYPE(DELETE)
:OBJECT.TYPE(shortname)

:ACTION.OBJTYPE(DELETE_EXT)
:OBJECT.TYPE(shortname)

:ACTION.OBJTYPE(MERGE)
:OBJECT.TYPE(shortname) EXTNAME() DESCRIPTION()  ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()

:ACTION.OBJTYPE(UPDATE)
:OBJECT.TYPE(shortname) EXTNAME() DESCRIPTION() ICWFILE()
```

:ACTION.RELATION(*option*)

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(ATTACHMENT | CONTACT | DICTIONARY | SUPPORTED |
              CONTAINS | INPUT | OUTPUT | CASCADE | LINKED)
 SOURCETYPE(type)
 TARGETYPE(type)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

```
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE(ATTACHMENT | CONTACT | DICTIONARY | SUPPORTED |
              CONTAINS | INPUT | OUTPUT | CASCADE | LINKED)
 SOURCETYPE(type)
 TARGETYPE(type)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

:ACTION.RELTYPE(*option***)**

```
:ACTION.RELTYPE(ADD)
:RELATIONTYPE.TYPE() EXTNAME() DESCRIPTION() CATEGORY()
:ACTION.RELTYPE(MERGE)
:RELATIONTYPE.TYPE() EXTNAME() DESCRIPTION() CATEGORY()
:ACTION.RELTYPE(DELETE)
:RELATIONTYPE.TYPE()
```

For specific information about the format of the tags. see INSTANCE, OBJECT, PROPERTY, and RELATIONTYPE tags

Committing changes to the database

The COMMIT tag commits changes to the information catalog database. When a COMMIT tag processes, the echo file is emptied before the next set of tags starts processing. This ensures that the echo file contains only tags that describe uncommitted changes.

If the Information Catalog Center encounters an error, it rolls back the database to the last committed checkpoint. Insert COMMIT tags in your file to keep your data consistent, and to limit the number of changes that are canceled when the database is rolled back.

You can insert a COMMIT tag after any complete set of tags that define an action. Do not insert a COMMIT tag between the ACTION tag and the last tag that defines the data that is associated with the ACTION tag.

```
:COMMIT.CHKPT(20)
```

Putting comments in the tag language file

You can use the COMMENT tag to put information in the tag language file, such as notes and labels, that you do not want to import into your information catalog.

```
:COMMENT.Updating the LASTDATE property
```

Related reference:

- "ACTION.OBJINST" on page 65
- "ACTION.OBJTYPE" on page 69
- "ACTION.RELATION" on page 72
- "ACTION.RELTYPE" on page 74
- "Tag language" on page 59

Tag descriptions

This sections describes the tags and keywords in the Information Catalog Center tag language.

ACTION.OBJINST

Identifies the action to be performed on the object that is described with the tags that follow the ACTION tag.

Context

ACTION.OBJINST is used to create, delete, or maintain Information Catalog Center objects.

ACTION.OBJINST is followed by one or more OBJECT and INSTANCE tags, which define the object to act on.

Syntax

:ACTION.OBJINST(*option***)**

Options

The following options are valid for ACTION.OBJINST:

ADD
DELETE
DELETE_TREE_ALL
DELETE_TREE_REL
MERGE
UPDATE

ACTION.OBJINST(ADD): Adds an object.

Context:

```
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
:INSTANCE.short_name()

:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
:INSTANCE.short_name()
```

Figure 1. Using the ACTION.OBJINST tag when adding objects

Rules:

- The object must not already exist.
- Both the OBJECT tag and the INSTANCE tag must follow the ACTION.OBJINST(ADD) tag.
  - The OBJECT tag identifies the object type for the new object.
  - The INSTANCE tag specifies the property values for the new object.
- One or more INSTANCE tags can follow a single OBJECT tag, if the objects are for the same object type.
- One or more sets of an OBJECT tag with INSTANCE tags can follow an ACTION.OBJINST(ADD) tag to describe objects of different object types to add.

ACTION.OBJINST(DELETE): Deletes an object.

Context:

```
:ACTION.OBJINST(DELETE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)

:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)
```

Figure 2. Using the ACTION.OBJINST tag when deleting objects

Rules:

- The specified object must already exist.
- Both the OBJECT tag and the INSTANCE tag must follow the ACTION.OBJINST(DELETE) tag.
  - The OBJECT tag identifies the object type for the object to be deleted.
  - The INSTANCE tag specifies the unique identifier property values for the object to be deleted.
- One or more INSTANCE tags can follow a single OBJECT tag, if the objects are for the same object type.
- One or more sets of an OBJECT tag with INSTANCE tags can follow an ACTION.OBJINST(DELETE) tag to describe objects of different object types to delete.

ACTION.OBJINST(DELETE_TREE_ALL):

Note: This option is for Information Catalog Manager Version 7 compatibility only.

Deletes a Grouping category object, all Comments objects that are attached to it, and all ATTACHMENT, CONTACT, and LINK relationships in which it participates. Deletes all objects that are contained in the Grouping category object, all Comments objects attached to them, and all ATTACHMENT, CONTACT, and LINK relationships in which they participate.

Context:

```
:ACTION.OBJINST(DELETE_TREE_ALL)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)

:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)
```

Figure 3. Using the ACTION.OBJINST tag when deleting Grouping category objects and contained objects

Rules:

- The specified object must already exist and be a Grouping category object.
- Both the OBJECT tag and the INSTANCE tag must follow the ACTION.OBJINST(DELETE_TREE_ALL) tag.
  - The OBJECT tag identifies the object type for the object to delete.
  - The INSTANCE tag specifies the UUI property values for the object that is being deleted.
- One or more INSTANCE tags can follow a single OBJECT tag, if the objects are for the same object type.

- One or more sets of an OBJECT tag with INSTANCE tags can follow an ACTION.OBJINST(DELETE_TREE_ALL) tag to describe objects of different object types to be deleted.

ACTION.OBJINST(DELETE_TREE_REL):

Note: This option is for compatibility for Information Catalog Manager Version 7 only.

Deletes a Grouping category object, all Comments objects attached to it, and all ATTACHMENT, CONTACT, CONTAIN, and LINK relationships in which it participates.

Context:

```
:ACTION.OBJINST(DELETE_TREE_REL)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)

:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)
```

Figure 4. Using the ACTION.OBJINST tag when deleting Grouping category objects and relationships

Rules:

- The specified object must already exist and be a Grouping category object.
- Both the OBJECT tag and the INSTANCE tag must follow the ACTION.OBJINST(DELETE_TREE_REL) tag.
  - The OBJECT tag identifies the object type for the object being deleted.
  - The INSTANCE tag specifies the unique identifier property values for the object being deleted.
- One or more INSTANCE tags can follow a single OBJECT tag, if the objects are for the same object type.
- One or more sets of an OBJECT tag with INSTANCE tags can follow an ACTION.OBJINST(DELETE_TREE_REL) tag to describe objects of different object types to be deleted.

ACTION.OBJINST(MERGE): Searches for the input object's Unique Identifier in the information catalog to see whether the input object exists.

If the object exists, the Information Catalog Center updates the property values of the object in the information catalog. If the object does not exist, the Information Catalog Center creates a new object.

Context:

```
:ACTION.OBJTYPE(MERGE)
:OBJECT.TYPE(shortname)  EXTNAME() DESCRIPTION() ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()

:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
```

Figure 5. Using the ACTION.OBJINST tag when merging objects

Rules:

- If the object exists, the Information Catalog Center updates the property values of the object in the information catalog. If the object does not exist, the Information Catalog Center creates a new object.
- Both the OBJECT tag and the INSTANCE tag must follow the ACTION.OBJINST(MERGE) tag.
  - The OBJECT tag identifies the object type for the object being merged.
  - The INSTANCE tag specifies the property values for the object being merged.

ACTION.OBJINST(UPDATE): Updates the value of an object.

Context:

```
:ACTION.OBJINST(UPDATE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...) short_name()
```

Figure 6. Using the ACTION.OBJINST tag when updating objects

Rules:

- The specified object must already exist.
- Both the OBJECT tag and the INSTANCE tag must follow the ACTION.OBJINST(UPDATE) tag.
  - The OBJECT tag identifies the object type for the object being updated.
  - The INSTANCE tag specifies the unique identifier property values, which identify the object to be updated, and the property values that are being updated.

  Only the property values specified on the INSTANCE tag are updated.

Related reference:

- "OBJECT" on page 81
- "INSTANCE" on page 76
- "Tag language" on page 59

ACTION.OBJTYPE

Identifies the action to perform on the object type that is described with the tags that follow ACTION.OBJTYPE.

Context

ACTION.OBJTYPE is used to create, delete, or maintain Information Catalog Center object types.

ACTION.OBJTYPE is followed by one or more OBJECT and PROPERTY tags, which define the object type being acted on.

Syntax

:ACTION.OBJTYPE(*option***)**

Options

The following options are valid with ACTION.OBJTYPE:

ADD
APPEND
DELETE
DELETE_EXT
MERGE
UPDATE

ACTION.OBJTYPE(ADD): Creates the object type.

Context:

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE(shortname) EXTNAME() DESCRIPTION() ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 7. Using the ACTION.OBJTYPE tag when adding object types

Rules:

- The object type must not exist.
- An OBJECT tag and its associated PROPERTY tags must immediately follow the ACTION.OBJTYPE(ADD) tag.
  - The OBJECT tag defines the attributes of the new object type.
  - The PROPERTY tags define the properties that belong to the new object type.

ACTION.OBJTYPE(APPEND): Appends a property to an existing object type.

Context:

```
:ACTION.OBJTYPE(APPEND)
:OBJECT.TYPE(shortname)
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 8. Using the ACTION.OBJTYPE tag when adding properties to object types

Rules:

- The object type must exist.
- The property being appended must not exist.
- Do not assign the property a UUISEQ value other than 0 (the default). Appended properties must be optional with NULLS(Y) and cannot be part of the UI.

- An OBJECT tag and one or more PROPERTY tags must immediately follow the ACTION.OBJTYPE(APPEND) tag.
  - The OBJECT tag identifies the object type being appended.
  - Each PROPERTY tag defines a property being appended.

ACTION.OBJTYPE(DELETE): Deletes the object type.

Context:

```
:ACTION.OBJTYPE(DELETE)
:OBJECT.TYPE(shortname)
```

Figure 9. Using the ACTION.OBJTYPE tag when deleting object types

Rules:

- The object type must exist. No objects of the object type can exist.
- One or more OBJECT tags must follow an ACTION.OBJTYPE(DELETE) tag. Each OBJECT tag identifies the object type being deleted.

ACTION.OBJTYPE(DELETE_EXT): Deletes the object type and objects of that object type.

Context:

```
:ACTION.OBJTYPE(DELETE_EXT)
:OBJECT.TYPE(shortname)
```

Figure 10. Using the ACTION.OBJTYPE tag when deleting object types and all objects of that type

Rules:

- The object type must exist.
- The object cannot contain objects of a different object type.
- One or more OBJECT tags must follow the ACTION.OBJTYPE(DELETE) tag. Each OBJECT tag identifies the object type being deleted.

ACTION.OBJTYPE(MERGE): Checks the information catalog for the input object type name to see if the object type exists.

If the object type exists, the Information Catalog Center compares properties of the input object type to the properties of the stored object type. If the properties match, then the object types are treated as identical; if not, the input object type is not valid.

If the object type does not exist, the Information Catalog Center creates a new object type.

Context:

```
:ACTION.OBJTYPE(MERGE)
:OBJECT.TYPE(shortname) EXTNAME() DESCRIPTION() ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()

:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
```

Figure 11. Using the ACTION.OBJTYPE tag when merging object types

Rules:

- An OBJECT tag and its associated PROPERTY tags must immediately follow the ACTION.OBJTYPE(MERGE) tag.
  - The OBJECT tag defines the object type being merged.
  - Each PROPERTY tag defines a property that belongs to the object type.

ACTION.OBJTYPE(UPDATE): Changes an object-type external name and ICON file information.

Context:

```
:ACTION.OBJTYPE(UPDATE)
:OBJECT.TYPE(shortname) EXTNAME() ICWFILE()
```

Figure 12. Using the ACTION.OBJTYPE tag when updating object types

Rules:

- The object type must already exist.
- One or more OBJECT tags must follow the ACTION tag.

Related reference:

- "OBJECT" on page 81
- "PROPERTY" on page 86
- "Tag language" on page 59

ACTION.RELATION

Identifies the action to perform on the relationship that is described with the tags that follow ACTION.RELATION.

Context

ACTION.RELATION is used to create or delete information catalog relationships.

ACTION.RELATION is followed by one or more RELTYPE and INSTANCE tags, which define the relationships being acted on.

Syntax

:ACTION.RELATION(*option***)**

Options

The following options are valid with ACTION.RELATION:

ADD

DELETE

ACTION.RELATION(ADD): Defines an ATTACHMENT, CONTACT, DICTIONARY, SUPPORTED, CONTAINS, INPUT, OUTPUT, CASCADE, LINKED, or user defined relationship.

Context:

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(type_short_name) SOURCETYPE(source_object_type_short_name)
    TARGETTYPE(target_object_type_short_name)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

Figure 13. Using the ACTION.RELATION tag when adding relationships

Rules:

- If the specified relationship does not exist, the relationship is added. If the specified relationship exists, the Information Catalog Center writes an informational message and continues processing.
- A RELTYPE tag and one or more INSTANCE tags must immediately follow the ACTION.RELATION(ADD) tag.
  - The RELTYPE tag defines the type of relationship that is being added and specifies the object types of the objects to associate.
  - Each INSTANCE tag specifies the unique identifier property values that identify the two objects that are being associated.

ACTION.RELATION(DELETE): Deletes a relationship.

Context:

```
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE(type_short_name) SOURCETYPE(source_object_type_short_name)
    TARGETTYPE(target_object_type_short_name)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

Figure 14. Using the ACTION.RELATION tag when deleting relationships

Rules:

- The relationship is deleted if it exists; otherwise, the Information Catalog Center writes an informational message and continues processing.
- A RELTYPE tag and one or more INSTANCE tags must immediately follow the ACTION.RELATION(DELETE) tag.
  - The RELTYPE tag defines the type of relationship that is being deleted and specifies the object types of the associated objects.
  - Each INSTANCE tag specifies the unique identifier property values that identify the two associated objects.

Related reference:

- "INSTANCE" on page 76
- "RELATIONTYPE" on page 89
- "Tag language" on page 59

ACTION.RELTYPE

Identifies the action to perform on the relationship type that is described with the tags that follow ACTION.RELTYPE.

Context

ACTION.RELTYPE is used to create or delete information catalog relationship types.

ACTION.RELTYPE is followed by one or more RELATIONTYPE tags, which define the relationship types being acted on.

Syntax

:ACTION.RELTYPE(*option***)**

Options

The following options are valid with ACTION.RELTYPE:

ADD
DELETE

ACTION.RELTYPE(ADD): Defines a relationship type.

Context:

```
:ACTION.RELTYPE(ADD)
:RELATIONTYPE.TYPE(RelTypeNAME), CATEGORY(CategoryName), EXTNAME(ExtName),
DESCRIPTION(Description) :SOURCETYPE.NAME(ObjectTypeName) NAME(ObjectTypeName)
:TARGETTYPE.NAME(ObjectTypeName) NAME(ObjectTypeName)
```

Figure 15. Using the ACTION.RELTYPE tag when adding relationship types

Rules:

- If the specified relationship type does not exist, the relationship type is added. If the specified relationship type exists, the Information Catalog Center writes an informational message and continues processing.
- A RELATIONTYPE tag must immediately follow the ACTION.RELTYPE(ADD) tag.
  - The RELATIONTYPE tag defines the relationship type that is being added.

ACTION.RELTYPE(DELETE): Deletes a relationship type, and all relationships of that type.

Context:

```
:ACTION.RELTYPE(DELETE)
:RELATIONTYPE.TYPE()
:RELATIONTYPE.TYPE()
```

Figure 16. Using the ACTION.RELTYPE tag when deleting relationship types

Rules:

- The relationship type is deleted if it exists; otherwise, the Information Catalog Center writes an informational message and continues processing.

- A RELATIONTYPE tag must immediately follow the ACTION.RELTYPE(DELETE) tag.
  - The RELATIONTYPE tag defines the relationship type that is being deleted.

Related reference:
- "RELATIONTYPE" on page 89
- "Tag language" on page 59

COMMENT

Identifies comments in the tag language file. Place this tag between any complete tag specifications in your file.

The Information Catalog Center ignores comments when importing a tag language file.

Syntax

```
:COMMENT.your comments
```

```
:COMMENT.This is the text of a comment.
```

Figure 17. Example of a COMMENT tag

Rules

- You cannot place a COMMENT tag between another tag and its keywords or between keywords.
- The comment text must not contain any Information Catalog Center tags (for example `:ACTION.`), because each tag ends either at the end of the file or at the beginning of the next valid tag.

Related reference:
- "Tag language" on page 59

COMMIT

Identifies a commit point. Requests that the Information Catalog Center commit the current changes to the database.

If the Information Catalog Center encounters an error while importing a tag language file, it rolls back all changes that are made to the information catalog since the last time changes were committed.

Include COMMIT checkpoints at regular intervals so that you import Information Catalog Center tag language files more efficiently.

Including COMMIT checkpoints before and after defining or deleting object types, sets of objects, and sets of relationships can help maintain the integrity of your descriptive data.

Regular COMMIT checkpoints limit the number of changes that the Information Catalog Center cancels when it rolls back the information catalog.

Context

Place this tag after one or more complete action specifications (a set of ACTION, OBJECT, RELTYPE, and INSTANCE tags).

Syntax

```
:COMMIT.CHKPID(checkpt_id)

:COMMIT.CHKPID(Added_relationships)
```

Figure 18. Example of a COMMIT tag

Keywords

CHKPID
Required keyword.

checkpt_id
An identifier that the Information Catalog Center saves when it processes a COMMIT tag.

If the import of a tag language file fails after a COMMIT tag processes successfully, you need to import the rest of the tag language file starting at the last checkpoint. This option is available with the import function. The Information Catalog Center uses the stored *checkpt_id* to locate the proper COMMIT tag.

The value of *checkpt_id* must be unique within each tag language file. Otherwise, the results of restart processing are unpredictable.

The maximum length of *checkpt_id* is 26 characters.

checkpt_id is not case-sensitive.

Rules

Specify a COMMIT tag when the data is consistent.

To prevent the target information catalog transaction log from filling up, specify COMMIT tags at regular intervals in the tag language file.

An ACTION tag must follow the COMMIT tag, if additional data in the same tag language file needs to be processed.

Related reference:

- "Tag language" on page 59

INSTANCE

Context

This tag is required following:

:ACTION.OBJINST	The INSTANCE tag follows an OBJECT tag.
:ACTION.RELATION	The INSTANCE tag follows a RELTYPE tag.

Syntax

There are four formats for the INSTANCE tag, depending on the format of the ACTION tag:

ACTION.OBJINST(ADD) or ACTION.OBJINST(MERGE): Adding or merging objects

:INSTANCE.*short_name* **(***property_value***) . . .**

Context:

```
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
```

Figure 19. Using the INSTANCE tag when adding objects

```
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
:short_name()
:short_name()
```

Figure 20. Using the INSTANCE tag when merging objects

Keywords:

short_name
Identifies each property by its short name. If an INSTANCE tag has multiple short names associated with it, use only one INSTANCE tag followed the short names as shown in Figure 20.

property_value
Specifies the value of the property for the given object. This value is case sensitive.

Rules:

- When adding an object:
  - You must specify all unique identifier values, a value for the NAME property, and values for any other properties that are defined as required.
  - You can omit a property that does not have a value to add from the INSTANCE tag.
- When merging an object:
  - You must specify all unique identifier values, to ensure that matching objects can be identified.
  - You can omit a property that does not have a value to be added or updated.

ACTION.OBJINST(DELETE) or ACTION.OBJINST(DELETE_TREE_ALL) or ACTION.OBJINST(DELETE_TREE_REL): Deleting an object

:INSTANCE.SOURCEKEY(*UI_short_name* **(***UI_property_value***) . . .)**

Context:

```
:ACTION.OBJINST(DELETE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)
```

Figure 21. Using the INSTANCE tag when deleting objects

```
:ACTION.OBJINST(DELETE_TREE_ALL)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)
```

Figure 22. Using the INSTANCE tag when deleting Grouping category objects and contained objects

```
:ACTION.OBJINST(DELETE_TREE_REL)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)
```

Figure 23. Using the INSTANCE tag when deleting Grouping category objects and relationships

Keywords:

SOURCEKEY
Specifies the unique identifier property values that identify a particular object.

SOURCEKEY must be the first keyword of the INSTANCE tag.

UI_short_name
Identifies a unique identifier property name by its short name. Specify all of the *UI_short_name*(*UI_property_value*) combinations. The *UI_short_name* is case sensitive; you can specify this value by using uppercase or lowercase characters.

UI_property_value
Specifies the value of a unique identifier property for a particular object. This value is case sensitive.

Rules: You must specify one *UI_short_name*(*value*) combination for each property that is defined as a unique identifier property for the object type. Each object type has one or more properties defined as unique identifier properties. These properties uniquely identify an object in the information catalog.

ACTION.OBJINST(UPDATE): Updating property values for an object

```
:INSTANCE.SOURCEKEY(UI_short_name (UI_property_value) . . . )
                    short_name (property_value) . . .
```

Context:

```
:ACTION.OBJINST(UPDATE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...) short_name()
```

Figure 24. Using the INSTANCE tag when updating objects

Keywords:

SOURCEKEY
Specifies the unique identifier property values that identify a particular object.

SOURCEKEY must be the first keyword of the INSTANCE tag.

UI_short_name
Identifies a unique identifier property by its short name. The *UI_short_name* is case sensitive; you can specify this value by using uppercase or lowercase characters.

UI_property_value
This value is case sensitive. With *UI_short_name*, specifies the value of a unique identifier property for a particular object.

short_name
Identifies the property to be updated by its short name. The *short_name* is not case sensitive; you can specify this value by using uppercase or lowercase characters.

Do not use the first four characters `ICM$` in your property short names (*short_name*) with the PROPERTY or INSTANCE tags. This character prefix is reserved by Information Catalog Center

property_value
With the property *short_name*, specifies the new value of the property for the given object. This value is case sensitive.

Rules: You must specify one *UI_short_name*(*value*) combination for each property that is defined as a unique identifier property for the object type. Each object type has one or more properties defined as unique identifier properties. These properties uniquely identify an object in the information catalog.

If you specify a property value, that value is updated in the information catalog. If you do not specify a property value, the value is not updated.

ACTION.RELATION(ADD) or ACTION.RELATION(DELETE): Adding or deleting relationships

```
:INSTANCE.SOURCEKEY(UI_short_name (UI_property_value)...)
                   TARGETKEY(UI_short_name (UI_property_value)...)
```

Context:

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(type_short_name) SOURCETYPE(source_object_type_short_name)
    TARGETTYPE(target_object_type_short_name)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

Figure 25. Using the INSTANCE tag when adding relationships

```
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE(type_short_name) SOURCETYPE(source_object_type_short_name)
    TARGETTYPE(target_object_type_short_name)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

Figure 26. Using the INSTANCE tag when deleting relationships

Keywords:

SOURCEKEY
Specifies the unique identifier property values that identify the first object in a relationship.

When the relationship is:	The SOURCEKEY identifies:

Attachment	The object the comment is for
Contact	The object the contact is for
Dictionary	The object the glossary term is for
Supported	The object the support is for
Contains	The parent object
Input	The preceding object for a transformation object
Output	The succeeding object for a transformation object
Cascade	The preceding object in a lineage
Linked	Either object to link

SOURCEKEY must be the first keyword of the INSTANCE tag.

TARGETKEY
Specifies the unique identifier property values that identify the second object in a relationship.

When the relationship is:	The TARGETKEY identifies:
Attachment	The attachment object
Contact	The contact object
Dictionary	The glossary term object
Supported	The supported object
Contains	The child object
Input	The succeeding object for a transformation object
Output	The preceding object for a transformation object
Cascade	The succeeding object in a lineage
Linked	Either object to link

TARGETKEY must be the second keyword of the INSTANCE tag.

UI_short_name
Identifies a unique identifier property name by its short name. This value is case sensitive; you can specify this value by using uppercase or lowercase characters.

UI_property_value
Specifies the value of a unique identifier property for a particular object. This value is case sensitive.

Rules: For each object, you must specify one *UI_short_name*(*value*) combination for each property that is defined as a unique identifier property for the object type. Each object type has one or more properties defined as unique identifier properties. These properties uniquely identify an object in the information catalog.

You must separate each *UI_short_name*(*value*) and *short_name*(*value*) pair with a blank, as shown in Figure 27 on page 81.

```
:INSTANCE.SOURCEKEY(UIname1(value1) UIname2(value2))
  sname3(value3) sname4(value4)
```

Figure 27. Example of an INSTANCE tag with several short names

Leading blanks that are included between the parentheses for a value become part of the value; trailing blanks are removed. The Information Catalog Center counts these blanks as part of the data length when determining whether the length of the value is valid. An error occurs if you include extra leading blanks or trailing blanks on a value that make the entire value longer than the maximum allowed length.

Related reference:

- "ACTION.OBJINST" on page 65
- "ACTION.RELATION" on page 72
- "Tag language" on page 59

NL

Specifies a new line within a property value.

The Information Catalog Center manager reads only NL tags that are specified within non-Unique Identifier property values and ignores all others.

Syntax

```
:NL.
```

Rules

Use NL tags only within the specification of *property_values* in INSTANCE tags.

Related reference:

- "Tag language" on page 59

OBJECT

Defines the attributes for an object type or identifies an object type.

Context

This tag is required immediately following:
ACTION.OBJTYPE
ACTION.OBJINST

Syntax

```
:OBJECT.TYPE(short_name) CATEGORY(category)
    EXTNAME(name) DESCRIPTION(description)
    PHYNAME(table_name) ICWFILE(GIF_file_name)
```

Different OBJECT tag keywords are required or valid depending on the type of ACTION tag the OBJECT tag follows.

ACTION.OBJTYPE(ADD) or ACTION.OBJTYPE(MERGE): Adding or merging object types

Context:

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE(shortname) CATEGORY() EXTNAME() PHYNAME() DESCRIPTION() ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 28. Using the OBJECT tag when adding object types

```
:ACTION.OBJTYPE(MERGE)
:OBJECT.TYPE(shortname) CATEGORY() EXTNAME() PHYNAME() DESCRIPTION() ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 29. Using the OBJECT tag when merging object types

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Defines and identifies the short name for a specific object type.

The value of *short_name* must be unique to an object type across all related information catalogs that contain the same object type. This ensures that objects of this object type can be shared among the related information catalogs. If the value of *short_name* already exists, it is used as a search argument.

The maximum length for the value is 16 characters. This value can start with the characters A - Z, @, or #, and can contain any of these characters plus 0 - 9 and _. No leading blanks or embedded blanks are allowed.

After you create the object type, you cannot change the value of *short_name*.

CATEGORY
Specifies which category to which this object type belongs .

Required keyword.

Note: This option is for Information Catalog Manager Version 7 compatibility.

category
Specifies an Information Catalog Center object category. This value can be one of the following:
GROUPING
ELEMENTAL
Support
CONTACT
DICTIONARY

You cannot specify PROGRAM or ATTACHMENT as the category for a new object type.

You cannot change the information on this keyword after the object type is defined.

EXTNAME
Specifies a longer, descriptive name for the object type. Required keyword.

name
Specifies an extended, descriptive name for the object type. The maximum length for *name* is 200 characters.

This name must be unique within related information catalogs.

The value of *name* is stored in mixed case.

You can change the information on this keyword after the object type is defined.

DESCRIPTION
A description of the object type. Optional keyword.

description
Specifies a description for the object type. The maximum length for *name* is 254 characters.

You can change the information on this keyword after the object type is defined.

PHYNAME
Specifies the name to use when creating the database table that contains information about this object type.

Optional keyword.

Note: This option is for Information Catalog Manager Version 7 compatibility.

table_name
Specifies the name to use when creating the database table that contains object type information.

The maximum length of the name is defined when the Information Catalog Center is installed. The *table_name* value must be unique within the information catalog and cannot contain any SQL reserved words.

By default, *table_name* is the *short_name* that is specified for the **TYPE** keyword. This value is not case sensitive; you can specify this value with uppercase or lowercase characters.

This value can start with the characters A - Z, @, or # and can contain any of these characters, plus 0 - 9 and _. No $, leading blanks, or embedded blanks are allowed. This value cannot be any of the SQL reserved words for the database that is used for the information catalog.

After the table is created, you cannot change its name.

ICWFILE
Specifies the file that contains the Windows icon that is associated with the object type.

Optional keyword.

GIF_File_Name
Specifies the name of the gif icon file to associate with the object type. The maximum length of *GIF_File_Name* is 250 characters. However, this name, combined with the icon path (ICOPATH), can have a maximum length of 259, so the true maximum length depends on the length of the icon path. This file can have any extension. This value is not case sensitive; you can specify this value by using uppercase or lowercase characters.

You can change this value after the object type is created by using ACTION.OBJTYPE(UPDATE). After you specify an icon file to associate with an object type, you can change the associated icon, but the object type must always be associated with an icon.

ACTION.OBJTYPE(APPEND):

Context:

```
:ACTION.OBJTYPE(APPEND)
:OBJECT.TYPE(shortname)
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 30. Using the OBJECT tag when adding properties to object types

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Identifies a specific object type by its short name.

ACTION.OBJTYPE(DELETE) or ACTION.OBJTYPE(DELETE_EXT): Deleting an existing object type.

Context:

```
:ACTION.OBJTYPE(DELETE)
:OBJECT.TYPE(shortname)
```

Figure 31. Using the OBJECT tag when deleting object types

```
:ACTION.OBJTYPE(DELETE_EXT)
:OBJECT.TYPE(shortname)
```

Figure 32. Using the OBJECT tag when deleting object types and all objects of that type

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Identifies a specific object type by its short name.

ACTION.OBJTYPE(UPDATE): Updating object type information.

Context:

```
:ACTION.OBJTYPE(UPDATE)
:OBJECT.TYPE(shortname) EXTNAME() DESCRIPTION() ICWFILE()
```

Figure 33. Using the OBJECT tag when updating object types

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Identifies a specific object type by its short name. You cannot update this value.

EXTNAME
Specifies a descriptive name for the object type. Optional keyword.

name
Specifies an extended, descriptive name for the object type. The maximum length for *name* is 200 characters.

You can update this value.

This name must be unique within related information catalogs.

The value of *name* is stored in mixed case.

DESCRIPTION
A description of the object type. Optional keyword.

description
A description for the object type. The maximum length for *description* is 254 characters.

You can change the information on this keyword after the object type is defined.

ICWFILE
Specifies the file that contains the Windows icon that is associated with the object type.

Optional keyword.

GIF_File_Name
Specifies the name of the gif icon file to associate with the object type.

You can update this value.

The maximum length of *GIF_File_Name* is 250 characters. You cannot use this keyword to specify the drive and path information that identifies where the ICON file resides. You must specify this information as an input parameter for the import function on the user interface or the IMPORT option of the Information Catalog Center command.

ACTION.OBJINST: Adding, updating, deleting, or merging objects

Context:

```
:ACTION.OBJINST(ADD)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
```

Figure 34. Using the OBJECT tag when adding objects

```
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(shortname)
:INSTANCE.short_name()
```

Figure 35. Using the OBJECT tag when merging objects

```
:ACTION.OBJINST(UPDATE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...) short_name()
```

Figure 36. Using the OBJECT tag when updating objects

```
:ACTION.OBJINST(DELETE)
:OBJECT.TYPE(shortname)
:INSTANCE.SOURCEKEY(UI_short_name()...)
```

Figure 37. Using the OBJECT tag when deleting objects

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Identifies a specific object type by its short name.

Related reference:
- "ACTION.OBJINST" on page 65
- "ACTION.OBJTYPE" on page 69
- "Tag language" on page 59

PROPERTY

Defines a property that belongs to an object type.

This tag is required following these ACTION tags:
:ACTION.OBJTYPE(ADD)
:ACTION.OBJTYPE(MERGE)
:ACTION.OBJTYPE(APPEND)

Syntax

```
:PROPERTY.EXTNAME(name) DT(data_type) DL(data_length)
          SHRTNAME(short_name) NULLS(Y | N) UUISEQ(UI_number)
```

Context

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE(shortname) CATEGORY() EXTNAME()  DESCRIPTION() ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 38. Using the PROPERTY tag when adding object types

```
:ACTION.OBJTYPE(MERGE)
:OBJECT.TYPE(shortname) CATEGORY() EXTNAME()  DESCRIPTION() ICWFILE()
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 39. Using the PROPERTY tag when merging object types

```
:ACTION.OBJTYPE(APPEND)
:OBJECT.TYPE(shortname)
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

Figure 40. Using the PROPERTY tag when adding properties to object types

Keywords

EXTNAME
Specifies a descriptive name for the property.

Required keyword.

name
Specifies an extended descriptive name.

The maximum length of *name* is 200 characters. The *name* must be unique within the object type. *name* is stored in mixed case.

DT
Specifies the data type for the property.

Required keyword.

data_type
The data type for the property. You can specify this in either uppercase or lowercase. Valid Values are:

I (INTEGER)
4 bytes

S (SMALLINT)
2 bytes

G (BIGINT)
8 bytes

E (DECIMAL)
16 bytes

U (DOUBLE)
8 bytes

R (REAL)
4 bytes

B (BLOB)
0 bytes to 2 gigabytes of bytes

O (CLOB)
0 bytes to 2 gigabytes of characters

C (CHAR)
Up to 254 characters

V (VARCHAR)
Up to 4 000 characters

L (LONG VARCHAR)
Up to 32 700 characters

T (TIMESTAMP)
26 characters, in this format:

yyyy-mm-dd-hh.mm.ss.nnnnnn

M (TIME)
15-character time in the following format:

hh.mm.ss.nnnnnn

D (DATE)
10-character date in the following format:

yyyy-mm-dd

DL
Specifies the data length or maximum data length for the property.

Required property.

data_length
The data length or maximum data length for the property. Valid values for *data_length* depend on the *data_type* that is defined for this property:

SHRTNAME
Specifies the property short name.

Required keyword.

short_name
The short name for the property. The *short_name* value can be up to 18 characters long. This value can contain only SBCS characters.

This value is case sensitive.

This value can start with the characters A - Z, @, or #,and can contain any of these characters, plus 0 - 9 and _. No leading blanks or embedded blanks are allowed.

This value cannot be any of the SQL reserved words for the database that is used for the information catalog.

NULLS
Specifies whether a value for the property is required for every object. This value can be specified in uppercase or lowercase.

Required keyword.

Y indicates that this value can be null. When appending a new property with the ACTION.OBJTYPE(APPEND) tag, you must specify NULLS(Y), because appended properties must be optional.

N indicates that a value for this property is required.

UUISEQ
Identifies the properties that are used in the Unique Identifier.

Optional keyword; the default value is 0. The UUISEQ keyword is optional for properties that are not part of the UI. The unique identifier is a set of properties that are defined by the administrator as the key that uniquely identifies each object.

UI_number
Specifies the position of the property in the unique identifier sequence. Valid values are 0 — 16. The value 0 means that the property is not part of the UI. A nonzero value for *UI_number* indicates that the property is part of the UI.

All object types defined in the tag language file must have at least one property that is part of the UI. The unique identifier can consist of up to 16 properties.

At least one property must be defined as part of the UI.

When assigning *UI_number* values to more than one property, the numbers of the unique identifier properties must range from 1 to the number of properties in the UI. For example, if three properties are defined as part of the UI, the *UI_number* values must be 1, 2, and 3. You cannot skip numbers in the sequence. The *UI_number* values do not need to be in the same order that the properties are specified.

Rules

- You can define the reserved property NAME as part of the unique identifier when you add a new object type or merge object types. Figure 41 shows the general syntax for identifying NAME as an unique identifier property.

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE(shortname) CATEGORY() EXTNAME()  DESCRIPTION() ICWFILE()
:PROPERTY.SHRTNAME(NAME) UUISEQ()
```

Figure 41. Example of specifying the NAME property as part of the UI

  Empty parentheses in this figure denote values that you must provide in a tag language file.
- The maximum length of the unique identifier fields is 250 bytes.

Related reference:

- "ACTION.OBJTYPE" on page 69
- "Tag language" on page 59

RELATIONTYPE

Identifies the relationship type to add, delete, or update for a relationship category.

This tag is required immediately following one of these tags:
:ACTION.RELTYPE(ADD)
:ACTION.RELTYPE(DELETE)

Syntax

```
:RELATIONTYPE.TYPE(short_name) EXTNAME(name)
                   CATEGORY(relationship_category)
                   DECSCRIPTION(description)
```

Context

```
:ACTION.RELTYPE(ADD)
:RELATIONTYPE.TYPE() EXTNAME() DESCRIPTION()
:RELATIONTYPE.TYPE() EXTNAME() DESCRIPTION()
```

Figure 42. Using the RELATIONTYPE tag when adding relationship types.

```
:ACTION.RELTYPE(DELETE)
:RELATIONTYPE.TYPE()
:RELATIONTYPE.TYPE()
```

Figure 43. Using the RELATIONTYPE tag when deleting relationship types.

Keywords

TYPE
Specifies the short name of a relationship type.

Required keyword.

short_name
Defines and identifies the short name for a specific relationship type.

The value of *short_name* must be unique to an relationship type across all related information catalogs that contain the same relationship type. This ensures that this relationship type can be shared among the related information catalogs. If the value of *short_name* already exists, it is used as a search argument.

The maximum length for the value is 18 characters.

After you create the relationship type, you cannot change the value of *short_name*.

CATEGORY
Specifies the relationship category for the relationship type. Required keyword.

relationship_category
Use `Support`, `Hierarchical`, `Precedence` , or `Peer to Peer`.

EXTNAME
Specifies a longer, descriptive name for the relationship type. Required keyword.

name
Specifies an extended, descriptive name for the relationship type. The maximum length for *name* is 200 characters.

This name must be unique within related information catalogs.

The value of *name* is stored in mixed case.

You can change the information on this keyword after the relationship type is defined.

DESCRIPTION
Specifies a description for the relationship type. Optional keyword.

description
Specifies a description for the relationship type. The maximum length for *description* is 254 characters.

The value of *description* is stored in mixed case.

You can change the information on this keyword after the relationship type is defined.

Related reference:

- "ACTION.RELTYPE" on page 74
- "Tag language" on page 59

RELTYPE

Identifies the type of relationship to add or delete and the object types of the objects involved in the relationship.

This tag is required immediately following these tags:
:ACTION.RELATION(ADD)
:ACTION.RELATION(DELETE)

Syntax

```
:RELTYPE.TYPE(Attachment | Contact | Dictionary |
              Supported | Contains | Input |
              Output | Cascade | Linked |
              user_defined_relationship_type)
        SOURCETYPE(source_type) TARGETTYPE(target_type)
```

Context

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(type_short_name) SOURCETYPE(source_object_type_short_name)
    TARGETTYPE(target_object_type_short_name)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

Figure 44. Using the RELTYPE tag when adding relationships

```
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE(type_short_name) SOURCETYPE(source_object_type_short_name)
    TARGETTYPE(target_object_type_short_name)
:INSTANCE.SOURCEKEY(UI_short_name()...) TARGETKEY(UI_short_name()...)
```

Figure 45. Using the RELTYPE tag when deleting relationships

Keywords

TYPE

Specifies the type of relationship.

Required keyword.

Valid values are:

ATTACHMENT
Attachment relationship: target object is attached to the source object.

CONTACT
Contact relationship: Source object is associated with the target Contact object.

DICTIONARY
Dictionary relationship: Target object is the glossary term object.

SUPPORTED
Supported relationship: Target object is the support object.

CONTAINS
Contains relationship: Source object contains the target object.

INPUT
Input relationship: Source object is the preceding object to a transformation object.

OUTPUT
Input relationship: Target object is the succeeding object to a transformation object.

CASCADE
Cascade relationship: Source object is the preceding object in a lineage.

LINKED
Link relationship: Source object is linked with the target object.

User_defined_relationship_type
Relationship and roles defined by the user.

SOURCETYPE
Identifies the source object type.

Required keyword.

source_type
The source object type name *source_type* corresponds to the *type* value for the TYPE keyword of the OBJECT tag. The maximum length for *source_type* is 18 characters. This value is case sensitive.

For an Attachment relationship, *source_type* is a non-Attachment object-type name.

For a Contains relationship, *source_type* is the container object type name.

For a Contact or link relationship *source_type* is the Grouping or Elemental object type name.

TARGETYPE
Identifies the target object type.

Required keyword.

target_type
The target object type name. *target_type* corresponds to the *type* value for the TYPE keyword on the OBJECT tag. The maximum length for *target_type* is 18 characters. This value is not case sensitive; you can specify this value with uppercase or lowercase characters.

For an Attachment relationship, *target_type* is the Attachment object-type name.

For a Contains relationship, *target_type* is the contained object type name.

For a Contact relationship, *target_type* is the Contact object-type name.

For a link relationship, *target_type* is a Grouping or Elemental object type name.

Related reference:

- "ACTION.RELATION" on page 72
- "Tag language" on page 59

TAB

Specifies a tab within a property value.

The Information Catalog Center reads only TAB tags that are specified within non-UI property values and ignores all others.

Syntax

```
:TAB.
```

Rules

Use TAB tags only within the specification of *property_values* in INSTANCE tags.

Related reference:

- "Tag language" on page 59

Appendix A. Predefined Information Catalog Center object types

This appendix provides a brief description of the Information Catalog Center predefined object types. It also shows how the object types relate to each other using the predefined relationship types.

Information Catalog Center predefined object types

The Information Catalog Center includes predefined object types that you can exchange with metadata from other warehouse components. The following list gives a brief description of each object type.

Application data
The Information Catalog Center uses the Application data object type internally for some data exchanges. Objects of this object type might appear in your information catalog. However, you will not use this object type to create objects.

Attribute
The attribute object type represents an attribute of an entity.

Audio clips
The Audio clips object type represents files that contain audio information. These objects might represent electronic (AUD files) or physical (for example, CDs and tapes) audio information.

Business subject areas
The Business subject areas object type represents logical groupings of objects.

Case models
The Case Models object type represents the logical or physical representation of data such as a table.

Charts The Charts object type represents either hardcopy or electronic charts.

Column mapping
The Column mapping object type represents column mappings in the Data Warehouse Center.

Columns or fields
The Columns or fields object type represents columns within a relational table, fields within a file, or fields within an Internet Management Specification (IMS) segment.

Comments
The Comments object type holds comments about other objects in the information catalog. The Comments object type is created when an information catalog is created.

Databases
The Databases object type represents relational databases.

Dimensions within a multidimensional database
The Dimensions within a multidimensional database object type represents dimensions within a multi-dimensional database. A dimension consists of members.

Documents
The Documents object type represents books, manuals, and technical papers. These publications might be printed or electronic, found locally or within a library.

DWC process
This object type represents a process in the Data Warehouse Center. A process commonly operates on source data and changes data from its original form into a form conducive to decision support. In the Data Warehouse Center, a process commonly consists of one or more sources, one or more steps, and one or more targets.

Elements
The Elements object type represents element objects that do not map directly to the Columns or fields object type.

Entity The Entity object type represents an entity within case model.

Files The Files object type represents a file within a file system.

Glossary entries
The Dictionary category contains the Glossary entries object type. The Glossary entries object type represents definitions for terms that are used in the information catalog.

Images or graphics
The Images or graphics object type represents graphic images, such as bitmaps.

IMS database definitions (DBD)
The IMS database definitions (DBD) object type represents IMS database definitions.

IMS program control block (PCB)
The IMS program control block (PCB) object type represents an IMS program control blocks.

IMS program specification (PSB)
The IMS program specification (PSB) object type represents IMS program specification blocks.

IMS segments
The IMS segments object type represents IMS segments.

Information catalog news
The information catalog news object type conveys information to end users about changes to the information catalog.

Internet documents
The Internet documents object type represents Web sites and other documents on the Internet that might be of interest.

Lotus Approach queries
The Lotus Approach queries object type represents available Lotus Approach queries for use with your organization's data.

Members within a multidimensional database
The Members within a multidimensional database object type represents a

member within a multidimensional database. A member is part of a dimension, and a dimension is part of a multidimensional database.

Multidimensional databases
The Multidimensional databases object type represents multidimensional databases.

OLAP integration server model
The OLAP integration server model object type represents an OLAP Integration Server model. It can be linked to one or more OLAP Multi-dimensional database objects.

Online news services
The Online news services object type represents news services and information services that you can access online.

Online publications
The Online publications object type represents publications and other documents that you can access through online services.

People to contact
The People to contact object type identifies a person or group that is responsible for single or multiple objects within the information catalog.

Presentations
The Presentations object type represents various hardcopy or electronic presentations. These presentations might include product, customer, quality, and status presentations.

Programs
The Program category can contain only the Programs object type. The Programs object type is created when an information catalog is created. It is used to define an application capable of processing a particular object type.

In the sample information catalog the Programs object type is named ″Programs that can be invoked from information catalog objects″ .

Records
The Records object type represents record objects that do not map directly to the Files or Relational tables or views object types. Records consist of Elements.

Relational tables and views
The Relational tables and views object type represents tables or views of relational databases.

Spreadsheets
The Spreadsheets object type represents desktop spreadsheets (for example, Lotus 1-2-3 or Microsoft Excel spreadsheets).

Star Schemas
This object type represents a relational star schema structure. A star schema contains a fact table and one or more dimension tables.

Subschemas
The Subschemas object type represents logical groupings of records within a database.

Text-based reports
The Text-based reports object type represents either hardcopy or electronic reports.

Transformations

The Transformations object type represents expressions or logic that is used to populate columns of data within the target database. Transformations objects indicate either the expression used to convert source-operational data to target columns, or the one-to-one mapping of source fields to target columns.

Video clips

The Video clips object type represents files that contain video information. These objects might represent electronic (AVI files) or physical (for example, videotapes or laser disks) video information.

Related concepts:

- "Object types" on page 7

Related reference:

- "Predefined relationship type models" on page 100
- "Predefined program objects" on page 98

Predefined program objects

The following table shows the predefined program objects included with Information Catalog Center Version 8. The table also shows the property name that you use to associate with the Information Catalog Center program object when launching a program.

Table 9. Generic predefined program objects in the Information Catalog Center

Type of information	Program name	Object type	Property name
Multimedia files	Microsoft Media Player	Audio clips	Audio clip filename
	Microsoft Media Player	Business subject areas	Filename
	Microsoft Media Player	Presentations	Presentation filename
	Microsoft Media Player	Video clips	Video clip filename
Bitmap files	Microsoft Paint	Images or graphics	Graphic filename
	Microsoft Paint	People to contact	Contact's picture filename
Spreadsheet files	Microsoft Excel	Spreadsheets	Spreadsheet filename
	Microsoft Paint	Spreadsheets	Spreadsheet bitmap filename
	Lotus 1-2-3	Spreadsheets	Spreadsheet filename
Web pages	Netscape Navigator	Online news	URL to access data
	Netscape Navigator	Online publications	URL to access data
	Microsoft Internet Explorer	Internet documents	URL to access data
	Microsoft Internet Explorer	Online news	URL to access data
	Microsoft Internet Explorer	Online publications	URL to access data

Table 10 on page 99 lists specific applications that are integrated with the Information Catalog Center. The information in this table similar to that in Table 9.

Table 10. Predefined program objects in sample information catalog — specific applications

Type of information	Program name	Object type	Property name
Lotus	Approach	Lotus Approach	Approach object filename
	Freelance Graphics	Presentations	Presentation object filename
	Lotus 1-2-3 with Essbase Spreadsheet add-in	Spreadsheets	Spreadsheet filename
Hyperion	Microsoft Excel with Essbase Spreadsheet add-in	Spreadsheets	Spreadsheet filename
Brio	Brio Query	Text based reports	Report filename
	Netscape Navigator (use with Brio. Insights plug-in)	Text based reports	URL to access data
	Microsoft Internet Explorer (use with Brio.Insights plug-in)	Text based reports	URL to access data
BusinessObjects	BusinessObjects	Databases	None
	BusinessObjects	Text based reports	Report filename
	Microsoft Excel (used with BusinessQuery add-in)	Spreadsheets	Spreadsheet filename
	Microsoft Internet Explorer (used to access WebIntelligence Java applet)	Internet documents	URL to access data
	Netscape Navigator (used to access WebIntelligence Java applet)	Internet documents	URL to access data
Cognos	PowerPlay	Text-based reports	Report filename
	Impromptu	Text based reports	Report filename
	Microsoft Internet Explorer (used with Impromtu Web Query)	Internet documents	URL to access data
	Netscape Navigator (used with Impromptu Web Query)	Internet documents	URL to access data
	Netscape Navigator (used to access PowerPlay Web edition HTML pages)	Internet documents	URL to access data
Wired for OLAP	Wired for OLAP View	Text based reports	configure Default user login, and Startup options
	Wired for OLAP Home Page within Netscape	Text based reports	configure Default user login, and Startup options
	Wired for OLAP Home Page within Microsoft Internet Explorer	Text based reports	configure Default user login, and Startup options
Seagate	Crystal Reports	Text based reports	Report filename
Microsoft Access	Microsoft Access	Database	
Microsoft PowerPoint™	Microsoft PowerPoint Viewer	Text based reports	Report filename
	Microsoft PowerPoint Viewer within Netscape	Text based reports	URL to access data

Table 10. Predefined program objects in sample information catalog — specific applications (continued)

Type of information	Program name	Object type	Property name
	Microsoft PowerPoint Viewer within Microsoft Internet Explorer	Text based reports	URL to access data

Related concepts:

- "Object types" on page 7

Related reference:

- "Information Catalog Center predefined object types" on page 95

Predefined relationship type models

Information Catalog Center predefined object types follow the data models shown in the following figures. The figures show how the predefined relationship types work with the predefined object types.

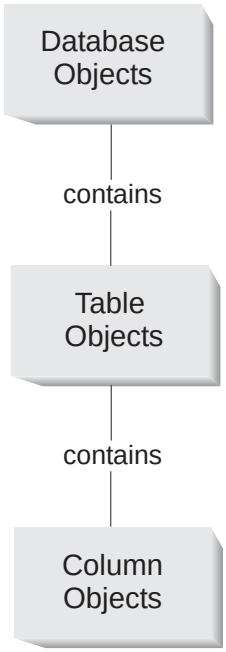

Figure 46. Relational model with the Contains relationship type

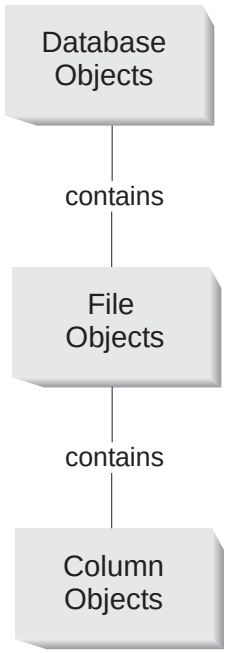

Figure 47. File model with the Contains relationship type

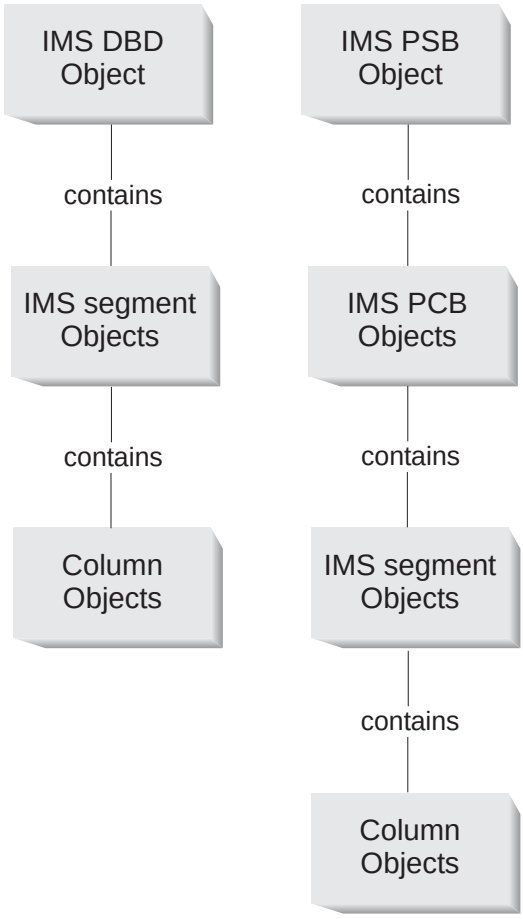

Figure 48. IMS models with the Contains relationship type

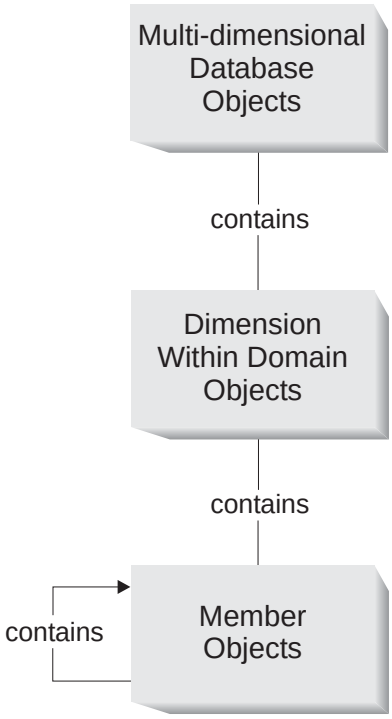

Figure 49. Multi-dimensional model with the Contains relationship type

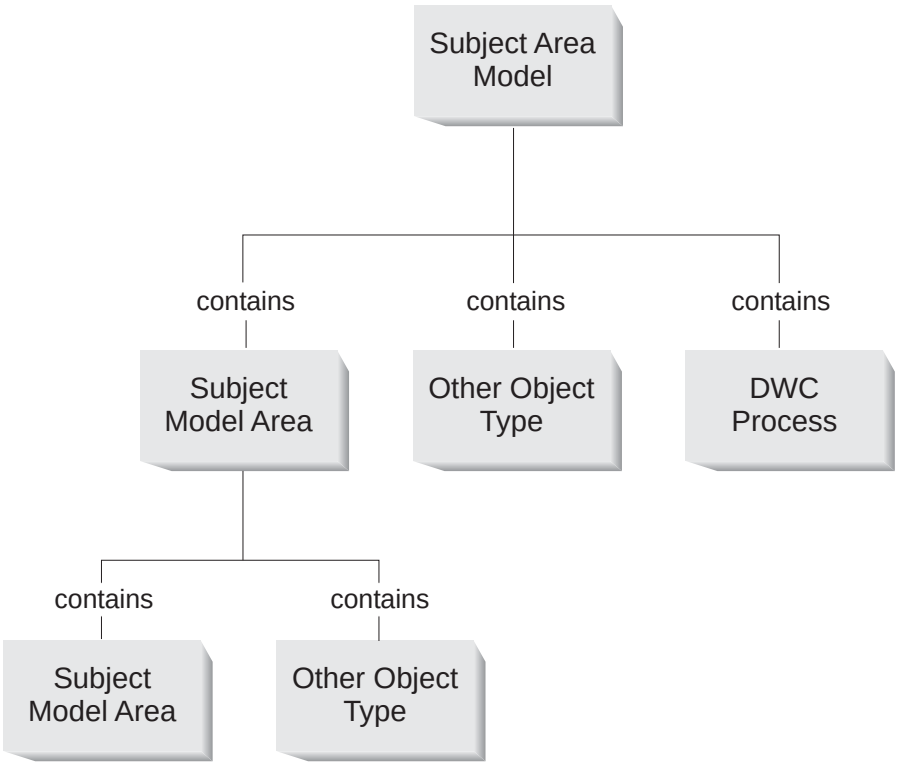

Figure 50. Subject area model with the Contains relationship type

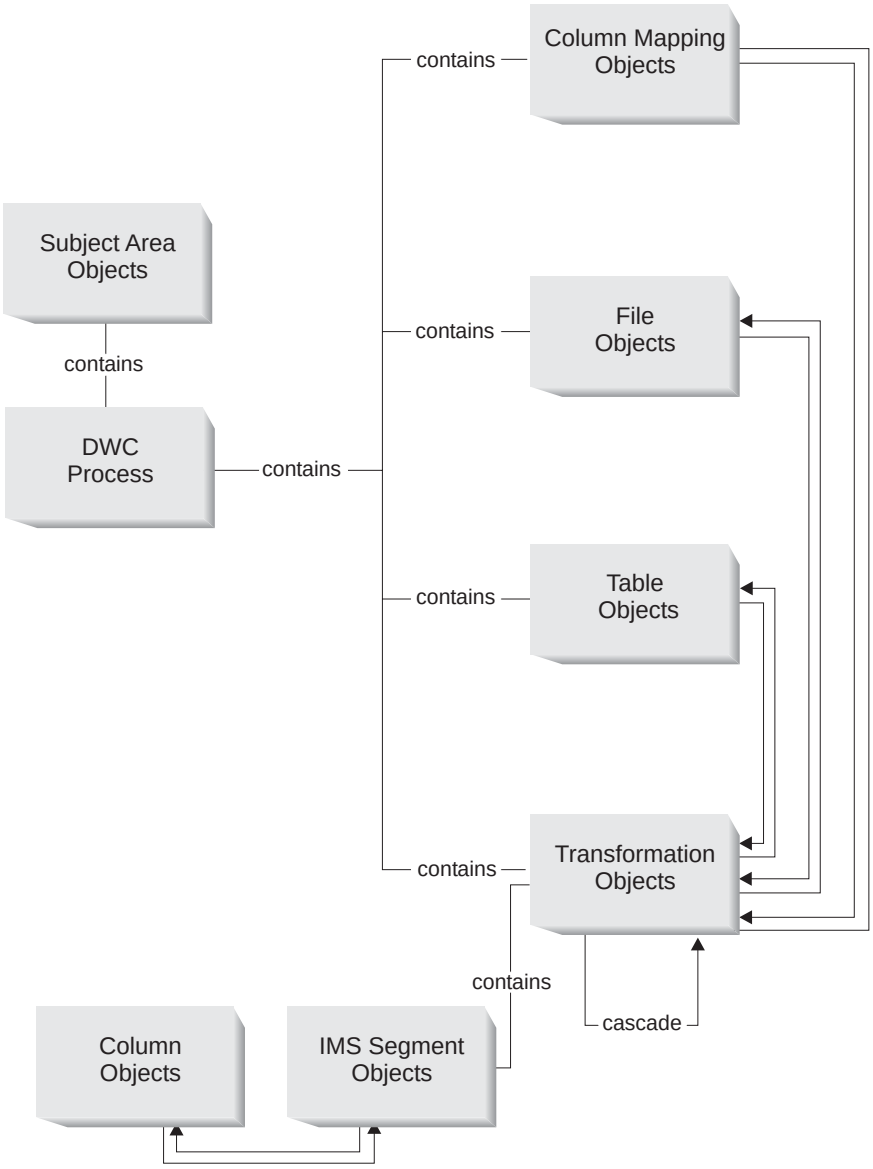

Figure 51. Process and transformation model with the Cascade and Contains relationship types

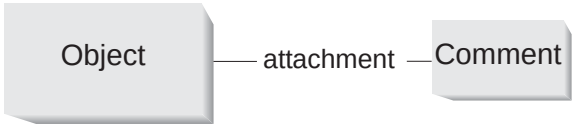

Figure 52. Attachment model with the Attachment relationship type

Figure 53. Contact model with the Contact predefined relationship type

Figure 54. Supported model with the Supported predefined relationship type

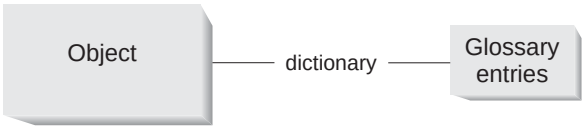

Figure 55. Dictionary model with the Dictionary predefined relationship type

Figure 56. Linked model with the Linked predefined relationship type

Related concepts:

- "Relationship types" on page 23

Related tasks:

- "Adding a relationship between objects" on page 29
- "Removing a relationship between objects" on page 30

Related reference:

- "Mapping Version 7 object type categories to Version 8 relationship types, categories, and roles" on page 123
- "Information Catalog Center predefined object types" on page 95

Appendix B. Metadata mappings

This section lists object types and object type properties for the following metadata:

- Information Catalog Center metadata to Data Warehouse Center metadata.
- Information Catalog Center metadata to OLAP server metadata.
- Information Catalog Center metadata to ERwin Version 4.0 metadata.

Metadata Mappings between the Information Catalog Center and the Data Warehouse Center

The following tables show the metadata mappings between the Information Catalog Center and the Data Warehouse Center for each object type. The Information Catalog Center column shows object type properties as they are displayed in the Description view for an object. The Data Warehouse Center column shows the names of object properties as they are displayed in the various object property notebooks. In some cases, Data Warehouse Center property information is taken from the Work in Progress window.

The metadata mappings shown here are used when publishing data from Data Warehouse Center to the Information Catalog Center.

Table 11. Database objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Warehouse source or Warehouse target name
Short description	Description
Long description	Notes
URL to access data	N/A
Actions	N/A
Database or subsystem name	Database name
Database type	The value for this property can be either `RELATIONAL` or `FILE`. The mapping is derived from the warehouse source or warehouse target type.
Agent type	N/A
Database location	N/A
Database host server name	System name
System code page	This metadata is internal to the Data Warehouse Center.[1]
Database owner	N/A
Timestamp source definition last changed	Last update time stamp for the database definition. This metadata is internal to the Data Warehouse Center.[1]
Database status	N/A

Table 11. Database objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
Database extended type	Database subtype and database version. The mapping is derived from the warehouse source or warehouse target type. For example, if your warehouse target is a DB2 UDB for Windows NT database, the database extended type is DB2 NT.
Timestamp source definition created	N/A
For further information	Administrator

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Table 12. IMS DBD (database description definition) objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Warehouse source name
Short description	Description
Long description	Notes
Actions	N/A
Database last refreshed	N/A
For further information	Administrator
Database owner	N/A
Database host server name	System name
Database server type	Database type and database version. The mapping is derived from the warehouse source type. The property value for IMS DBDs is `IMS`.
Database or subsystem name	Datasource name
Database type	The mapping is derived from the warehouse source type.
Database extended type	Database subtype and database version. The mapping is derived from the warehouse source type. The property value for IMS DBDs is `IMS`.
Database status	N/A
IMS access method	N/A
Operating system access method	N/A
Shared index names	N/A
URL to access data	N/A
Timestamp source definition created	N/A
Timestamp source definition last changed	Last update time stamp for the IMS DBD. This metadata is internal to the Data Warehouse Center.[1]

Table 12. IMS DBD (database description definition) objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
Note:	
1. If a property is specified as internal to the Data Warehouse Center, it is only displayed in the Information Catalog Center.	

Table 13. Relational table or view objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Table name
Short description	Description
Long description	N/A
URL to access data	N/A
Actions	N/A
Catalog remarks	N/A
Local database alias	N/A
Transformation program type	The value for this property is `Data Warehouse Center`. There is no specific metadata in the Data Warehouse Center for this property.
Database or subsystem name	Database name of the warehouse source or warehouse target database that contains the table
Table owner	Table schema
Table name	Table name
Timestamp source definition last changed	Last update time stamp for a table definition. This metadata is internal to the Data Warehouse Center.[1]
Base table owner name	N/A
Base table name	N/A
Transformation program run mode	N/A
Transformation program last run	N/A
Transformation program run frequency	N/A
Partial or full table copy/update	N/A
Copied/updated data is in a consistent state	N/A
Catalog refresh/update frequency	N/A
Transformation program last changed	N/A
Transformation program compiled	N/A
Table type	The mapping is derived from the warehouse source or warehouse target subtype of the database that contains the table. For example, if your warehouse source or target is a DB2 UDB for Windows NT database, the database table type is DB2 NT.
Definition represents a view	N/A

Table 13. Relational table or view objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
Internal IDS name of table	N/A
Table is used as a dimension table	Dimension table
Database host server name	System name of the warehouse source or warehouse target database that contains the table.
Time stamp source definition created	N/A
For further information	Administrator of the warehouse source or warehouse target database that contains the table.

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Table 14. Column or field objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Column or field name
Short description	Description
Long description	N/A
URL to access data	N/A
Actions	N/A
Catalog remarks	N/A
Datatype of column or field	Data type
Position of column or field in the primary key	N/A
Length of the column or field	Length or precision (depending on the data type)
Scale of the column or field	Scale
Can column or field be null	Nullable
Column or field position	Position in the list of columns or fields displayed in the table or file notebook for a warehouse source or warehouse target.
Database or subsystem name	Database name of the warehouse source or warehouse target that contains the table that contains the column.
Table owner	Table schema of the table that contains the column.
Table name	Name of the table that contains the column.
Containing dimension	N/A
Column or field name	Column name
File name	File name of the file that contains the field (Data Warehouse Center files only)
Byte offset of column or field from start	Offset of this field in a file of fixed type. This metadata is internal to the Data Warehouse Center.[1]
Is column or field part of a key	N/A

Table 14. Column or field objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
Is column or field a unique key	N/A
Is data a before or after image, or computed	N/A
Source column/field name or expression used to populate column	N/A
Timestamp source definition last changed	Last update time stamp for the column definition. This metadata is internal to the Data Warehouse Center.[1]
String used to represent null values	N/A
Resolutions of dates	N/A
Precision of column or field	N/A
Is data text	Is text The value for this property is `Y` or `N`.
Database host server name	System name of the database that contains the table that contains the column.
Column or field last refreshed	N/A
Timestamp source definition created	N/A
For further information	Administrator for the database that contains the table that contains the column.
Column ordinality	N/A

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Table 15. Column mapping objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	*inputColumnname—>outputColumnName*
Input column name	Physical input column name
Output column name	Physical output column name
Short description	N/A
Long description	N/A
Transformation name	Transformation Name
Type	DWC map type
Sequence number	DWC sequence number, generate if not available
Expression	Expression used for column mapping

Table 16. File objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	The value for this property is derived from the file name.
Short description	Description
Long description	N/A

Table 16. File objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
URL to access data	N/A
Actions	N/A
Transformation program type	The value for this property is `Data Warehouse Center`. There is no specific metadata in the Data Warehouse Center for this property.
Database host server name	System name of the warehouse source or warehouse target that contains the file.
Database or subsystem name	Database name of the warehouse source or warehouse target that contains the file.
File owner	N/A
File path or directory	The property value for the file path or directory is derived from the file name.
File name	The property value is derived from the file name.
File class or type	File type
Source definition last changed	Last update time stamp for a file definition. This metadata is internal to the Data Warehouse Center.[1]
Transformation program last run	N/A
Transformation program run frequency	N/A
Partial or full file copy/update	N/A
Copied/updated data is in a consistent state	N/A
Transformation program last changed	N/A
Transformation program last compiled	N/A
Timestamp source definition created	N/A
For further information	Administrator of the warehouse source or warehouse target that contains the file.

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Table 17. IMS segment objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Table name
Short description	Description
Long description	N/A
URL to access data	N/A
Actions	N/A
Database or subsystem name	Datasource name
Segment name	N/A
Segment maximum length	N/A
Segment minimum length	N/A

Table 17. IMS segment objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
Real logical child segment source	N/A
Logical parent concatenated key source	N/A
Transformation program last run	N/A
Transformation program run frequency	N/A
Timestamp source definition last changed	Last update time stamp for a segment definition. This metadata is internal to the Data Warehouse Center.[1]
Database host server name	System name for the IMS database definition (DBD)
Segment owner	N/A
Segment last refreshed	N/A
Timestamp source definition created	N/A
For further information	Administrator for the IMS DBD that contains the segment.

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Table 18. Transformation objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Step name
Short description	Description
Long description	N/A
URL to access data	N/A
Actions	N/A
Transformation Identifier	Unique identifier for the transformation. This metadata is internal to the Data Warehouse Center.[1]
Transformation program name	Program name
Transformation class or type	Program type
Source column/field name, expression or parameters	For SQL steps, the value of this property is `SQL statement`. For non-SQL steps, the value is the concatenation of any Parameter values for the step.
Source definition last changed	Last update time stamp for step definition. This metadata is internal to the Data Warehouse Center.[1]
Database host server name	Target database system name
Transformation owner	N/A
Source sequence	N/A
Transformation ordinality	N/A
Transformation bidirectionality	N/A

Table 18. Transformation objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
Timestamp source definition created	N/A
For further information	Administrator

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Table 19. Business subject area objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Subject area name
Short description	Description
Long description	Notes
Actions	N/A
Data refresh frequency	N/A
URL to access data	N/A
Filename	N/A
For further information	Administrator

Table 20. Star schema objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Warehouse schema name
Short description	Description
Long description	Notes
Actions	N/A
For further information	Administrator
URL to access data	N/A
Timestamp source definition last changed	Last update time stamp for warehouse schema definition. This metadata is internal to the Data Warehouse Center.[1]

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Table 21. Process objects

Information Catalog Center metadata	Data Warehouse Center metadata
Name	Process name
Short description	Description
Long description	Process notes
Actions	N/A
For further information	Administrator
URL to access data	N/A

Table 21. Process objects (continued)

Information Catalog Center metadata	Data Warehouse Center metadata
Timestamp source definition last changed	Last update time stamp for the process definition. This metadata is internal to the Data Warehouse Center.[1]

Note:

1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.

Related concepts:

- "How Data Warehouse Center metadata is displayed in the information catalog" on page 52

Related tasks:

- "Preparing to publish Data Warehouse Center metadata" on page 51

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" in the *Data Warehouse Center Application Integration Guide*
- "Metadata mappings between the Information Catalog Center and OLAP server" on page 113

Metadata mappings between the Information Catalog Center and OLAP server

Table 22 shows the mapping of OLAP server metadata to the Information Catalog Center common object types. OLAP server metadata refers to metadata for DB2 OLAP Server, or Hyperion Essbase Server. The metadata mappings shown here are used when publishing data from the OLAP server to the Information Catalog Center.

The left column of the table shows the name of the Essbase API structure. The right column shows the Information Catalog Center object and object type properties.

Table 22. Common object types

OLAP server metadata	Information Catalog Center metadata
Outline	*Multidimensional databases*
Four part name of the OLAP object in the following format: server.application.database.outline	Name
Message indicating width and depth limits	Long description
OLAP server (part 1 of Name)	Database host server name
OLAP database (part 3 of Name)	Database or subsystem name
N/A	Database type The value of this property is MULTIDIMENSIONAL.

Table 22. Common object types (continued)

OLAP server metadata	Information Catalog Center metadata
usOutlineType in ESB_OUTLINEINFO_T	Database extended type The value for this property can be NORMAL or CURRENCY.
N/A	Database status The value for this property is PRODUCTION.
Dimensions in an outline	*Dimension within a multidimensional database*
Dimension alias from EssOtlGetMemberAlias or name	Name
OLAP server	Database host server name
OLAP database	Database or subsystem name
OLAP application	Using application name
Dimension name	Dimension name
usCategory in ESS_MBRINFO_T	Dimension class or type
Members in a dimension	*Members within a multidimensional database*
Member alias from EssOtlGetMemberAlias or name	Name
OLAP server	Database host server name
OLAP database	Database or subsystem name
OLAP application	Using application name
Dimension name	Dimension name
Member name	Member name
last calc string or calc string from EssGetMemberCalc	Derived from
usShare in ESS_MBRINFO_T	This property is treated as a shared member (a member with multiple parents).

Related tasks:

- "Preparing to publish OLAP server metadata" on page 50

Related reference:

- "Metadata Mappings between the Information Catalog Center and the Data Warehouse Center" on page 105

Metadata mappings between ERwin Version 4.0 object attributes and Information Catalog Center properties

The following tables show how ERwin Version 4.0 object attributes in an XML file correspond to Information Catalog Center properties.

Table 23. Information catalog physical model for mappings.

XML tag	Information Catalog Center property
ModelProps.Name	NAME
ModelProps.Definition	SHRTDESC

Table 23. Information catalog physical model for mappings. (continued)

XML tag	Information Catalog Center property
Not applicable	LONGDESC
ModelProps.Author	RESPNSBL
Not applicable	SERVER (unique identifier)
ModelProps.Name	DBNAME (unique identifier)
″RELATIONAL″	DBTYPE (unique identifier)
ModelProps.Target_Server	DBETYPE

Table 24. Information catalog logical model mappings.

XML tag	Information Catalog Center property
ModelProps.Name	NAME (unique identifier)
ModelProps.Definition	SHRTDESC
Not applicable	LONGDESC
ModelProps.Author	RESPNSBL
Not applicable	URL
ModelProps.File_Name	FILENAME

Table 25. Information catalog physical entity mappings.

XML tag	Information Catalog Center property
EntityProps.Name	NAME
EntityProps.Definition	SHRTDESC
Not applicable	LONGDESC
EntityProps.Comment	REMARKS
ModelProps.Author	RESPNSBL
Not applicable	SERVER
ModelProps.Name	DBNAME (unique identifier)
EntityProps.DB_Owner	OWNER (unique identifier)
EntityProps.Physical_Name	TABLES (unique identifier)

Table 26. Information catalog logical entity mappings

XML tag	Information Catalog Center property
EntityProps.Name	NAME (unique identifier)
EntityProps.Definition	SHRTDESC
EntityProps.Note + Note_2 + Note_3	LONGDESC
ModelProps.Author	RESPNSBL
ModelProps.Name	MODLNAME (unique identifier)

Table 27. Information catalog logical attribute mappings

XML tag	Information Catalog Center property
AttributeProps.Name	NAME (unique identifier)
AttributeProps.Definition	SHRTDESC

Table 27. Information catalog logical attribute mappings (continued)

XML tag	Information Catalog Center property
AttributeProps.Note	LONGDESC
ModelProps.Author	RESPNSBL
AttributeProps.Logical_Datatype	DATATYPE CHAR(30)
Determined from Datatype	LENGTH CHAR(20)
ModelProps.Name	MODLNAME (unique identifier)
EntityProps.Physical_Name	ENTYNAME (unique identifier)

Table 28. Information catalog physical column mappings.

XML tag	Information Catalog Center property
AttributeProps.Physical_Name	NAME (unique identifier)
AttributeProps.Definition	SHRTDESC
Not applicable	LONGDESC
Not applicable	REMARKS
ModelProps.Author	RESPNSBL
AttributeProps.Datatype	DATATYPE CHAR(30) [1]
Determined from Datatype	LENGTH CHAR(20) [2]
Determined from Datatype	SCALE CHAR(5)
Determined from Datatype	PRECDIG CHAR(5)
AttributeProps.Attribute_Required or AttributeProps.Null_Option	NULLS CHAR(1)
AttributeProps.Order - 1	ORDINAL CHAR(5)
AttributeProps.Order	POSNO CHAR(5)
NULL	STARTPOS CHAR(10)
See KEYPOSNO	ISKEY CHAR(1)
NULL	UNIQKEY CHAR(1)
Entity.Key_Group_Groups.Key_Group. Key_group_member_Groups	KEYPOSNO CHAR(5)
Not applicable	SERVER
ModelProps.Name	DBNAME (unique identifier)
EntityProps.DB_Owner	OWNER (unique identifier)
EntityProps.Name	TABLES (unique identifier)
AttributeProps.Name	COLUMNS (unique identifier)
Not applicable	FILENAME (unique identifier)
Determined from Datatype	ISTEXT CHAR(1) [3]

1. Datatype is determined by search order. If a value cannot be found, the value of the XML element is put into the information catalog property. The following search order is used:
   a. AttributeProps.Datatype
   b. DomainProps.Datatype
   c. ModelProps.Default_Datatype

2. LENGTH is determined by searching for a value in Datatype that is inside of parentheses. If a numeric value is found, LENGTH is set to that value. If none is found, LENGTH is set to NULL. If a string between the parentheses contains a comma separating 2 numeric values, the first value is the precision (PRECDIG), and the second numeric is the scale (SCALE).
3. ISTEXT is determined from Datatype. If Datatype is a CHAR or GRAPHIC type, ISTEXT is set to Y, otherwise it is set to F.

Appendix C. Performing Information Catalog Center tasks from the command line

This appendix describes the Information Catalog Center functions that you can perform from a command line, including:

- Creating an information catalog
- Importing tag language files

Preparing an information catalog from a command line

Use the **db2icminit** command to create an information catalog from a command line. You can create the information catalog in a DB2 Tools Catalog or in another DB2 database. Preparing an information catalog from a command line instead of the user interface allows you to create a catalog and populate with a batch job. When you create the information catalog, specify the **def** option to create the predefined Information Catalog Center object types.

Prerequisites:

Before you prepare an information catalog:

- The Information Catalog Center must be installed and configured.
- The DB2 Warehouse Manager must be installed.
- There must be a DB2 Universal Database cataloged on the local workstation.

Restrictions:

You must be a database administrator for DB2 Universal Database.

Procedure:

Decide if you are creating the information catalog in DB2 tools catalog or in another database.

- If you are creating an information catalog in a DB2 tools catalog, type the following command at a command prompt.

  ```
  db2icminit -db database[-s schema] [-u user -p password]
  -app {skip | merge | replace} [def]
  ```

- If you are creating an information catalog in another database, type the following command at a command prompt:

  ```
  db2icminit -db database[-s schema] [-u user -p password]
  -api create -app {skip | merge | replace} [def]
  ```

database
Specifies the database name

schema
Optional. Specifies the schema that holds the ICM catalog. If *schema* is not specified, the default schema, ICM, is used.

user
Optional. Specifies the user ID required by the database that stores your information catalog. If *user* is not specified, the current operating system user is used for authentication.

password
Specifies the password for *user*. Specify *password* only when *user* is specified.

-api
Specifies that you are initializing a metadata store

create. Creates a new metadata store.

-app
Specifies that you are initializing the ICM application

skip. Does not create new object types if existing object types have the same name.

merge. Attempts to modify existing object types that have the same name as new object types so that both object types have compatible definitions

replace. If new object types have the same name as existing object types, the existing object types are replaced by the new ones.

def
Optional. Creates the default predefined object types.

Related tasks:

- "Preparing an information catalog" on page 3

Importing tag language files from the command line

Use the db2icmimport command from a command prompt to import a tag language file into your information catalog.

Prerequisites:

Before you import a tag language file:

- You must have created an information catalog
- You must have created a tag language file

Restrictions:

You must be a database administrator for DB2 Universal Database.

Procedure:

To import a tag language file from the command line, type the following command at a command prompt.

```
db2icmimport userid userid password password database database catalog catalog
tagfile tagfile iconpath:iconpath logfile:logfile
restart:checkpoint trace:YES|NO
```

The first 5 values of the command are positional and the last 4 are optional and not positional. Fully qualified path names are required for all references to files.

database
The name of the database that contains your information catalog.

catalog
The name of the schema that contains your information catalog.

userid
Specifies the user ID required by the database that stores your information catalog.

password
The password associated with *userid*.

tagfile
The directory path and filename of the tag language file that you are importing.

logfile
The directory path and filename of the file where messages, errors, and trace information are logged. If you do not specify a value, the log file is saved to the same directory as the tag language file.

iconpath
Optional. Use only if you are importing icons. Specifies the icon path the import function uses.

checkpoint
Indicates that the import function will restart from the beginning of the tag file. If this parameter is not specified, the import function restarts from the last successful database commit.

Trace
The level of trace information to send to the Information Catalog Center trace file. Trace is on if this keyword is specified. Trace is off if this keyword is not specified. The trace file can be found in the logging directory.

Related tasks:

- "Importing tag language files" on page 40

Related reference:

- "Metadata import from a tag language file" in the *Data Warehouse Center Application Integration Guide*
- "Metadata export from the Information Catalog Manager" in the *Data Warehouse Center Application Integration Guide*
- "Tag language" on page 59

Appendix D. Version 7 compatibility

This appendix shows how the Versioin 7 object type categories relate to the Version 8 relationship types, categories, and roles.

Mapping Version 7 object type categories to Version 8 relationship types, categories, and roles

The Information Catalog Center does not use the Information Catalog Manager Version 7 object type categories. Information Catalog Center replaces the object type categories with relationship types, categories and roles. The following table maps the Version 7 object type categories to the Version 8 relationship types, categories, and roles.

Table 29.

Version 7 object type category	Version 8 relationship type, relationship category, and relationship role
Elemental	Contains, Hierarchical, child Contact, t, object Attachment, Support, object Linked, Peer to Peer, object
Grouping	Contains, Hierarchical, parent Contains, Hierarchical, child, Contact, , object Attachment, Support, object Linked, Peer to Peer, object Input, Precedence, data resource Output, Precedence, data resource Input, Precedence, operation Output, Precedence, operation
Program	none
Contact	
Dictionary	
Support	
Attachment	Attachment, Support, support object

Related concepts:

- "Relationship types" on page 23

Related reference:

- "Predefined relationship type models" on page 100

Glossary

A

attachment relationship type. The relationship type that is used to attach comments to other objects. Comments can contain additional information about the object that they are attached to. See also *relationship type*.

B

BLOB. See *binary large object*.

binary large object (BLOB). A sequence of bytes with a size that can range from 0 bytes to 2 gigabytes, less 1 byte. This string does not have an associated code page and character set. BLOBs can contain image, audio, and video data. See also *character large object*.

browse. To view information catalog objects that are grouped by subject. Contrast with *search*.

business metadata. Data that describes information assets in business terms. Business metadata is stored in the information catalog and accessed by users to find and understand the information that they need. For example, business metadata for a program might contain a description of what the program does and which tables it uses. See also *technical metadata* and *metadata*.

C

catalog. See *information catalog*, *database catalog*, and *RDBMS catalog*.

CelDial sample catalog. A sample information catalog (ICCSAMP) that is available when you install the Information Catalog Center that can be used for installation verification. An administrator initializes the catalog, and users can use the sample data to become familiar with the Information Catalog Center.

character large object (CLOB). A sequence of characters (single-byte, multibyte, or both) with a size that can range from 0 bytes to 2 gigabytes, less 1 byte. In general, character large object values are used whenever a character string might exceed the limits of the VARCHAR data type. See also *binary large object*.

CLOB. See *character large object*.

collection. A container for objects. A collection can be used to gather objects that the user has privileges to see, similar to a personal folder of objects.

comments object type. An object type that annotates another object in the Information Catalog Center. For example, you can attach a comment to a chart object that contains notes about the data in the chart.

The comments object type is predefined in the Information Catalog Center. You cannot add properties to it.

commit. The operation that ends a unit of work by releasing locks so that the database changes made by that unit of work can be perceived by other processes. This operation makes the data changes permanent.

contact relationship type. The relationship type that is used to identify contacts. A contact relationship type provides more information about an object. Such information might include the person who created the information that the object represents or the department that is responsible for maintaining the information. See also *relationship type*.

contains relationship type. The relationship type that is used to identify Information Catalog Center objects that contain other objects. For example, use the contains relationship type to denote an object with a "parent" role, meaning that object can contain other objects. You can also use the contains relationship type to denote an object with a "child" role, meaning an object that can be contained in another object. See also *relationship type*.

D

database catalog. A collection of tables that contains descriptions of database objects such as tables, views, and indexes.

DBCS. See *double-byte character set*.

decision-support system. A system of applications that help users make decisions. This kind of system allows users to work with information that is presented in meaningful ways; for example, spreadsheets, charts, and reports.

delete history. A log of delete activity, the capture of which is turned on and off by the Information Catalog Center administrator. The log can be transferred to a tag language file.

derived data. Data that is copied or enhanced (perhaps by summarizing the data) from operational data sources into an informational database.

descriptive data. See *metadata*.

dictionary relationship type. The relationship type that is used to associate a glossary entry object type with another object. A glossary entry object type can be used to define terminology that is associated with the object. See also *relationship type*.

double-byte character set (DBCS). A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. See also *single-byte character set* and *multibyte character set*.

E

extract control file. A file that contains statements that control the operation of an extractor utility program.

extract program. A utility program that copies metadata from a metadata source (such as an *RDBMS catalog*) translates the metadata into tag language, and places this output in a tag language file.

F

FAT. File allocation table. A table that is used to allocate space on a disk for a file and to locate the file.

H

hierarchical relationship category. A category of relationship types that are used to connect objects that have a hierarchical relationship.

I

import. To read the contents of a tag language file to initially populate the information catalog, change the information catalog contents, or copy the contents of another information catalog.

information catalog. A set of tables that contain descriptive data (business metadata) that helps users identify and locate information that is available to them. An information catalog also includes some technical metadata.

Information Catalog Center application program interface (API). A set of Java classes that can be used to write programs that read, create, and update the metadata that is stored in the information catalog.

information source. An item of data or information, such as a table or chart, that is represented by an Information Catalog Center object.

input relationship type. A relationship type that is used to connect objects that transform to their input data resource. See *transformation relationship category*. See also *relationship type*.

instance. See *object*.

K

keyword. An element of the Information Catalog Center tag language that identifies the meaning of a data value that is imported into an information catalog.

keyword search. See *search*.

L

linked relationship type. A relationship type that is used to connect two or more objects in an information catalog. Objects in a linked relationship are peers, rather than a parent-child relationship.

For example, in the sample information catalog that is included with the Information Catalog Center, the object called **CelDial Sales Information** is linked with objects that describe CelDial advertisements for the year. See also *relationship type*.

log file. A file that is produced by the Information Catalog Center when it imports a tag language file into the information catalog. This file records the times and dates when the import process started and stopped and any error information for the process.

M

metadata. Data that describes the characteristics of stored data; descriptive data. For example, the metadata for a database table might include the name of the table, the name of the database that contains the table, the names of the columns in the table, and the column descriptions, either in technical terms or business terms. Database catalogs and information catalogs contain metadata.

multibyte character set (MBCS). A set of characters in which each character is represented by 2 or more bytes. A character set that uses only two bytes are more commonly known as a double-byte character set. See also *single-byte character set*.

O

object. An item that represents a unit or distinct grouping of information. Each Information Catalog Center object identifies and describes information but does not contain the actual information. For example, an object can provide the name of a report, list its creation date, and describe its purpose.

object type. A classification for objects. An object type is used to reflect a type of business information, such as a table, report, or image.

The Information Catalog Center provides a set of default object types. You can also create additional object types to meet the needs of your organization.

operational data. Data that is used to run the day-to-day operations of an organization.

option. In the Information Catalog Center tag language, a parameter of the ACTION tag that defines the action to be performed on objects or object types in the information catalog when the tag language file is imported.

output relationship type. A relationship type that is used to connect objects that transform to their output data resource. See *transformation relationship category*. See also *relationship type*.

P

peer to peer relationship category. A category of relationship types that are used to connect object that have a peer relationship.

populate. To add object types, objects, or metadata to the Information Catalog Center.

privileges. The right to access a specific database object in a specific way. These rights are controlled by users with SYSADM (system administrator) authority or DBADM (database administrator) authority or by creators of objects. Privileges include creating, updating, and deleting objects from the information catalog.

Three default classifications of users are included with Information Catalog Center. These classifications are user, power user, and administrator. Their privileges can vary by object, but, in general, users have read-only rights to specific objects that an administrator has granted to them. Power users have the same privileges as a user. They can also be granted privileges to create objects and update objects that they create. Administrators have all privileges to all objects.

programs object type. An object type that identifies and describes applications capable of processing the actual information that is described by Information Catalog Center objects.

The programs object type is included with the Information Catalog Center. Administrators specify which programs can be used to access certain object types.

property. A characteristic or attribute that describes a unit of information. Each object type has a set of associated properties. For example, the "Graphics and Images" object type in the sample information catalog includes the following properties:

- Name
- Description
- Image type
- Image filename

For each object, a set of values is assigned to the properties. Administrators can specify what type of values are permitted for each property.

property display name. A 254 character name that is used by the Information Catalog Center to display the name of a property in the Properties window.

property name. The 254-byte descriptive name of a property that is displayed in the Information Catalog Center user interface.

R

RDBMS. See *relational database management system*.

RDBMS catalog. A collection of tables that contains descriptions of SQL objects, such as tables, views, and indexes, maintained by an RDBMS.

relational database. A database that can be perceived as a set of tables and manipulated in accordance with the relational model of data. Each database includes a set of system catalog tables that describe the logical and physical structure of the data, a configuration file that contains the parameter values allocated for the database, and a recovery log with ongoing transactions and archivable transactions.

relational database management system (RDBMS). A software system, such as DB2 Universal Database for Windows, that manages and stores relational data.

relationship category. A basis to define the relationship type. There are four relationship categories:

- Support
- Hierarchical
- Transformational
- Peer to Peer

Each of these relationship categories has roles associated with it that define how an object can relate to other objects. For example, the support relationship category has "object" and "support object" roles available.

relationship type. A definition that defines the roles an object type can play in a relationship. The default relationship types are:

- Attachment
- Contact
- Contains
- Dictionary
- Input

- Output
- Linked
- Supported

Each default relationship has a specific set of roles that object types can play. For example, the contains relationship type allows parent and child roles. If you added a contains relationship between two objects, one object takes on the ″parent″ role and the other object takes on the ″child″ role.

role. A descriptor that is associated with the relationship category. The relationship category that you choose determines what roles are available for each object type.

S

saved search. A set of search criteria that is saved for subsequent use. A saved search is displayed as an object under the **Saved Searches** folder in the tree.

SBCS. See *single-byte character set*.

search. In the Information Catalog Center, to request the display of the objects that meet user-specified criteria.

search criteria. Options and character strings that are used to specify how to perform a search. The search criteria can include object type names, property values, whether the search is for an exact match, and whether the search is case sensitive.

single-byte character set (SBCS). A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set*.

subject area. An object type that identifies and groups the processes that relate to a logical area of the business. For example, if you are creating an information catalog of marketing and sales data, you define object types Sales and Marketing and select to make them subject areas. Then any objects of type Sales or Marketing are grouped under the corresponding subject.

subject search. See *browse*.

support relationship category. A category for relationship types that connects supporting objects to another object (for example, you can connect a News object to a Spreadsheet object).

support relationship type. A category of relationship types that provides additional information about your information catalog or enterprise (for example, the “Information Catalog Center News” object type in the sample information catalog). See also *relationship type*.

T

tag. An element of the tag language. Tags indicate actions to be taken when the tag language file is imported to the information catalog.

tag language. A format for defining object types and objects, and actions to be taken on those object types and objects, in the Data Warehouse Center or the information catalog.

tag language file. A file that contains tag language that describes objects and object types to be added, updated or deleted in the Data Warehouse Center or in the information catalog, when the file is imported.

In the Information Catalog Center, a tag language file is produced when you:

- Transfer a delete history log.
- Extract descriptive data from another database system using an extract program.

technical metadata. Data that describes the technical aspects of the data, such as its database type and length. Technical metadata includes information about where the data comes from and the rules that are used to extract, clean, and transform the data. See also *business metadata*.

transformation relationship category. A category for relationship types that connects transformation objects to data resources (for example, you can connect a Transformation object to a File object). Objects that are connected with this category of relationship are displayed in the Information Catalog Center Show Lineage Tree window.

Tree view. A view that provides a hierarchical view of an object and the objects that it contains.

U

unit of work. A recoverable sequence of operations within an application process. At any time, an application process is a single unit of work, but the life of an application process can involve many units of work as a result of commit or rollback operations.

unique identifier (UI). A key for an object. The key is comprised of up to 16 properties, which, when concatenated in a designated order, uniquely identify the object during the import function.

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.

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

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

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

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

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

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

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

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

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 measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

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

COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both, and have been used in at least one of the documents in the DB2 UDB documentation library.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 UDB documentation library:

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

Intel and Pentium are 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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-888-426-4343 to learn about available service options
- 1-800-IBM-4YOU (426-4968) for DB2 marketing and sales

In Canada, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-800-465-9600 to learn about available service options
- 1-800-IBM-4YOU (1-800-426-4968) for DB2 marketing and sales

To locate an IBM office in your country or region, check IBM's Directory of Worldwide Contacts on the web at http://www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at http://www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, product downloads, newsgroups, FixPaks, news, and links to web resources.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at www.ibm.com/planetwide

Index

J

L

M

N

O

P

R

S

T

U

V

W

Printed in USA

SC27-1125-01

Spine information:

IBM® DB2® Warehouse Manager
Standard Edition

Information Catalog Center Administration Guide

Version 8.2