

IBM® DB2® Warehouse Manager



# Information Catalog Manager Administration Guide

*Version 7*



IBM® DB2® Warehouse Manager



# Information Catalog Manager Administration Guide

*Version 7*

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 195.

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

Order publications through your IBM representative or the IBM branch office serving your locality or by calling 1-800-879-2755 in the United States or 1-800-IBM-4YOU in Canada.

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.

© **Copyright International Business Machines Corporation 1994, 2000. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About this book</b> . . . . .	<b>vii</b>	Information Catalog Manager categories and object types . . . . .	25
Who should use this book . . . . .	vii	Creating your own object types . . . . .	28
How to use this book . . . . .	vii	Notes for creating object types for the Information Catalog Manager for the Web . . . . .	28
Administrator task information . . . . .	vii	Creating an object type by using the Information Catalog Manager windows . . . . .	29
Performing Information Catalog Manager tasks with the user interface or tag language . . . . .	ix	Creating an object type by using the Information Catalog Manager tag language . . . . .	35
How to send your comments. . . . .	x	Updating an object type . . . . .	38
		Updating an object type by using the Information Catalog Manager windows . . . . .	38
		Updating an object type using Information Catalog Manager tag language . . . . .	39
		Deleting an object type . . . . .	41
		Deleting an object type by using the Information Catalog Manager windows . . . . .	41
		Deleting an object type by using the Information Catalog Manager tag language . . . . .	41
<b>Chapter 1. Setting up an information catalog</b> 1			
Authorizing users to the Information Catalog Manager . . . . .	2	<b>Chapter 3. Populating the catalog with information</b> . . . . .	<b>43</b>
Creating the information catalog . . . . .	3	Creating an object . . . . .	43
Defining an information catalog on DB2 Universal Database for OS/2 . . . . .	4	Creating an object by using the Information Catalog Manager windows . . . . .	43
Defining an information catalog on DB2 Universal Database for OS/390 . . . . .	5	Creating an object using the Information Catalog Manager tag language . . . . .	44
Defining an information catalog on DB2 Universal Database for AS/400 . . . . .	7	Copying an object . . . . .	45
Defining an information catalog on UNIX® systems . . . . .	9	Updating an object. . . . .	46
Defining an information catalog on DB2 Universal Database for Windows NT or DB2 Universal Database for Windows 2000. . . . .	11	Updating an object by using the Information Catalog Manager windows . . . . .	46
Migrating a DataGuide Version 5.2 information catalog . . . . .	14	Updating an object by using the Information Catalog Manager tag language . . . . .	47
Registering a server node and information catalog . . . . .	14	Deleting an object . . . . .	48
Registering a server node using the DB2 Control Center . . . . .	14	Deleting an object by using the Information Catalog Manager windows . . . . .	48
Registering a server node with the Information Catalog Manager . . . . .	15	Deleting an object by using the Information Catalog Manager tag language . . . . .	49
Registering a remote information catalog . . . . .	17		
Opening an information catalog . . . . .	18	<b>Chapter 4. Making the information catalog convenient for users</b> . . . . .	<b>51</b>
Ensuring that users can start programs from the Information Catalog Manager . . . . .	19	Grouping objects by subject . . . . .	51
Additional requirements for Information Catalog Manager for the Web users . . . . .	19	Grouping objects by subject by using the Information Catalog Manager windows . . . . .	51
Authorizing Information Catalog Manager users to manage objects . . . . .	21		
Setting and changing status values for comments. . . . .	22		
<b>Chapter 2. Organizing your information resources</b> . . . . .	<b>25</b>		

Grouping objects by subject by using tag language . . . . .	53
Creating a linked relationship between objects	54
Linking objects by using the Information Catalog Manager windows . . . . .	54
Linking objects by using tag language . . . . .	55
Associating contact names with objects . . . . .	56
Associating contact names with objects by using the Information Catalog Manager windows . . . . .	56
Associating contact names with objects by using tag language. . . . .	57
Associating comments and objects . . . . .	58
Creating a comment . . . . .	58
Copying a comment . . . . .	59
Updating a comment . . . . .	59
Deleting a comment . . . . .	60
Attaching and detaching comments and objects . . . . .	61
Associating a program with object types . . . . .	61
Creating a Programs object . . . . .	62
Copying a program that is associated with an object type . . . . .	66
Updating a program association for an object type . . . . .	66
Disassociating a program from an object type. . . . .	67
Creating a dictionary facility . . . . .	68
Creating a support facility . . . . .	68

**Chapter 5. Expanding and automating your information catalog . . . . . 71**

Extracting descriptive data from other sources	72
Extracting descriptive data with the information catalog extract programs . . . . .	72
Writing customized descriptive data extract programs . . . . .	72
Logging deletions from your information catalog. . . . .	77
Importing and exporting tag language files	78
Importing tag language files . . . . .	78
Exporting metadata . . . . .	79
Solving import and export problems . . . . .	82

**Chapter 6. Exchanging metadata with other products . . . . . 85**

Publishing and synchronizing metadata. . . . .	85
How metadata is synchronized. . . . .	85
Before you begin: Establishing the environment for publishing metadata . . . . .	86

Preparing to publish OLAP server metadata . . . . .	87
Preparing to publish Data Warehouse Center metadata . . . . .	92
Exchanging MDIS-conforming metadata with other products . . . . .	96
Exchanging metadata by using the MDIS conversion utilities. . . . .	97
Importing MDIS-conforming tag language files . . . . .	98
Exporting MDIS-conforming tag language files . . . . .	100

**Chapter 7. Maintaining the Information Catalog Manager . . . . . 105**

Preventing problems. . . . .	105
Monitoring available disk space . . . . .	105
Ensuring that users can access the information catalog concurrently. . . . .	106
Backing up information catalog databases and configuration information . . . . .	106
Solving problems . . . . .	108
Using online information and messages	108
Resolving administrator logon problems	108
Recovering Information Catalog Manager components and data . . . . .	109
Using trace files for problem diagnosis	110

**Appendix A. Information Catalog Manager extract programs . . . . . 113**

Extract programs supplied with the Information Catalog Manager. . . . .	113
Planning to run extract programs . . . . .	113

**Appendix B. Predefined Information Catalog Manager object types . . . . . 115**

Accessing the Information Catalog Manager sample data. . . . .	115
Creating a sample information catalog	115
Initializing your information catalog with the predefined object types. . . . .	117
Predefined object type models . . . . .	117
Predefined object type descriptions . . . . .	121
Grouping category . . . . .	121
Elemental category . . . . .	125
Contact category . . . . .	127
Dictionary category . . . . .	127
Support category . . . . .	127
Program category. . . . .	128
Attachment category. . . . .	128

Predefined program objects . . . . .	128
<b>Appendix C. Metadata mappings . . . . .</b>	<b>133</b>
Metadata Mappings between the Information Catalog Manager and the Data Warehouse Center . . . . .	133
Metadata mappings between the Information Catalog Manager and OLAP server . . . . .	143
<b>Appendix D. Tag language . . . . .</b>	<b>147</b>
Rules for writing tag language files . . . . .	147
How the Information Catalog Manager reads tag language files . . . . .	148
Valid data types for Information Catalog Manager descriptive data . . . . .	149
How to read the tag language syntax diagrams. . . . .	150
ACTION.OBJINST . . . . .	150
Context . . . . .	151
Syntax . . . . .	151
Options . . . . .	151
ACTION.OBJTYPE . . . . .	155
Context . . . . .	155
Syntax . . . . .	155
Options . . . . .	155
ACTION.RELATION. . . . .	159
Context . . . . .	159
Syntax . . . . .	159
Options . . . . .	159
COMMENT. . . . .	160
Syntax . . . . .	160
Rules . . . . .	161
COMMIT . . . . .	161
Context . . . . .	161
Syntax . . . . .	161
Keywords . . . . .	162
Rules . . . . .	162
DISKCNTRL . . . . .	162
Context . . . . .	162
Syntax . . . . .	162
Keywords . . . . .	163
Rules . . . . .	163
INSTANCE . . . . .	163
Context . . . . .	163
Syntax . . . . .	163
NL. . . . .	168
Syntax . . . . .	168
Rules . . . . .	169
OBJECT . . . . .	169
Context . . . . .	169
Syntax . . . . .	169
PROPERTY . . . . .	174
Syntax . . . . .	175
Context . . . . .	175
Keywords . . . . .	175
Rules . . . . .	177
RELTYPE . . . . .	178
Syntax . . . . .	178
Context . . . . .	178
Keywords . . . . .	178
TAB . . . . .	179
Syntax . . . . .	179
Rules . . . . .	180
<b>Appendix E. What a tag language file should look like . . . . .</b>	<b>181</b>
Start your tag language file with DISKCNTRL	181
Define your additions, changes, and deletions. . . . .	181
Defining what you want to do . . . . .	181
Defining the information . . . . .	181
Putting it all together . . . . .	182
Committing changes to the database . . . . .	183
Putting comments in the tag language file	183
<b>Appendix F. Performing Information Catalog Manager functions from the command line . . . . .</b>	<b>185</b>
Starting the Information Catalog Manager from the command line . . . . .	185
Importing a tag language file from the command line . . . . .	185
Exporting metadata from the information catalog . . . . .	188
Creating an information catalog from the command line . . . . .	191
Importing common object types into the information catalog . . . . .	194
<b>Notices . . . . .</b>	<b>195</b>
Trademarks . . . . .	198
<b>Glossary . . . . .</b>	<b>201</b>
<b>Bibliography . . . . .</b>	<b>207</b>
<b>Index . . . . .</b>	<b>209</b>
<b>Contacting IBM . . . . .</b>	<b>221</b>
Product Information . . . . .	221





---

## About this book

This book can help information catalog administrators adapt and tailor the Information Catalog Manager 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 with the Information Catalog Manager, such as searching and browsing for information. These tasks are described in *Information Catalog Manager User's Guide*, which is available in the Information Center as an HTML book and on the documentation CD-ROM as a PDF file. The *Information Catalog Manager User's Guide* is structured so that you learn how to use the Information Catalog Manager 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 you 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. DB2 Universal Database for Enterprise - Extended Edition is the follow-on product to DB2 Parallel Edition and is supported on AIX®, Windows NT®, and in the Solaris Operating Environment.

### Administrator task information

The tasks of an administrator are in these categories:

#### Setting up the information catalog

You authorize users, create the information catalog, set up

some sample information for your users, and give users access to the software and resources they need. Description of these tasks begins 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 25.

### **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 with which users are familiar. Description of these tasks begins on page 43.

### **Making the information catalog convenient for users**

You group objects together to make them easier to browse, add contact names so that users can find someone to ask about the information, and set up programs so that users can start them and retrieve the information they want. You can maintain a *support facility* to inform users about changes in the information catalog, and a *dictionary facility* as a quick reference to terminology used in the information catalog. Description of these tasks begins on page 51.

### **Expanding and automating your information catalog**

You use the Information Catalog Manager tag language to make it easier to work with large amounts of descriptive data at once. 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 program. 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 71.

### **Maintaining the Information Catalog Manager**

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

Manager. You prevent or solve some of the problems that your users might have with the Information Catalog Manager. Description of these tasks begins on page 105.

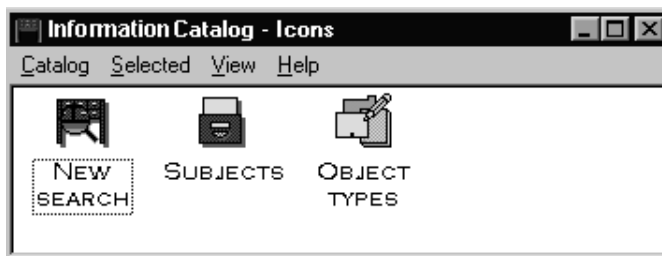
## Performing Information Catalog Manager tasks with the user interface or tag language

The Information Catalog Manager provides a graphical interface to your information catalog Information Catalog Manager. The Information Catalog Manager 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 Manager 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 Manager task, it is presented following the user interface description, under a heading “<task> using the tag language.”

### The user interface

To use the user interface, start from the administrators Information Catalog window, shown here.



### The tag language

The easiest way to use the tag language is to copy and paste the tag language templates that are provided online directly into your tag language file. To use an online tag language template:

1. Press F1 from any product window (after you open an information catalog). A help window opens.
2. Click **Contents** at the top of the help window.
3. From the table of contents, click the topic for the task you want to perform and then click the **Tag language template you can copy and modify** link that follows the task.

The tag language window opens for that task.

4. Click **Options** and click **Copy** from the menu to copy the entire window to the clipboard.

5. Paste the template from the clipboard into the desired tag file. You might need to reformat some of the information pasted from the clipboard.
6. Edit the variables in the template. Short descriptions for these variables are provided online. For more detailed descriptions, see the section on the task you want to perform within this book or “Appendix D. Tag language” on page 147 for the complete tag language reference.

---

## 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.software.ibm.com/data/vw/>

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 all the tasks necessary to set up an information catalog for users. Some of the tasks discussed in this chapter include information about:

- Setting up an information catalog in a LAN environment or as a stand-alone product.
- Authorizing users to an information catalog.
- Defining an information catalog on the various DB2 Universal Database platforms.
- Registering an information catalog that resides on a remote server so that users can access it.
- Ensuring that users can start programs from information catalog objects.

***If you plan to use the Information Catalog Manager in a LAN environment:***

After you install the Information Catalog Manager (see *DB2 Warehouse Manager Installation Guide*), you perform seven main tasks:

1. Authorize yourself and your users to use your information catalog.
2. When you install either the DB2 Warehouse Manager or the DB2 OLAP Server™, a default information catalog is created on DB2 Universal Database for Windows NT. If you do not use the default information catalog, you can create and identify an information catalog depending on your configuration, as described in Table 1 on page 2. If you plan to publish and synchronize Data Warehouse Center metadata to the information catalog, you must use the warehouse control database as your information catalog database. For more information about publishing and synchronizing metadata, see “Chapter 6. Exchanging metadata with other products” on page 85.
3. Open the Information Catalog Manager as an administrator.
4. Ensure that users can start applications to access the information that is described in the information catalog.
5. Authorize specific users to perform object management tasks, if necessary.
6. Set the status choices available to users when they create comments in the information catalog.
7. If you have not done so already, create a sample information catalog to help your users learn to use the Information Catalog Manager. (This task is part of the installation process in *DB2 Warehouse Manager Installation Guide*, and “Accessing the Information Catalog Manager sample data” on page 115.)

*Table 1. Creating and identifying information catalogs by configuration*

<b>Information Catalog Manager on DB2 UDB for Windows NT server</b>	<b>Information Catalog Manager administrator on client<sup>1</sup></b>
1. Create an information catalog on the server.	1. Register the server node and remote information catalog.
2. Register the server node and remote information catalog on client workstations.	2. Create the information catalog (by initializing either a new or existing information catalog).
	3. Register the server node and remote information catalog database.

**Note:**

1. Remote information catalogs can exist on DB2 Universal Database for AIX, DB2 Universal Database for the Solaris Operating Environment, DB2 Universal Database for Enterprise - Extended Edition, DB2 Universal Database for OS/2, DB2 Universal Database for OS/390, or DB2 Universal Database for Windows NT.

*If you plan to use the Information Catalog Manager as a stand-alone product:* After installing the Information Catalog Manager (see the *DB2 Warehouse Manager Installation Guide*):

1. Create the database that you plan to use for your information catalog.
2. Create an information catalog (users on other workstations will not be able to access this information catalog).
3. Complete steps 3, 6, and 7 from the previous procedure.

## **Authorizing users to the Information Catalog Manager**

You must authorize your users to use both the Information Catalog Manager for Windows and the Information Catalog Manager for the Web.

Start by deciding who will act as the primary and backup administrators of your information catalog.

Table 2 shows the level of authority that is required to create user IDs and passwords for each Information Catalog Manager configuration.

*Table 2. Database storage locations for information catalogs and the authority required to administer them*

<b>Database location for the information catalog</b>	<b>Authority required</b>
DB2 Universal Database for OS/2 server, file server	LAN domain administrator
DB2 Universal Database for OS/2 server, non-file server	System administrator for the server

## Authorizing users to the Information Catalog Manager

Table 2. Database storage locations for information catalogs and the authority required to administer them (continued)

Database location for the information catalog	Authority required
DB2 Universal Database for OS/390	RACF <sup>®</sup> administrator
DB2 Universal Database for AIX or DB2 Universal Database for Enterprise - Extended Edition (DB2 UDB EEE) <sup>1</sup>	AIX administrator
DB2 Universal Database for AS/400	AS/400 <sup>®</sup> security officer
DB2 Universal Database for Windows NT or DB2 Universal Database for Windows 2000	Administrator for the server

**Notes:**

1. If you plan to store your information catalog on DB2 Universal Database for Enterprise - Extended Edition (DB2 UDB EEE), follow the instructions for the platform on which your information catalog database resides.

**Note for Information Catalog Manager for the Web users:** See the *DB2 Warehouse Manager Installation Guide* for information on authorizing users to the Information Catalog Manager for the Web.

---

### Creating the information catalog

You can create an information catalog on any of these database management systems:

**DB2 Universal Database for OS/2**

You need to install the administration tools for DB2 UDB for OS/2.

**DB2 Universal Database for OS/390**

You must have the DB2 Connect product on your workstations to use DB2 UDB for OS/390.

**DB2 Universal Database for AS/400**

You must have the DB2 Connect product on your workstations to use DB2 UDB for AS/400.

**DB2 Universal Database for AIX**

You must have TCP/IP installed on your workstations to use DB2 UDB for AIX.

**DB2 Universal Database for Enterprise - Extended Edition**

If you plan to store your information catalog on DB2 UDB EEE, follow the instructions for the platform on which your information catalog database resides.

## Creating the information catalog

### DB2 Universal Database for the Solaris Operating Environment

You must have TCP/IP installed on your workstations.

### DB2 Universal Database for Windows NT and DB2 Universal Database for Windows 2000

You must have TCP/IP or NetBIOS installed on your workstations.

You can define your information catalog using the Information Catalog Manager user interface or from an MS-DOS™ command prompt. To define your information catalog from the user interface, use the information in this chapter. To define your information catalog from an MS-DOS command prompt, see “Creating an information catalog from the command line” on page 191.

## Defining an information catalog on DB2 Universal Database for OS/2

Table 3 describes the tasks you need to complete before defining the Information Catalog Manager on DB2 UDB for OS/2.

*Table 3. Preparing to define an information catalog on DB2 Universal Database for OS/2*

Task	Who	
	Remote database administrator	You
Create the database in which the information catalog will be stored.	X	
Ensure that you have SYSADM authority to define the new information catalog.	X	

*To define the information catalog:*

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Initialize Information Catalog**.

The Initialize Information Catalog window opens.

2. From the **Select type of information catalog** list, click **DB2 UDB for OS/2**.
3. Click **OK**.

The Define Catalog on DB2 UDB for OS/2 window opens.

4. In the **Information catalog name** field, type the alias name for the remote database that is cataloged on your local workstation.
5. From the **Not-applicable symbol** list, click a character:
  - a. Click on the down arrow to display a list of valid symbols.



- b. Click the symbol that you want to use from the list.
6. In the **Primary administrator's user ID** field, type the user ID of the person who will be the primary administrator of the information catalog. This user ID must have SYSADM authority.
7. In the **Backup administrator's user ID** field, type the user ID of the person who will back up the primary administrator.
8. Select the **Import common object types** check box to populate your information catalog with object types that you can use to exchange metadata with other conforming products.
9. Click **Define**.  
The Connect to Information Catalog window opens.
10. In the **User ID** field, type the LAN user ID (which is specified with UPM on the remote workstation).
11. In the **Password** field, type the password for the user ID that you entered in the **User ID** field.
12. Click **Connect**.

Your new catalog is defined, and two Information Catalog Manager program icons are created for the Information Catalog Manager entry on the **Start** menu. One icon represents administrator functions, which you can use only if you have the information catalog administrator component installed on your workstation. The other icon represents user functions.

If you receive an error message stating that the Information Catalog Manager could not import the common object types, you need to initialize your information catalog with the predefined object types (see page 117). Then you can add the object types to your new information catalog.

### Defining an information catalog on DB2 Universal Database for OS/390

Table 4 on page 6 describes the tasks you or your remote database administrator need to complete before defining an information catalog on DB2 for OS/390.

## Creating the information catalog

Table 4. Preparing to define an information catalog on DB2 for OS/390

Task	Who	
	Remote database administrator	You
Create the database in which the information catalog will be stored.  The Information Catalog Manager provides a sample JCL file that the database administrator can change to create the database, storage groups, and table spaces. The sample file is called DGCRITDB.JCL and resides in the \SQLLIB\BIN directory in the path where the DB2 Universal Database is installed.	X	
Ensure that you have SYSADM authority to define the new information catalog.	X	
Ask your database administrator for the following names: <ul style="list-style-type: none"><li>• Database name</li><li>• Storage group name for tables</li><li>• Storage group name for indexes</li></ul>	X	X

### *To define the information catalog:*

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Initialize Information Catalog**.

The Initialize Information Catalog window opens.

2. From the **Select type of information catalog** list, click **DB2 UDB for OS/390**.
3. Click **OK**.

The Define Catalog on DB2 UDB for OS/390 window opens.

4. In the **Information catalog name** field, type the alias name for the remote database that is cataloged on your local workstation.

5. In the **DB2 database name** field, type the name of the DB2 database.
6. In the **Name of storage group for tables** entry field, type the name of the storage group that you will use for tables.
7. In the **Name of storage group for indexes** field, type the name of the storage group that you will use for indexes.
8. From the **Not-applicable symbol** list, click a character:
  - a. Click on the down arrow to display a list of valid symbols.
  - b. Click the symbol that you want to use.
9. In the **Primary administrator's user ID** field, type the user ID of the person who will be the primary administrator of the Information Catalog Manager. This user ID must have SYSADM authority on DB2 UDB for OS/390.
10. In the **Backup administrator's user ID** field, type the user ID of the person who will back up the primary administrator.
11. Optional: To save the property values of each object in uppercase, select the **Save object values in uppercase** check box. The values are stored in the DB2 UDB for OS/390 database in uppercase, but you can enter the values in lowercase when you search for them.
12. Select the **Import common object types** check box to populate your information catalog with object types that you can use to exchange metadata with other conforming products.
13. Click **Define**.

The Connect to Information Catalog window opens.
14. In the **User ID** field, type your RACF user ID.
15. In the **Password** field, type the password for the user ID that you entered in the **User ID** field.
16. Click **Connect**.

Your new catalog is defined, and two Information Catalog Manager program icons are created for the Information Catalog Manager entry on the **Start** menu. One icon represents administrator functions, which you can use only if you have the information catalog administrator function installed on your workstation. The other icon represents user functions.

If you receive an error message stating that the Information Catalog Manager could not import the common object types, you need to initialize your information catalog with the predefined object types (see page 117). Then you can add the object types to your new information catalog.

### Defining an information catalog on DB2 Universal Database for AS/400

Table 5 on page 8 describes the tasks that you or your remote database administrator need to complete before defining an information catalog on DB2 UDB for AS/400.

## Creating the information catalog

Table 5. Preparing to define an information catalog on DB2 Universal Database for AS/400

Task	Who	
	Remote database administrator	You
Create the database in which the information catalog will be stored.	X	
Ensure that you have ALLOBJ authority to define the new information catalog.	X	

### To define the information catalog:

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Initialize Information Catalog**.  
The Initialize Information Catalog window opens.
2. From the **Select type of information catalog** list, click **DB2 UDB for AS/400**.
3. Click **OK**.  
The Define Catalog on DB2 UDB for AS/400 window opens.
4. In the **Information catalog name** field, type the alias name of the remote database that is cataloged on your local workstation.
5. From the **Not-applicable symbol** list, click a character:
  - a. Click on the down arrow to display a list of valid symbols.
  - b. Click the symbol that you want to use.
6. In the **Primary administrator's user ID** field, type the user ID of the person who will be the primary administrator of the Information Catalog Manager.  
The user ID must have ALLOBJ authority on DB2 UDB for AS/400.
7. In the **Backup administrator's user ID** field, type the user ID of the person who will back up the primary administrator.
8. Select the **Import common object types** check box to populate your information catalog with object types that you can use to exchange metadata with other conforming products.
9. Click **Define**.  
The Connect to Information Catalog window opens.
10. In the **User ID** field, type your AS/400 user ID.
11. In the **Password** field, type the password for the user ID you entered in the **User ID** field.

### 12. Click **Connect**.

Your new catalog is defined, and two Information Catalog Manager program icons are created for the Information Catalog Manager entry on the **Start** menu. One icon represents administrator functions, which you can use only if you have the information catalog administrator function installed on your workstation. The other icon represents user functions.

If you receive an error message stating that the Information Catalog Manager could not import the common object types, you need to initialize your information catalog with the predefined object types (see page 117). Then you can add the object types to your new information catalog.

### Defining an information catalog on UNIX<sup>®</sup> systems

Table 6 describes the tasks that you or your remote database administrator need to complete before defining the information catalog on DB2 UDB for AIX and DB2 UDB for the Solaris Operating Environment. These steps also apply to DB2 UDB EEE for AIX and DB2 UDB EEE for the Solaris Operating Environment.

*Table 6. Preparing to define an information catalog on UNIX systems*

Task	Who	
	Remote database administrator	You
Create or identify the database in which the information catalog will be stored. If the database is created on the remote host, the database administrator might want to specify an authentication level for database security reasons.	X	
Ensure that you have SYSADM authority to define the new information catalog.	X	
Ask your database administrator for the name of the database.	X	X

## Creating the information catalog

Table 6. Preparing to define an information catalog on UNIX systems (continued)

Task	Who	
	Remote database administrator	You
<p>If you are defining a DB2 UDB EEE information catalog, enter the following SQL commands from the DB2 Command Line Processor:</p> <pre>CREATE NODEGROUP FLG32K ON NODE number CREATE REGULAR TABLESPACE FLG32K IN NODEGROUP FLG32K MANAGED BY SYSTEM USING ('FLG32K')</pre> <p>where number is the identifying number of the node.</p>	X	

### To define the information catalog:

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Initialize Information Catalog**.  
The Initialize Information Catalog window opens.
2. From the **Select type of information catalog** list, click **DB2 family**.
3. Click **OK**.  
The Define Catalog on DB2 Family Database window opens.
4. In the **Information catalog name** field, type the alias name of the remote database that is cataloged on your local workstation.
5. From the **Not-applicable symbol** list, click a character:
  - a. Click on the down arrow to display a list of valid symbols.
  - b. Click the symbol that you want to use.
6. In the **Primary administrator's user ID** field, type the user ID of the person who will be the primary administrator of the Information Catalog Manager.  
This user ID must have SYSADM authority.
7. In the **Backup administrator's user ID** field, type the user ID of the person who will back up the primary administrator.
8. Select the **Import common object types** check box to populate your information catalog with object types that you can use to exchange metadata with other conforming products.

9. Click **Define**.

The Connect to Information Catalog window opens.

10. In the **User ID** field, type your AIX user ID.

11. In the **Password** field, type the password for the user ID you entered in the **User ID** field.

Passwords are case-sensitive on UNIX operating systems; you must type them exactly as specified.

12. Click **Connect**.

Your new catalog is defined, and two Information Catalog Manager program icons are created for the Information Catalog Manager entry on the **Start** menu. One icon represents administrator functions, which you can use only if you have the information catalog administrator function installed on your workstation. The other icon represents user functions.

If you receive an error message stating that the Information Catalog Manager could not import the common object types, you need to initialize your information catalog with the predefined object types (see page 117). Then you can add the object types to your new information catalog.

### **Defining an information catalog on DB2 Universal Database for Windows NT or DB2 Universal Database for Windows 2000**

Table 7 on page 12 describes the tasks that you or your remote database administrator need to complete before defining an information catalog on DB2 UDB for Windows NT or DB2 UDB for Windows 2000. These steps also apply to DB2 UDB EEE for Windows NT.

## Creating the information catalog

Table 7. Preparing to define an information catalog on DB2 Universal Database for Windows NT or DB2 Universal Database for Windows 2000

Task	Who	
	Remote database administrator	You
<p>If you are defining a DB2 UDB EEE information catalog, enter the following SQL commands from the DB2 Command Line Processor:</p> <pre>CREATE NODEGROUP FLG32K ON NODE number CREATE REGULAR TABLESPACE FLG32K IN NODEGROUP FLG32K MANAGED BY SYSTEM USING ('FLG32K')</pre> <p>where number is the identifying number of the node.</p>	X	
<p>Create or identify the database in which the information catalog will be stored. If the database is created on the remote host, the database administrator might want to specify an authentication level for database security reasons.</p>	X	
<p>Ensure that you have administrator authority to define the new information catalog.</p>	X	
<p>Ask your database administrator for the name of the database.</p>	X	X

### To define the information catalog:

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Initialize Information Catalog**.  
The Initialize Information Catalog window opens.
2. From the **Select type of information catalog** list, click **DB2 UDB for Windows NT**.



3. Click **OK**.

The Define Catalog on DB2 UDB for Windows NT window opens.

4. In the **Information catalog name** field, type the name that you want to assign to the local information catalog or the alias name of the remote database that is cataloged on your local workstation.
5. From the **Not-applicable symbol** list, click a character:
  - a. Click on the down arrow to display a list of valid symbols.
  - b. Click the symbol that you want to use.
6. In the **Primary administrator's user ID** field, type the user ID of the person who will be the primary administrator of the Information Catalog Manager.

This user ID must have SYSADM authority.
7. In the **Backup administrator's user ID** field, type the user ID of the person who will back up the primary administrator.
8. Select the **Import common object types** check box to populate your information catalog with object types that you can use to exchange metadata with other conforming products.
9. Click **Define**.

The Connect to Information Catalog window opens.

10. In the **User ID** field, type the user ID required by the database that contains your information catalog:

**DB2 UDB for Windows NT (local)**

Windows NT<sup>®</sup> user ID

**DB2 UDB for Windows NT or DB2 UDB for Windows 2000 (remote)**

LAN user ID, which is specified with User Manager on the remote workstation

**DB2 UDB for Windows 2000 (local)**

Windows 2000 user ID

11. In the **Password** field, type the password for the user ID that you entered in the **User ID** field.

Passwords on Windows systems are case-sensitive; you must type them exactly as specified.

12. Click **Connect**.

Your new catalog is defined, and two Information Catalog Manager program icons are created for the Information Catalog Manager entry on the **Start** menu. One icon represents administrator functions, which you can use only if you have the information catalog administrator installed on your workstation. The other icon represents user functions.

## Creating the information catalog

If you receive an error message stating that the Information Catalog Manager could not import the common object types, you need to initialize your information catalog with the predefined object types (see page 117). Then you can add the object types to your new information catalog.

### Migrating a DataGuide Version 5.2 information catalog

You can migrate a DataGuide® 5.2 information catalog to a DB2 Universal Database Version 7.1 information catalog if the information catalog resides on the following database systems:

- DB2 UDB for OS/2
- DB2 UDB for AIX
- DB2 UDB EEE
- DB2 UDB for OS/390
- DB2 UDB for AS/400
- DB2 UDB for Windows NT

You cannot migrate information catalogs that reside on DB2 UDB for Windows 95 or DB2 UDB for Windows 98. However, you can connect remotely to them.

After you install either the DB2 Warehouse Manager or the DB2 OLAP Server with the Information Catalog Manager component, use the Initialize Information Catalog window to define the information catalog. The steps for defining an information catalog begin on page 4. Select the task that is appropriate for the database system on which your information catalog database resides.

---

## Registering a server node and information catalog

An information catalog can either be local (stored on your workstation) or remote. If an information catalog is remote, you must register it and the server where it resides.

### Registering a server node using the DB2 Control Center

If you previously connected to a database that resides on the server on which your remote information catalog resides, you can skip this section and continue with “Registering a remote information catalog” on page 17.

You can 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. See the DB2 Control Center online help for more details.

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

## Registering a server node and information catalog

After these tasks are completed in the DB2 Control Center, you can then use the Information Catalog Manager Register New Information Catalog window to complete the registration process. Make sure that you have:

- The node name of the remote server.
- The database name under which your information catalog is cataloged on the remote server.

You can skip the next section and continue with “Registering a remote information catalog” on page 17.

### Registering a server node with the Information Catalog Manager

If you previously connected to a database that resides on the server on which your remote information catalog resides, you can skip this section and continue with “Registering a remote information catalog” on page 17.

Table 8 on page 16 describes the tasks that you or your remote database administrator need to complete before registering a server node and information catalog, if you do not use the DB2 Control Center.

## Registering a server node and information catalog

Table 8. Preparing to register a server node and information catalog

Task	Who	
	Remote database administrator	You
<p>Edit the appropriate protocol command file for your information catalog users' environment. There are four protocol command files available:</p> <ul style="list-style-type: none"> <li>• DGNTBIOS uses NetBIOS protocol</li> <li>• DGTCP/IP uses TCP/IP protocol</li> <li>• DGCPIC uses CPIC protocol</li> <li>• DGIPXSPX uses IPX/SPX protocol</li> </ul> <p>The command files in this list are provided as examples, but you are not limited to those. If you change the command files or create new ones, they must have a file extension of BAT and reside in the \SQLLIB\PROTOCOL directory.</p> <p>You must modify each file to add the node name where the remote information catalog resides.</p>		X
Ask your database administrator for the node name of the remote server.	X	X
Ask your database administrator for the database name under which your information catalog is cataloged on the remote server.	X	X

To register a server node:

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Register Server Node and Information Catalog**.

## Registering a server node and information catalog

The Register Server Node and Information Catalog window opens.

2. Click **Register new server node**, and click **OK**.  
The Register New Server Node window opens.
3. Select a command file that is appropriate for your environment. The **Contents of command file** field displays the protocol and database commands that establish the connection between the Information Catalog Manager and the remote database where your information catalog is stored.
4. Click **Register**.
5. The Register Server Node and Information Catalog window remains open. You can register other server nodes or information catalogs, or click **Cancel** to close the window.

### Registering a remote information catalog

To register a remote information catalog:

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Register Server Node and Information Catalog**.  
The Register Server Node and Information Catalog window opens.
2. Click **Register new information catalog**, and click **OK**.  
The Register New Information Catalog window opens.
3. In the **Information catalog name** field, type the name the remote catalog will have on your local workstation.
4. In the **Server information catalog name** field, type the name the information catalog has on the remote server.
5. From the **Server node ID** list, click the server node or DB2 Connect gateway workstation where the remote information catalog is located.
6. Click **Register**.  
The Connect to Information Catalog window opens.
7. In the **User ID** field, type the user ID required by the database that stores your information catalog:

#### **DB2 Universal Database for OS/2 (remote)**

LAN user ID, which is specified with UPM on the remote workstation

#### **DB2 Universal Database for OS/390**

RACF user ID

#### **DB2 Universal Database for AS/400**

AS/400 user ID

#### **DB2 Universal Database for AIX**

AIX user ID

## Registering a server node and information catalog

### DB2 Universal Database for Enterprise - Extended Edition

AIX, Solaris Operating Environment, or LAN user ID (depending on the operating system)

### DB2 Universal Database for the Solaris Operating Environment

Solaris Operating Environment user ID

### DB2 UDB for Windows NT or DB2 UDB for Windows 2000 (remote)

LAN user ID, which is specified with User Manager on the remote workstation

8. In the **Password** field, type the password for the user ID that you entered in the **User ID** field.

Passwords are case sensitive for accessing databases on the following operating systems, you must type them exactly as specified:

- AIX
- Windows NT and Windows 2000
- Solaris Operating Environment

9. Click **Connect**.

Your new catalog is defined, and two Information Catalog Manager program icons are created for the Information Catalog Manager entry on the **Start** menu. One icon represents administrator functions, which you can use only if you have the information catalog administrator function installed on your workstation. The other icon represents user functions.

10. The Register Server Node and Information Catalog window remains open. You can register other server nodes or information catalogs, or click **Cancel** to close the window.

---

## Opening an information catalog

You start working with the Information Catalog Manager by opening an information catalog:

1. Click **Start** → **Programs** → **IBM DB2** → **Information Catalog Manager** → **Initialize Information Catalog**.

2. Click the icon that represents the information catalog that you want to open.

The Open Information Catalog window opens.

3. In the **User ID** field, type the user ID required for the operating system on which your information catalog resides.

4. In the **Password** field, type the password for the user ID that you entered in the **User ID** field.

Passwords are case sensitive for accessing databases on the operating systems, you must type them exactly as specified:

- AIX

- Windows® 32-bit operating systems
  - Solaris Operating Environment
5. Click **Open**. The Information Catalog window opens.

You can also open your information catalog from an MS-DOS command prompt; see “Importing a tag language file from the command line” on page 185 for more information.

---

### Ensuring that users can start programs from the Information Catalog Manager

You set up the objects in your information catalog so that your users can run application programs to work with the actual information that the objects describe. Users can run the application programs that they are familiar with, including the programs that were originally used to create the information.

Ensure that the following requirements are met:

- Your users need the appropriate application software installed on their workstations or on the LAN.
- Users can launch any program that can be started from the command line with the command `start program_name` without a path, regardless of where the program is installed.

Many programs write their path to the program registry when they are installed. The `start` command retrieves the path. If a program does not write its path to the program registry, you might need to add the directory path of the program to the path environment variable on users' workstations.

- Your users need the necessary authorization to the databases or file systems where the information that they need is stored.
- The Programs objects in the information catalog must include the correct invocation syntax for the operating systems on which your users will run the programs.

### Additional requirements for Information Catalog Manager for the Web users

When you set up the Web environment for Information Catalog Manager for the Web users, ensure that the following requirements are met:

- The data that users want to use with the application program must be accessible to the Web server. For example, the Information Catalog Manager sample data file is located in a directory on the Web server.
- The program that users want to start must be installed on the Web client. For example, if users are accessing a Lotus® 1-2-3 file®, then Lotus 1-2-3 must be installed on the Web client.

## Ensuring that users can start programs from the Information Catalog Manager

If the application program is a Java™ applet, the application does not need to be installed; it can be accessed directly from the Web browser.

The client should also have any necessary browser plug-in programs. The Information Catalog Manager for the Web server must be able to locate any associated files that are used by the plug-in program. For example, if the users want to view Adobe Acrobat files, they need the browser plug-in program for the Acrobat Reader installed on the Information Catalog Manager for the Web client. The Information Catalog Manager for the Web server must be able to locate the file that the user wants to view to download it to the client.

- The required MIME types must be identified in the Web server configuration file for the application program that users will start. An AddType directive with the file extension of the program that users want to start must be included in the configuration file. For example, if users want to use Lotus 1-2-3 spreadsheets with a file type of WK4, define the AddType directive for Lotus Domino™ Go Webserver as shown in this example:

```
AddType .WK4 application/x-lotus1-2-3 binary
```

If users are using a Web server other than Lotus Domino Go Webserver, the MIME types are defined differently. See your Web server documentation for more information.

- If you are using Websphere IBM HTTP WebServer, the MIME types are defined in the \conf\mime.types file as shown in this example:  
application/vnd.lotus-1-2-3 wks 123 wk1 wk2 wk3 wk4
- For some versions of Netscape Navigator, helper programs recognize file types and start the corresponding application program. Microsoft Internet Explorer does not use helper programs. Instead, Internet Explorer uses file type and program associations that are used by Windows Explorer; no setup is required for Internet Explorer to recognize a file type.
- The **URL to access data** property must be defined for the object from which users want to start the program. The value for the property is a link to directly launch the program.

To start a program from an Information Catalog Manager for the Web object:

1. In the list pane, click on the object from which you want to start the program.

The object description page opens in the description pane.

2. Find the **URL to access data** property.
3. Click the property value.

The Web browser is launched using the Web address that is specified by the property value.



### Authorizing Information Catalog Manager users to manage objects

You can give specific Information Catalog Manager users in your organization authority to manage and update objects in an information catalog. Authorized Information Catalog Manager users can do tasks such as creating, updating, and exporting objects.

To authorize Information Catalog Manager users:

1. From the menu bar of the Information Catalog window, click **Catalog** → **Manage users**.

The Manage Information Catalog Users window opens.

2. In the **New user ID** field, type the user ID of the Information Catalog Manager user that you want to authorize to manage objects.
3. Click **Add**. The new user ID is added to the **Users** list.
4. Click **OK** to complete the authorization and commit the changes to the information catalog.

To remove an Information Catalog Manager user ID from the list of authorized user IDs:

1. Click on the user ID in the **Users** list.
2. Click **Remove**.
3. Click **OK** to complete the authorization and commit the changes to the information catalog.

Setting and changing information catalog administrators: You can use the Manage Information Catalog Users window to change the user IDs of your information catalog administrators. You can enter the user IDs of the primary and backup administrators in the **Primary user ID** and **Backup user ID** fields, respectively.

Before you change the primary administrator, ensure that the new primary administrator has SYSADM authority if your information catalog is stored in a DB2 UDB for Windows NT, DB2 UDB for AIX, DB2 UDB for the Solaris Operating Environment, DB2 UDB EEE, DB2 UDB for OS/390, or DB2 UDB for OS/2 database. If your information catalog is stored in a DB2 UDB for AS/400 database, make sure that the new primary administrator has ALLOBJ authority.

If you enter an incorrect user ID for both primary and backup administrators and cannot access your information catalog, use the Information Catalog Manager ALTERKA command to set the correct user ID. To change an administrator for an information catalog, enter the following command at an MS-DOS command prompt:

```
X:Program Files\SQLLIB\BIN\ALTERKA
```

## Authorizing the Information Catalog Manager users to manage objects

where X is the drive where the DB2 Universal Database is installed.

At the prompt, enter the information catalog name, user ID, and password.  
Separate the values with blanks:

```
ICMSAMP longods secret
```

At the prompt, enter the administrator user ID that you want to alter, the action (add, delete, or update), and the user ID type (primary or backup).  
Separate the values with blanks:

```
valdezma D B
```

The example deletes the backup administrator user ID, valdezma.

---

## Setting and changing status values for comments

You can set the list of available status choices for users to assign to comments in your information catalog. For example, status choices might be Open, Pending, Action required, and Closed.

The status choices that you identify are displayed in the **Status** list in the Create Comment, Copy Comment, and Update Comment windows (as shown in Figure 1).

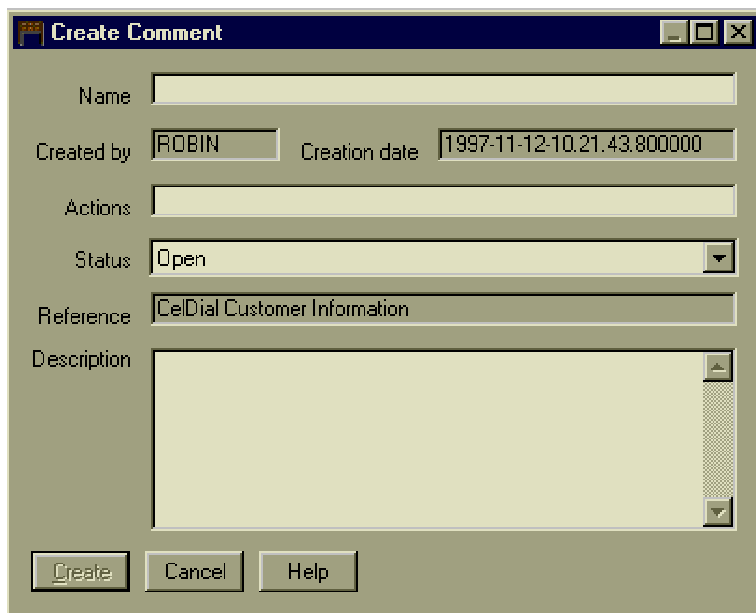


Figure 1. Create Comment window showing status choices

## Setting and changing status values for comments

To set or change the comment status choices:

1. From the Information Catalog window, Click **Catalog** → **Update comment status list**.

The Update Comment Status List window opens.

2. Set the list of up to ten status choices by typing one status in each **Comment status field choice** field. The order of the choices that you enter is the order of the **Status** list in the various comment windows.
3. Click **Update** when you finish setting status values.

To close the window without updating the status list, click **Cancel**.

## Setting and changing status values for comments

---

## Chapter 2. Organizing your information resources

By now, you have created an information catalog and ensured that you and your users can open it. Next, you must complete some necessary preparatory tasks 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.

When you have categorized the types of information that you want to include in your information catalog, you identify the types of information in your information catalog.

---

### Information Catalog Manager categories and object types

To organize your information resources in the information catalog, you create 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 (Figure 2 on page 26), which describes a set of objects that are digital bitmap images. For each object type, you define a set of *properties*, which describe the characteristics of the object

## Information Catalog Manager information categories and object types

type. For an object type that is called Image, you might define properties such as Resolution, Size, and Color.

Every object type must belong to the Information Catalog Manager *category*.

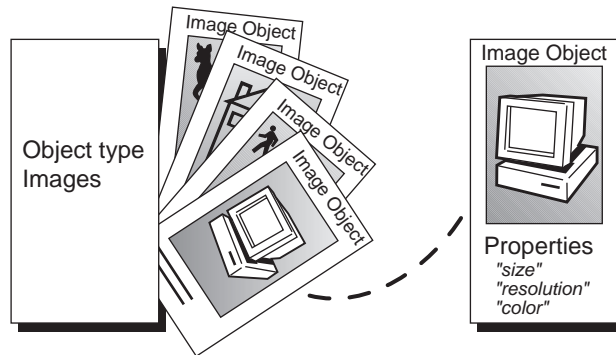


Figure 2. The characteristics of an information resource become properties of the Information Catalog Manager object type

An object type's category affects how the Information Catalog Manager handles it. Except for the Program and Attachment categories, you can create object types in any of the following Information Catalog Manager categories:

Category	Definition
Grouping	Object types that can contain other object types.
Elemental	Non-Grouping object types that are the building blocks for other Information Catalog Manager object types.
Contact	Object types that identify 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.
Program	A Programs object type that identifies and describes applications capable of processing the actual information that is represented by the Information Catalog Manager objects types. The only object type belonging to the Program category is the Programs object type, which is defined when you create an information catalog.
Dictionary	Object types that define terminology that is specific to your business.

## Information Catalog Manager information categories and object types

### Support

Object types that provide additional information about your information catalog or enterprise.

### Attachment

A Comments object type that identifies additional information attached to another Information Catalog Manager object. The only object type belonging to the Attachment category is the Comments object type, which is defined when you create an information catalog.

Table 9 summarizes the relationships among the Information Catalog Manager's object-type categories.

*Table 9. Information Catalog Manager category relationships*

Category	Can contain/contained by	Links with	Contacts associated	Comments attached	Programs launch from
Grouping	Contains other Grouping or Elemental objects	Other Grouping or Elemental objects	Yes	Yes	Yes
Elemental	Contained by any Grouping object	Other Grouping or Elemental objects	Yes	Yes	Yes
Contact	None	None	No	Yes	Yes
Program	None	None	No	Yes	No
Dictionary	None	None	No	Yes	Yes
Support	None	None	No	Yes	Yes
Attachment	None	None	No	No	Yes

You can establish object types for your information catalog in any of three ways:

- Use the object types that come with the Information Catalog Manager in the sample information catalog. (See "Appendix B. Predefined Information Catalog Manager object types" on page 115 for information about creating the sample information catalog and a description of the object types it includes.)
- Modify the object types that come with the Information Catalog Manager to fit your organization's needs (see "Updating an object type" on page 38 for information about modifying an object type).
- Create your own object types.

## Creating your own object types

---

### Creating your own object types

When you create your own object types, start by creating a prototype for each object type that you need. Then create one or two sample objects (see “Chapter 3. Populating the catalog with information” on page 43 for information about objects). Check how the objects appear in a Description view, especially the order in which the properties are listed. Try entering different values for each property to be sure 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, you can easily delete it and your sample objects and start over. After you create an object type, the only way to change or delete its properties is to delete the object type *and all objects of that type*. You must then create a new object type with different properties.

Also consider how many object types you will need. The Information Catalog Manager limits the number of object types that you can create in an information catalog to 999 999. It limits the number of objects that you can create for each type to 99 999 999. This limit includes all the object types that you ever created, even the ones that you deleted.

You can create an object type by using the Information Catalog Manager windows or tag language.

### Notes for creating object types for the Information Catalog Manager for the Web

When you create a new object type in the Information Catalog Manager, you can associate a unique icon for the object type. For the Information Catalog Manager for the Web, you should create each new object type icon using the following requirements:

- Create two .gif files: one file should be sized at 20 by 20 pixels; the second should be sized at 32 x 32 pixels for best visual display.
- Use the following files names for each .gif file. Make certain that the file names are in lower case:
  - For the .gif file that is 32 x 32, use `dg_obj_short_name.gif`  
Where `obj_short_name` is the short name (PTNAME) for the object type.
  - For the .gif file that is 20 x 20, use `dg_smobj_short_name.gif`  
Where `obj_short_name` is the short name (PTNAME) for the object type.

Make certain that you place the new .gif files on your Web server in the \ICONS directory.



### Creating an object type by using the Information Catalog Manager windows

Start from the Information Catalog window.

1. Right-click the **Object types** icon.
2. Click **Open as** → **Icon list**.
3. Right-click the **New Object Type** icon.
4. Click **Open**.

The Create Object Type window opens.

**Create Object Type**

Category:

Object type:

Short name:

Created by:

Updated by:

Last updated:

Windows icon:

Property name	Short name	Data type
Name	NAME	VARCH

5. Click on the down arrow to display a list of Information Catalog Manager categories and click one.
6. Type a unique external name for the new object type in the **Object type name** field.

The rules for object type names are:

## Creating your own object types

- 80 character maximum.
  - It must not contain null characters.
  - It must not be all blank characters.
7. Type a unique short name for the new object type in the **Short name** field.  
The rules for short names are:
    - 8 character (single-byte character set) maximum.
    - First character must be uppercase or lowercase English alphabetic, @, #, or \$.
    - Subsequent characters must be uppercase or lowercase English alphanumeric, @, #, \$, or \_.
  8. Optional: Identify the object type's icons. The default Windows icon for the selected category is displayed in the **Windows icon** field. To identify a different icon to represent the object type:
    - Click **Find** to locate a different Windows icon and display it in the window.
  9. Define all the properties for the object type (see "Defining the object type's properties" for detailed information).

<b>Click on:</b>	<b>To:</b>
<b>Add</b>	Define additional properties
<b>Modify</b>	Change a property before you create the object type
<b>Remove</b>	Remove a property before you create the object type

Click the property, and then click **Remove**.

10. Click **Define UI** to choose up to five properties that constitute the universal unique identifier (UI) for the object type (see "Defining a universal unique identifier for the object type" on page 33 for detailed information).
11. Click **Create** to save your changes in the database.  
Your changes display in the Object Types window, but not in other windows until you close and re-open them.  
To close the window without creating an object type, click **Cancel**.

### Defining the object type's properties

Each object type can have up to 255 properties. The order in which you define the properties is the order in which the user will see them. You cannot change or rearrange the properties after you create the object type.

The Information Catalog Manager defines five properties that are common to all the Information Catalog Manager object types. These five properties are summarized in Table 10 on page 31.

Table 10. Information Catalog Manager object type common properties

External name of property	Property short name	Definition
Object type identifier	OBJTYPID <sup>1</sup>	The Information Catalog Manager generates this value, which uniquely identifies the object type of an object within the scope of the local information catalog.
Instance identifier	INSTIDNT <sup>1</sup>	The Information Catalog Manager generates this value, which uniquely identifies an object within the scope of the local information catalog.
Name	NAME	You provide the name of an object. Choose names that users readily recognize and understand.
Last Changed Date and Time	UPDATIME <sup>1</sup>	The Information Catalog Manager generates this value, indicating the date and time the object was last changed.
Last Changed By	UPDATEBY <sup>1</sup>	The Information Catalog Manager generates this value, showing the user ID of the Information Catalog Manager session that last updated the Last Changed Date and Time property.

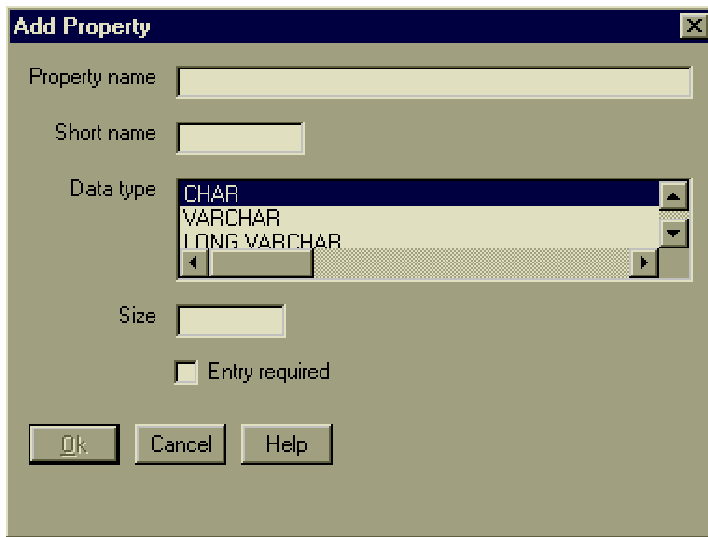
**Note:**

1. If you select the **Hide system generated properties** check box in the Information Catalog Manager Settings notebook, you will not see this property in the description of the object.

**To define additional properties:**

1. Click **Add** next to the **Properties** list. The Add Property window opens.

## Creating your own object types



Use this window to define properties.

2. In the **Property name** field, type a name for the property.

The rules for property names are:

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

3. In the **Short name** field, type a unique property short name.

The rules for short names are:

- 8 character (SBCS) maximum.
- First character must be uppercase or lowercase English alphabetic, @, #, or \$.
- Subsequent characters must be uppercase or lowercase English alphanumeric, @, #, \$, or \_.
- 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 Manager asks you for another name.

4. Choose a data type for the property from the **Data type** list:

### **CHAR**

Up to 254 characters

### **VARCHAR**

Up to 4 000 characters

### **LONG VARCHAR**

Up to 32 700 characters

### TIMESTAMP

Exactly 26 characters, in this format:

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

You can have up to 14 LONG VARCHAR properties for an object type.

5. Type a size for the property in the **Size** field. The size must be within the range for the data type you selected.
6. If you want to require entry of this property whenever you create an object of this type, select **Entry required**.
7. Click **OK** to return to the Object Types window.

### Defining a universal unique identifier for the object type

All object types must have at least one property that is part of the *universal unique identifier*, or UUI. The UUI is a string of characters that enables the Information Catalog Manager to tell one object from another. This requirement 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.

Without a way to uniquely identify these objects, you risk overwriting the descriptive data when you combine information catalogs.

The Information Catalog Manager prevents overwriting by having you define the UUI. 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 five properties of an object type and designate them in whatever order you want. The values for each of these properties, in the order you give them, become the UUI for any object of that type.

When you import an object into your information catalog, the Information Catalog Manager compares the values of the UUI properties to see if they match those of an existing object. If all the UUI properties have the same value in both objects, the Information Catalog Manager treats the two as the same object. It updates the values in the existing object's non-UUI properties. If the UUI properties have different values, the Information Catalog Manager adds the incoming object to the information catalog.

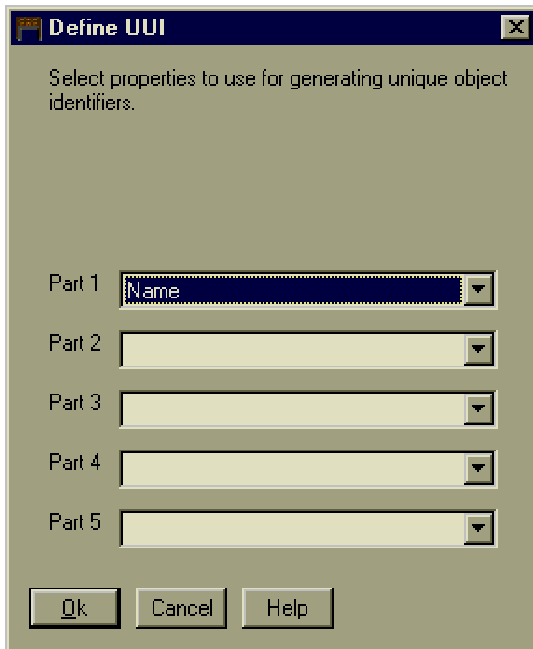
If you want to designate a property that you are sure is unique, such as a purchase requisition number or international standard book number (ISBN), you do not need to designate all five properties. You can enter the values for

## Creating your own object types

the UUI properties that you do not need with the not-applicable symbol. Or, when you create an object type, you can give it fewer UUI properties. (The not-applicable symbol is a hyphen unless you identified a different symbol when you created the information catalog.)

For performance reasons, be careful to select UUI properties so that the total number of characters in their combined values is fewer than 254.

*To define the UUI, start from the Define UUI window:*



1. Select up to five properties as parts of the UUI. The part number determines the position that the property has in the UUI sequence. The Information Catalog Manager limits your choice of properties:
  - You can choose only required properties.
  - You cannot choose properties whose data type is LONG VARCHAR.
  - You cannot choose properties whose data type is VARCHAR and whose length exceeds the 254-byte maximum.
  - You cannot use the same property for more than one part.
  - You cannot skip parts (for example, you cannot choose only parts 1, 3, and 5).

For each part, choose a property:

- a. Click the down arrow next to the **Part** field to see a list of available properties.

- b. Click a property.
2. When you finish filling in the parts, click **OK** to accept the UUI definition and return to the Object Types window.

### Creating an object type by using the Information Catalog Manager tag language

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

```
ACTION.OBJTYPE(ADD)
OBJECT.TYPE(short_name_of_object_type)
    PHYNAME (name_of_table)
    CATEGORY(category_of_object_type)
    EXTNAME(external_name_of_object_type)
    ICWFILE(name_of_Windows_icon_file)
```

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

#### **Keyword**

##### **Value**

- TYPE** The short name of the object type. The rules for short names are:
- 8 character (SBCS) maximum.
  - First character must be uppercase or lowercase English alphabetic, @, #, or \$.
  - Subsequent characters must be uppercase or lowercase English alphanumeric, @, #, \$, or \_.
  - It must be unique to the information catalog.

#### **PHYNAME**

The name of a DB2 table where the Information Catalog Manager stores objects of this type.

If your DB2 tables follow naming conventions, you can use PHYNAME to give the underlying tables in your information catalog a different name from the object type name.

If you do not specify this property, the Information Catalog Manager uses the short name you gave as TYPE.

You can add PHYNAME only if you use a tag language file to create the object type. You cannot add it through the user interface.

#### **CATEGORY**

The category: GROUPING, ELEMENTAL, CONTACT, DICTIONARY, or SUPPORT.

#### **EXTNAME**

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

- 80 character maximum.
- It must not contain null characters.

## Creating your own object types

- It must not be all blank characters.

### ICWFILE

The name of the Windows 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 lines 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:

- 8 character (SBCS) maximum.
- First character must be uppercase or lowercase English alphabetic, @, #, or \$.
- Subsequent characters must be uppercase or lowercase English alphanumeric, @, #, \$, or \_.
- 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 Manager asks you for another name.

**DT** The data type: **C**, **V**, **L**, or **T**.

<b>C (CHAR)</b>	Up to 254 characters
<b>V (VARCHAR)</b>	Up to 4 000 characters
<b>L (LONG VARCHAR)</b>	Up to 32 700 characters
<b>T (TIMESTAMP)</b>	26 characters, in this format: yyyy-mm-dd-hh.mm.ss.nnnnnn

**DL** The size for the property.

### UUISEQ

The position this property has in the UUI: **1**, **2**, **3**, **4**, or **5**. Include this keyword only if you want the property to be part of the UUI.

### NULLS

Entry required?

**N** Entry is required

**Y** Entry is not required

### EXTNAME

The property name. The rules for property names are:



- 80 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 UII for this object type, you can use only the keywords SHRTNAME and UISEQ for the property. The Information Catalog Manager 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 like Figure 3. Figure 3 shows an abbreviated version of the "Relational tables and views" object type, which is one of the predefined object types that are provided with the Information Catalog Manager. The complete object type definition is available in the \SQLLIB\DGWIN\TYPES directory located on the drive where the DB2 Universal Database is installed.

```
COMMENT.-----
COMMENT.Generating the report object definitions.
COMMENT.-----
ACTION.OBJTYPE(MERGE)
OBJECT.TYPE(REPORT) CATEGORY(ELEMENTAL) PHYNAME(REPORTS)
      EXTNAME(Text based reports) ICWFILE(flgnyprep.ico)
PROPERTY. SHRTNAME(NAME)                                UISEQ(0)
PROPERTY. SHRTNAME(SHRTDESC)  DT(V)  DL(250)  UISEQ(0)  NULLS(Y)
      EXTNAME(Short description)
PROPERTY. SHRTNAME(LONGDESC)  DT(L)  DL(32700) UISEQ(0)  NULLS(Y)
      EXTNAME(Long description)
PROPERTY. SHRTNAME(ACTIONS)   DT(V)  DL(254)  UISEQ(0)  NULLS(Y)
      EXTNAME(Actions)
PROPERTY. SHRTNAME(TITLE)     DT(V)  DL(254)  UISEQ(0)  NULLS(N)
      EXTNAME(Report title)
PROPERTY. SHRTNAME(RPRTDATE)  DT(C)  DL(26)   UISEQ(0)  NULLS(Y)
      EXTNAME(Report publication date)
PROPERTY. SHRTNAME(RPRTFRMT)  DT(V)  DL(80)   UISEQ(0)  NULLS(Y)
      EXTNAME(Report presentation format)
PROPERTY. SHRTNAME(DBPRESNT)  DT(V)  DL(254)  UISEQ(0)  NULLS(Y)
      EXTNAME(Report presentation requirements)
PROPERTY. SHRTNAME(OWNER)     DT(V)  DL(80)   UISEQ(0)  NULLS(Y)
      EXTNAME(Report owner)
PROPERTY. SHRTNAME(FILENAME)  DT(V)  DL(254)  UISEQ(1)  NULLS(N)
      EXTNAME(Report filename)
PROPERTY. SHRTNAME(TYPE)      DT(V)  DL(80)   UISEQ(2)  NULLS(N)
      EXTNAME(Report class or type)
PROPERTY. SHRTNAME(URL)       DT(V)  DL(254)  UISEQ(0)  NULLS(Y)
      EXTNAME(URL to access data)
```

Figure 3. Sample tag language file for an object type

## Updating an object type

---

### Updating an object type

You can make only the following changes to an existing object type:

- Change the external name
- Change the representative icon
- Add properties

You can update an object type by using the Information Catalog Manager windows or tag language.

### Updating an object type by using the Information Catalog Manager windows

Start from the Information Catalog window.

1. Right-click the **Object types** icon.
2. Click the **Open as → Icon list**.
3. Right-click the icon of the object type you want to change.
4. Click **Open**.

The Update Object Type window opens.

5. To change the external name, type a new name in the **Object type name** field. The rules for object type names are:
  - 80 character maximum.
  - It must not contain null characters.
  - It must not be all blank characters.
6. The default Windows icon for the selected category is displayed in the **Windows icon** field. To identify a specific icon to represent the object type:
  - Click **Find** to locate a different Windows icon and display it in the window.
7. To add a property to the object type (unless the type is Comments, which cannot be extended), click **Add**.

The Add Property window opens.

- a. Type a name for the property in the **Property name** field. The rules for property names are:
  - 80 character maximum.
  - It must not contain null characters.
  - It must not be all blank characters.
- b. Type a unique property short name in the **Short name** field. The rules for short names are:
  - 8 character (SBCS) maximum.
  - First character must be uppercase or lowercase English alphabetic, @ (at sign), # (number sign), or \$ (dollar sign).
  - Subsequent characters must be uppercase or lowercase English alphanumeric, @ (at sign), # (number sign), \$ (dollar sign), or \_ (underscore).
  - 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 Manager asks you for another name.
- c. From the list, click a data type.
- d. Type a size in the **Size** field.
- e. Click **OK** to add the property.

To close the window without adding a property, click **Cancel**.

This step is not available for AS/400 information catalogs.

8. To change a property you added during the current update action, click it in the **Properties** list, and then click **Modify** or **Remove**.

This step is not available for AS/400 information catalogs.

9. Click **Update** to save your changes in the database.

Your changes display in the Object Types window, but not in other windows until you close and reopen them.

To close the window without updating the object type, click **Cancel**.

### Updating an object type using Information Catalog Manager tag language

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

```
:ACTION.OBJTYPE(UPDATE)
:OBJECT.TYPE(short_name_of_object_type)
```
2. To change the external name, add the following line:
 

```
EXTNAME(new_external_name_of_object_type)
```
3. To change the object type's icon, add the following line:
 

```
ICOFILE(new_OS/2_icon_filename)
ICWFILE(new_Windows_icon_filename)
```

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

<b>Keyword</b>	<b>Value</b>
<b>TYPE</b>	The short name of the object type you are updating.
<b>EXTNAME</b>	The new external name of the object type. The rules for external names are: <ul style="list-style-type: none"> <li>• 80 character maximum.</li> <li>• It must not contain null characters.</li> <li>• It must not be all blank characters.</li> </ul>
<b>ICOFILE</b>	The name of the new OS/2 icon file, including its extension. You give the drive and path information where the icon file exists as part of the <b>IMPORT</b> command when you import your tag language file.

## Updating an object type

### ICWFILE

The name of the new Windows 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.

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

```
:ACTION.OBJTYPE(APPEND)
:OBJECT.TYPE(short_name_of_object_type)
:PROPERTY.SHRTNAME(short_name_of_new_property) DT(data_type) DL(size)
      UISEQ(0) NULLS(y) EXTNAME(external_name_of_new_property)
```

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 UISEQ 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:

- 8 character (SBCS) maximum.
- First character must be uppercase or lowercase English alphabetic, @ (at sign), # (number sign), or \$ (dollar sign).
- Subsequent characters must be uppercase or lowercase English alphanumeric, @ (at sign), # (number sign), \$ (dollar sign), or \_ (underscore).
- 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 Manager asks you for another name.

#### DT

The data type: C, V, L, or T.

#### C (CHAR)

Up to 254 characters

#### V (VARCHAR)

Up to 4 000 characters

#### L (LONG VARCHAR)

Up to 32 700 characters

	<b>T (TIMESTAMP)</b>	Exactly 26 characters, in this format (the line break is not significant)  yyyy-mm-dd-hh.mm.ss.nnnnnn
<b>DL</b>		The size for the property.
<b>EXTNAME</b>		The external name of the property. The rules for property names are: <ul style="list-style-type: none"> <li>• 80 character maximum.</li> <li>• It must not contain null characters.</li> <li>• It must not be all blank characters.</li> </ul>

---

### Deleting an object type

When you delete an object type, all objects of that type are also deleted (unless any objects are Grouping objects that contain objects of a different object type). You can delete an object type from your information catalog by using the Information Catalog Manager windows or tag language.

#### Deleting an object type by using the Information Catalog Manager windows

1. Optional: Search for objects of the object type you want to delete to ensure that you do not want to keep any of them.
2. Right-click on the **Object types** icon in the Information Catalog window.
3. Click **Open as** → **Icon list** .
4. Right-click on the icon of the object type you want to delete.
5. Click **Delete**.

The Delete window opens.

6. Click **Delete** to delete the object type.
7. Click **Yes** to confirm deletion.

When you delete an object type, the Information Catalog Manager closes all windows that are directly related to that object type.

#### Deleting an object type by using the Information Catalog Manager tag language

Enter the following lines in your tag language file:

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

## Deleting an object type

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

**Keyword**

**Value**

**TYPE**

The short name of the object type you are deleting.

---

## Chapter 3. Populating the catalog with information

After you define the object types you need, you populate, or fill, the information catalog with 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.

This chapter describes how to perform the following tasks:

- Create objects
- Copy existing objects
- Update existing objects
- Delete objects

---

### Creating an object

You create objects of various types to represent the actual information available in your organization. You can create objects by using the Information Catalog Manager windows or tag language.

#### Creating an object by using the Information Catalog Manager windows

Start from the Information Catalog window:

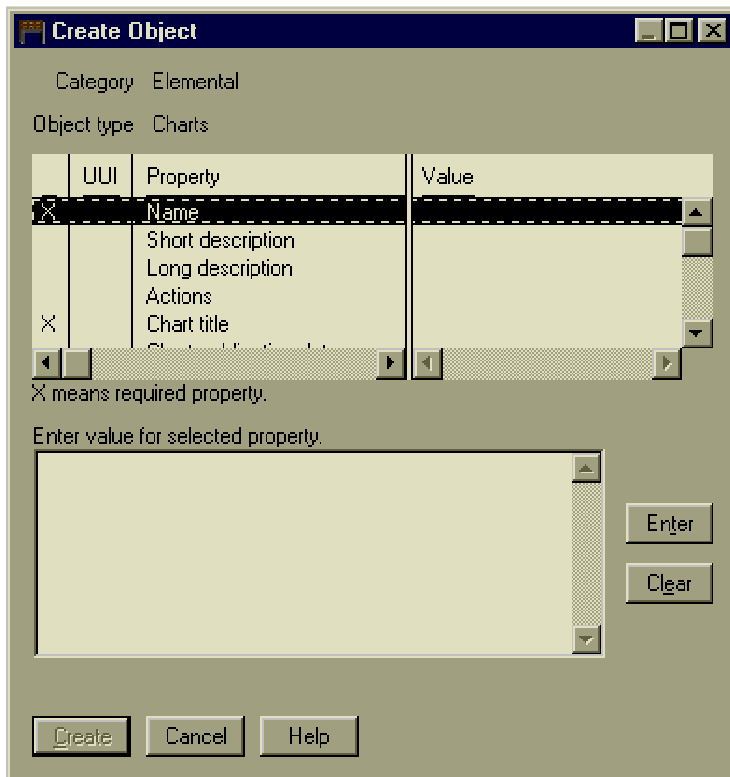
1. Right-click the **Object types** icon in the Information Catalog Manager Catalog window.
2. Click **Open as → Icon list**.
3. Right-click the icon of the object type for which you want to create an object.

You cannot create an object by using either the **Programs** or **Comments** icons. You create a Programs object when you associate a program with an existing object type (see "Associating a program with object types" on page 61). You create a Comments object from an existing object of another object type (see "Associating comments and objects" on page 58).

4. Click **Create object**.

The Create Object window opens.

## Creating an object



5. In the **Properties/Values** list, click a property.
6. Type a value for the property in the **Enter value for selected property** field.
7. Click **Enter** to move the value to the **Value** column in the **Properties/Values** list.  
If you want to erase what you entered in the **Enter value for selected property** field, click **Clear**.
8. Click **Create** when you finish entering values.  
To close the window without creating an object, click **Cancel**.

### Creating an object using the Information Catalog Manager tag language

You can create many objects at the same time by using the Information Catalog Manager tag language. You can include the tag language for creating an object in the same tag language file in which you defined the object type, after the object type definition. The properties can go in any order, and you can omit properties for which you do not have a value.



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

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

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

Keyword	Value
<b>TYPE</b>	The short name of the object type for which you are creating an object.
<b>short_name</b>	The short name of the object type property.

For each object, type the short name of each object type property, followed by a value for the property in parentheses. Figure 4 shows an example of tag language to create an object. The example uses the object type that is created with the definition in Figure 3 on page 37.

```
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://$$$$@/info_cat/db2www/dg_tableviewer.mac/Table_Login?DATABASE=
DGWDATA&TABLE=CUSTOMER&OWNER=USERID)
    SOURCE(DB2 SYSTEM CATALOGS)
```

Figure 4. Creating an object with tag language

---

## Copying an object

You can create a new object that has the values of an existing object. (For information about copying a comment, see “Copying a comment” on page 59.) Start from one of the following windows:

- Search Results
- Collection
- Found In
- Contacts
- Subjects

## Copying an object

Tree View  
Linked With

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

The Copy Object window opens.

3. From the **Properties / Values** list, click a property.
4. Edit the value for the property in the **Enter value for selected property** field.

You must change at least one UUI value for your new object to be unique.

If you want to erase the existing value in the **Enter value for selected property** field, click **Clear**.

5. Click **Enter** to move the changed value to the **Value** column in the **Properties/Values** list.
6. Click **Copy** when you finish changing values.  
To close the window without copying an object, click **Cancel**.

---

## Updating an object

You can change values for an existing object by using the Information Catalog Manager windows or tag language. (For information about updating a comment, see “Updating a comment” on page 59.)

### Updating an object by using the Information Catalog Manager windows

Start from one of the following windows:

Search Results  
Collection  
Found In  
Contacts  
Subjects  
Tree View  
Linked With

1. Right-click on the object you want to update.
2. Click **Update**.

The Update Object window opens.

3. From the **Properties/Values** list, click a property.
4. Edit the value for the property in the **Enter value for selected property** field.

If you want to erase the existing value in the **Enter value for selected property** field, click **Clear**.

5. Click **Enter** to move the changed value to the **Value** column in the **Properties/Values** list.
6. Click **Update** when you finish changing values.

To close the window without updating an object, click **Cancel**.

### Updating an object by using the Information Catalog Manager tag language

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

```
ACTION.OBJINST(UPDATE)
OBJECT.TYPE(short_name_of_object_type)
```
2. Enter the following lines, filling in the UUI properties and property values of the object you want to change:
 

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

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

Keyword	Value
<b>UUI_short_name</b>	The short name of a UUI property of the object type.

Properties and values that are specified after the SOURCEKEY keyword are the UUI. When you created the object type, you defined up to five properties in a certain order to make up the UUI. When you type those properties and values, the Information Catalog Manager 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 short 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 have to include all the object's properties. Any properties you omit will not be updated.

Figure 5 on page 48 shows an example of tag language to update an object. The example uses the object with UUI properties and values that are created in Figure 4 on page 45.

In this example, the value in SHRTDESC is updated.

## Deleting an object

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

Figure 5. Updating an object with tag language

---

## Deleting an object

You can delete an object from your information catalog by using the Information Catalog Manager windows or tag language. (For information about deleting a comment, see “Deleting a comment” on page 60.)

### Deleting an object by using the Information Catalog Manager windows

Start from one of the following windows:

- Search Results
- Collection
- Found In
- Contacts
- Subjects
- Tree View
- Linked With

1. Right-click on the object that you want to delete.
2. Click **Delete**.
  - If the object you are deleting is a Grouping object, the Delete Tree window opens.
  - If the object you are deleting is not a Grouping object, the Delete window opens.
3. (Optional) Deselect any objects in the **Object** list box that you do not want to delete.
4. If you are deleting a Grouping object, you must decide what you want the Information Catalog Manager to do with the objects that are contained in the Grouping object:
  - Delete all the contained objects by selecting the **Delete all underlying objects** check box.
  - Keep all the contained objects, but delete their relationships to the Grouping object by clearing the **Delete all underlying objects** check box.
5. Click **Delete** to delete the object.

The object is deleted from the information catalog.

### Deleting an object by using the Information Catalog Manager tag language

1. To delete a Grouping object and all objects it contains, enter the following line in your tag language file:

```
ACTION.OBJINST(DELETE_TREE_ALL)
```

To delete a Grouping object and all relationships in which it participates, including the underlying tree structure, enter the following line in your tag language file:

```
ACTION.OBJINST(DELETE_TREE_REL)
```

To delete a non-Grouping object, enter the following line in your tag language file:

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

2. Enter the following line, filling in the object type of the object you want to delete:

```
OBJECT.TYPE(short_name_of_object_type)
```

3. Enter the following lines, filling in the UII properties and property values of the object you want to delete:

```
INSTANCE.SOURCEKEY(UII_short_name(value_for_property)
    UII_short_name(value_for_property)
    UII_short_name(value_for_property))
```

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

<b>Keyword</b>	<b>Value</b>
<b>TYPE</b>	The short name of the object type for which you are deleting an object.
<b>UII_short_name</b>	The short name of a UII 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 UII. When you created the object type, you defined certain properties in a certain order to make up the UII. When you enter those properties and values, the Information Catalog Manager 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.

Figure 6 on page 50 shows an example of tag language to delete a Grouping object and all objects it contains. The example uses the object that is created in Figure 4 on page 45.

## Deleting an object

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

*Figure 6. Deleting an object with tag language*

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

---

## Chapter 4. Making the information catalog convenient for users

You can make your information catalog more convenient for your users. With the Information Catalog Manager you can perform the following tasks:

- Group objects by subject area for easy browsing
- Link related objects together
- Add contact names to objects
- Associate comments and objects
- Set up object types to start programs
- Set up glossaries of standard terminology for users
- Provide support and helpful information for users

Grouping objects, linking them, adding contact names to them, and associating comments with them are all ways of establishing relationships between objects. When you begin creating relationships, try to use a top-down approach, both to enhance performance and avoid errors. For example, the object CelDial Marketing Information contains the object Advertising Information, which in turn contains Advertisements on the WWW. Place Advertising Information into CelDial Marketing Information before placing Advertisements on the WWW into Advertising Information.

Especially try to avoid relating an object with one that already has relationships several layers deep.

---

### Grouping objects by subject

You can organize the Information Catalog Manager objects by grouping them together. By nesting the groupings, you can organize your information catalog hierarchically. The Information Catalog Manager shows the highest-level groupings in the Subjects window, represented by the **Subjects** icon in the Information Catalog window.

You can group objects by using the Information Catalog Manager windows or tag language.

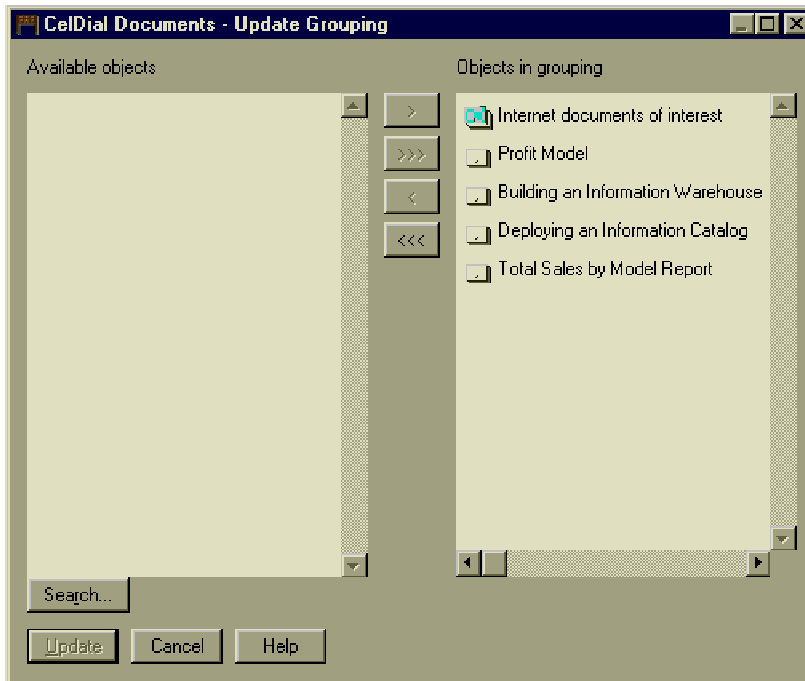
### Grouping objects by subject by using the Information Catalog Manager windows

1. Either create a Grouping category object (see “Creating an object” on page 43 for details) or locate an existing Grouping object in one of the following windows:
  - Search Results
  - Collection

## Grouping objects by subject

Found In  
Subjects  
Linked With  
Tree View

2. Right-click the object.
3. Click **Update grouping**.  
The Update Grouping window opens.



4. To add objects to the grouping:
  - a. Click **Search**.  
The Define Search - Grouping window opens. Use this window to search for objects you want to include. Objects that fit your search criteria are returned to the Update Grouping window in the **Available objects** list.
  - b. From the **Available objects** list, click one or more objects.
  - c. Click > to move selected objects to the **Objects in grouping** list.
5. To remove objects from the grouping:
  - a. From the **Objects in grouping** list, click one or more objects.
  - b. Click < to move selected objects out of the **Objects in grouping** list.
6. Click **Update**.



Changes are displayed as you make them from a Tree view, but you must close and reopen any other window to display your updates.

To close the window without changing the grouping, click **Cancel**.

### Grouping objects by subject by using tag language

To create groupings of the Information Catalog Manager objects with tag language, specify a Contains relationship between an object categorized as Grouping and an object categorized as Grouping or Elemental.

1. To add an object to a grouping, enter the following line in your tag language file:  
`:ACTION.RELATION(ADD)`
2. To delete an object from a grouping, enter the following line in your tag language file:  
`:ACTION.RELATION(DELETE)`
3. Specify the Contains relationship by typing the following lines. Specify the type of the Grouping object for SOURCETYPE and the object type that you want to include in the group for TARGETTYPE:  
`:RELTYPE.TYPE(CONTAIN) SOURCETYPE(short_name_of_object_type)  
TARGETTYPE(short_name_of_object_type)`
4. Type the following lines, filling in the UUI properties and property values of the Grouping object:  
`:INSTANCE.SOURCEKEY(UUI_short_name(value_for_property)  
UUI_short_name(value_for_property)  
UUI_short_name(value_for_property))`
5. Type the following lines, filling in the UUI properties and property values of the object you want to include in the grouping:  
`TARGETKEY(UUI_short_name(value_for_property)  
UUI_short_name(value_for_property)  
UUI_short_name(value_for_property))`

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

<b>Keyword</b>	<b>Value</b>
<b>SOURCETYPE</b>	The short name of the source object type.
<b>TARGETTYPE</b>	The short name of the target object type.
<b>UUI_short_name</b>	The short name of a UUI property of the object type.

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

## Grouping objects by subject

Figure 7 shows an example of tag language to add an object to a Grouping object. The example assumes that you already created the source and target objects.

```
ACTION.RELATION(ADD)
RELTYPE.TYPE(CONTAIN) SOURCETYPE(TABLES) TARGETTYPE(COLUMN)
INSTANCE.SOURCEKEY(DBNAME(DGWDATA) OWNER(USERID)
    TABLE(AR_HISTORY))
    TARGETKEY(DBNAME(DGWDATA) OWNER(USERID) TABLE(AR_HISTORY)
    COLUMN(BAL30))
```

*Figure 7. Adding an object to a grouping with tag language*

In this example, the object identified by DBNAME(DGWDATA) OWNER(USERID) TABLE(AR\_HISTORY) COLUMN(BAL30) is placed in the Grouping object identified by DBNAME(DGWDATA) OWNER(USERID) TABLE(AR\_HISTORY).

---

## Creating a linked relationship between objects

To show users that data represented by one object is related to data that is represented by another object, you create a linked relationship by associating objects. You can link objects by using the Information Catalog Manager windows or tag language.

### Linking objects by using the Information Catalog Manager windows

To create a linked relationship, start from one of the following windows:

- Search Results
- Collection
- Found In
- Subjects
- Tree View
- Linked With

1. Right-click the object you want to associate with other objects in a linked relationship. The object's pop-up menu opens.
2. Click **Update links**.  
The Update Links window opens.
3. Link other objects to the selected object:
  - a. Click **Search**.  
The Define Search - Links window opens. Use this window to search for objects you want to include in the linked relationship with the selected object. Objects that fit your search criteria are returned to the Update Links window in the **Available objects** list.
  - b. From the **Available objects** list, click one or more objects.
  - c. Click > to add selected objects to the **Linked objects** list.
4. To remove objects from the linked relationship:

## Creating a linked relationship between objects

- a. From the **Linked objects** list, click one or more objects.
  - b. Click < to remove selected objects from the **Linked objects** list.
5. When you finish adding and removing objects, click **Update**. All linked relationships are updated.

To close the window without changing any objects, click **Cancel**.

### Linking objects by using tag language

To link associated objects with tag language, specify a link relationship between two objects that are categorized as Grouping or Elemental.

1. To create a link between two objects, enter the following line in your tag language file:  
`:ACTION.RELATION(ADD)`
2. To remove a link between two objects, enter the following line in your tag language file:  
`:ACTION.RELATION(DELETE)`
3. Specify the link relationship by typing the following lines, filling in the type of the two objects you are linking for **SOURCETYPE** and **TARGETTYPE**:  
`:RELTYPE.TYPE(LINK) SOURCETYPE(short_name_of_object_type)  
TARGETTYPE(short_name_of_object_type)`
4. Type the following lines, filling in the UII properties and property values of an object you are linking:  
`:INSTANCE.SOURCEKEY(UII_short_name(value_for_property)  
UII_short_name(value_for_property)  
UII_short_name(value_for_property))`
5. Type the following lines, filling in the UII properties and property values of the other object you are linking:  
`TARGETKEY(UII_short_name(value_for_property)  
UII_short_name(value_for_property)  
UII_short_name(value_for_property))`

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

<b>Keyword</b>	<b>Value</b>
<b>SOURCETYPE</b>	The short name of the source object type.
<b>TARGETTYPE</b>	The short name of the target object type.
<b>UII_short_name</b>	The short name of a UII property of the object type.

Completely enclose in parentheses all the properties and values after the **SOURCEKEY** and **TARGETKEY** keywords.

## Creating a linked relationship between objects

Figure 8 shows an example of tag language to create a linked relationship between two Grouping objects. The example assumes that you already created the source and target objects.

```
ACTION.RELATION(ADD)
RELTYPE.TYPE(LINK) SOURCETYPE(TABLES) TARGETTYPE(TABLES)
INSTANCE.SOURCEKEY(DBNAME(DGWDATA) OWNER(USERID)
    TABLE(COMPONENTS))
    TARGETKEY(DBNAME(DGWDATA) OWNER(USERID) TABLE(CUSTSHIP))
```

*Figure 8. Linking two objects with tag language*

In this example, two relational tables are linked.

---

## Associating contact names with objects

You can help information catalog users in your organization find a person responsible for the actual data that is described by objects. You can identify contacts for objects by using the Information Catalog Manager windows or tag language.

### Associating contact names with objects by using the Information Catalog Manager windows

1. Create a Contact category object.
2. Either create a Grouping or Elemental category object (see “Creating an object” on page 43 for details) or locate an existing Grouping or Elemental object in one of the following windows:
  - Search Results
  - Collection
  - Found In
  - Subjects
  - Tree View
  - Linked With
3. Right-click the Grouping or Elemental object.
4. Select **Contacts** → **Associate**.  
The Associate Contacts window opens.
5. To add a contact to the object:
  - a. Click **Search**.  
The Define Search - Contacts window opens. Use this window to search for contact objects you want to include. Objects that fit your search criteria are returned to the Associate Contacts window in the **Available contacts** list.
  - b. From the **Available contacts** list, select one or more objects.
  - c. Click > to move selected objects to the **Contacts** list.

6. To delete a contact from an object:
  - a. From the **Contacts** list, click one or more objects.
  - b. Click < to move selected objects out of the **Contacts** list.
7. Click **Associate**. You must close and reopen the window for your updates to appear.

To close the window without adding or deleting contacts, click **Cancel**.

### Associating contact names with objects by using tag language

To add a Contact object to another Information Catalog Manager object with tag language, specify a Contact relationship between them.

1. To add a contact, enter the following line in your tag language file:  
:ACTION.RELATION(ADD)
2. To delete a contact, enter the following line in your tag language file:  
:ACTION.RELATION(DELETE)
3. Enter the following lines, filling in the type of the object that is associated with the contact for SOURCETYPE:  
:RELTYPE.TYPE(CONTACT) SOURCETYPE(short\_name\_of\_object\_type)  
TARGETTYPE(short\_name\_of\_contact\_object)
4. Enter the following lines, filling in the UII properties and property values for the object that is associated with the contact:  
:INSTANCE.SOURCEKEY(UII\_short\_name(value\_for\_property)  
UII\_short\_name(value\_for\_property)  
UII\_short\_name(value\_for\_property))
5. Enter the following lines, filling in the UII properties and property values of the Contact object:  
TARGETKEY(UII\_short\_name(value\_for\_property)  
UII\_short\_name(value\_for\_property)  
UII\_short\_name(value\_for\_property))

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

Keyword	Value
<b>SOURCETYPE</b>	The short name of the source object type.
<b>UII_short_name</b>	The short name of a UII property of the object type.

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

Figure 9 on page 58 shows an example of tag language to add a contact to a database object. The example assumes that you already created the source and target objects.

## Associating contact names with objects

```
ACTION.RELATION(ADD)
RELYTYPE.TYPE(CONTACT) SOURCETYPE(DATABASE) TARGETTYPE(CONTACT)
INSTANCE.SOURCEKEY(SERVER(STL11W71) DBNAME(DGWDATA)
    DBTYPE(RELATIONAL))
    TARGETKEY(NAME(Robin Noble-Thomas) RESPONSE(EUI team lead))
```

Figure 9. Adding a contact to a table object with tag language

In this example, the Contact object identified by NAME(Robin Noble-Thomas) RESPONSE(EUI team lead) is added to the object identified by SERVER(STL11W71) DBNAME(DGWDATA) DBTYPE(RELATIONAL).

---

## Associating comments and objects

With the Information Catalog Manager, you can attach comments to objects like you attach a “sticky” note to a page in a book. The note might contain additional information for yourself or other users of the book; you can later remove the note and discard it.

A *comment* is an object that annotates another object. For example, you might attach a comment to a chart object that contains notes about the data in the chart.

### Creating a comment

You can create comments about specific Information Catalog Manager objects. You can use comments as a reminder to yourself or as a way to communicate with other users of your information catalog in your organization.

Start from one of the following windows:

- Search Results
- Collection
- Found In
- Contacts
- Subjects
- Tree View
- Linked With
- Attachments

1. Right-click the object to which you want to attach a comment.  
This step is not necessary if you started from the Attachments window.
2. Click **Attachments** → **Create comment**.  
If you are starting from the Attachments window, click **Attachments** → **Create comment**.  
The Create Comment window opens.
3. Type a name for the comment in the **Name** field.
4. Assign a status to the comment.

5. Optional: In the **Actions** field, specify the actions that you want someone to take based on the text of the comment.
6. Type the complete text of the comment in the **Description** field.
7. Click **Create** to create the comment and attach it to the specified object.  
To close the window without creating a comment, click **Cancel**.

### Copying a comment

You can copy existing comments to create unattached comments, which you can later attach to an object.

Start from one of the following windows:

Search Results  
Collection  
Attachments

1. Right-click the comment that you want to copy.
2. Click **Copy**.  
The Copy Comment window opens.
3. (Optional) Change any of the following values:  
Name  
Actions  
Status  
Description

4. Click **Copy** to create the copied comment.  
To close the window without copying a comment, click **Cancel**.

### Updating a comment

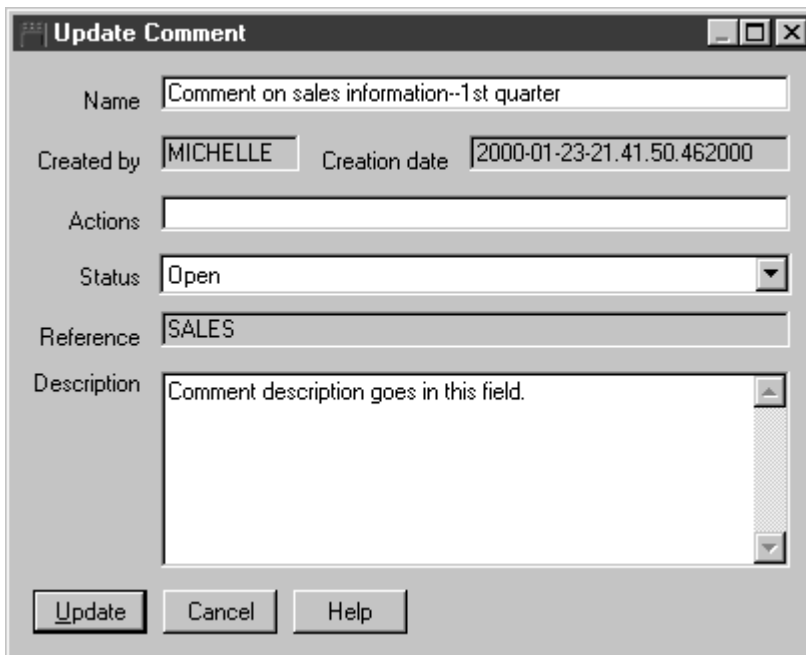
You can change various values, and the text, of existing comments.

Start from one of the following windows:

Search Results  
Collection  
Attachments

1. Right-click the comment you want to update.
2. Click **Update**.  
The Update Comment window opens.

## Associating comments and objects



The screenshot shows a dialog box titled "Update Comment". It contains the following fields and controls:

- Name:** A text box containing "Comment on sales information--1st quarter".
- Created by:** A text box containing "MICHELLE".
- Creation date:** A text box containing "2000-01-23-21.41.50.462000".
- Actions:** An empty text box.
- Status:** A dropdown menu currently showing "Open".
- Reference:** A text box containing "SALES".
- Description:** A large text area containing the text "Comment description goes in this field.".
- Buttons:** Three buttons at the bottom: "Update", "Cancel", and "Help".

3. Change at least one of the following values:

- Name
- Actions
- Status
- Description

4. Click **Update**.

To close the window without updating the comment, click **Cancel**.

### Deleting a comment

You can delete an existing comment, whether or not it is attached to an object.

Start from one of the following windows:

- Search Results
- Collection
- Attachments

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

2. Click **Delete**.

The Delete window opens.

3. (Optional) Deselect any comments in the **Object** list that you do not want to delete.

4. Click **Delete** to delete the comment.



The comment is deleted from the information catalog.

### Attaching and detaching comments and objects

You can attach existing unattached comments to specified objects. You can also detach comments from specified objects. To attach and detach existing comments, start from one of the following windows:

- Search Results
- Collection
- Found In
- Contacts
- Subjects
- Tree View
- Linked With

1. Right-click the object for which you want to change the Attachment relationships.
2. Click **Attachments** → **Associate comments**.  
The Associate Comments window opens.
3. To attach additional comments to the object:
  - a. In the **Available comments** list, click one or more comments that you want to attach to the selected object.
  - b. Click > to move the selected comments to the **Current comments** list.
4. To detach comments from the object:
  - a. In the **Current comments** list, select one or more comments that you want to detach from the selected object.
  - b. Click < to move the selected comments to the **Available comments** list.
5. Click **Associate** to save the specified Attachment relationships.  
To close the window without changing the Attachment relationships, click **Cancel**.

---

### Associating a program with object types

The Information Catalog Manager makes it easy to start a program that can retrieve the actual data that an object describes. For example, you might have objects that describe graphic charts. You can set up a graphic program, such as CorelDRAW!, so that you can retrieve the actual charts for editing, copying, or printing.

The Information Catalog Manager for Windows can start any program that runs on the Windows system that you are using, or that you can start from an MS-DOS command prompt.

## Associating a program with object types

A single object type can start more than one program (for example, the object type Spreadsheet can have both Lotus 1-2-3 and Microsoft Excel associated with it).

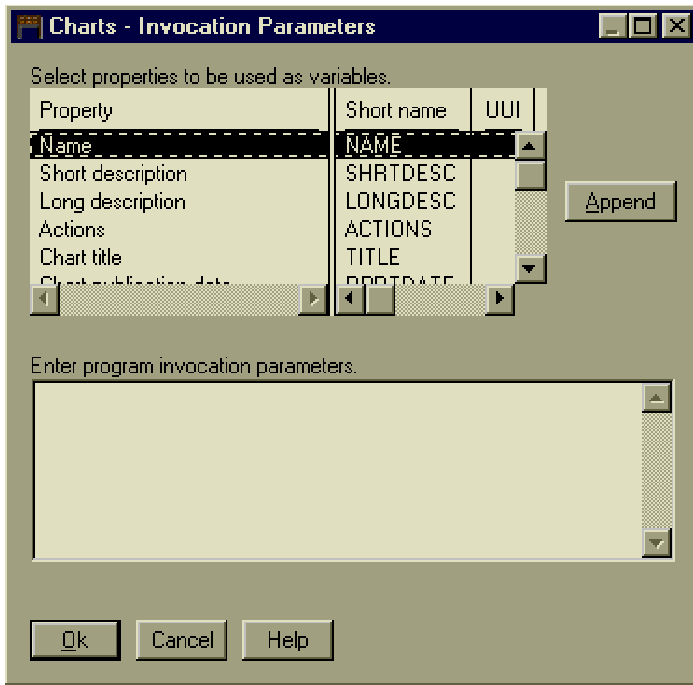
To enable an object to start a program, you create an association between a Programs object and any object type that is not categorized as Program.

### Creating a Programs object

You can create a Programs object by using the Information Catalog Manager windows or tag language.

#### Creating a Programs object by using the Information Catalog Manager windows

1. Right-click the **Object types** icon in the Information Catalog window.
2. Click **Open as** → **Icon list**.
3. Right-click the icon of the object type for which you want the program to start.
4. Click **Associate programs**.  
The Programs window displays a list of programs currently associated with the selected object type.
5. Click **Add**.  
The Add Program window opens.
6. From the **Properties/Values** list, click a property.
7. In the **Enter value for selected property** field, type a value for the property.  
For more specific information about entering these values, see “Supplying values for required Programs object properties” on page 63.
8. Click **Enter** to move the value to the **Value** column in the **Properties/Values** list.  
If you want to erase what you entered in the **Enter value for selected property** field, click **Clear**.
9. Click **Parameters** to specify the properties whose values you want to use as program parameters. The Invocation Parameters window opens.



10. Click **Add**.

To close the window without adding a program, click **Cancel**.

**Supplying values for required Programs object properties:** Programs objects have several required properties that allow you to differentiate among Programs objects when you want the same program to handle more than one object type. These properties are described in Table 11.

*Table 11. Entering values for required properties when associating a program with an object type*

Property	Example	Description
Name	View home page with Netscape Navigator	This value is displayed in the <b>Select one or more programs to start</b> list box when a user chooses to start a program from an object. When you are associating one program with several object types, you can enter the same value for the Name property on each.

## Associating a program with object types

Table 11. Entering values for required properties when associating a program with an object type (continued)

Property	Example	Description
Class	Browser	You can enter any values for these properties that help you classify and identify the Programs object. You can enter the not-applicable symbol if you do not have a value for a property. (The not-applicable symbol is a hyphen unless you identified a different symbol when you created the information catalog.)
Qualifier 1	Navigator	
Qualifier 2	Windows NT	
Qualifier 3	3.0	
Identifier	start netscape.exe	
<b>Note:</b> 1. The combination of the Class, Qualifiers 1, 2, and 3, and the Identifier properties must be unique across for all program objects in the information catalog. Each instance of an object type must be different.		

For the property that is called Start by invoking, enter the file name of the program and the recommended starting parameters. For Windows NT, Windows 95, and Windows 98, the recommended starting parameter is START filename.exe. The PATH statement must contain the directory where the program is located.

If the file name of the program is in high-performance file system (HPFS) format and contains blanks, surround the path and file name of the program with two sets of quotation marks, as in this example:

```
""D:\PROGPATH\My Program.EXE""
```

If the program name contains blanks, you cannot specify any other start options for the Start by invoking property. Instead, enter the options in the Parameters property. Do not change the value of the HANDLES property.

### Creating a Programs object by using the Information Catalog Manager tag language

Enter the following lines in your tag language file:

```
ACTION.OBJINST(ADD)
OBJECT.TYPE(PROGRAMS)
INSTANCE.NAME(name_of_program)
    UUICLASS(class_of_program)
    UUIQUAL1(identifier)
    UUIQUAL2(identifier)
    UUIQUAL3(identifier)
    UUIIDENT(identifier)
```

## Associating a program with object types

HANDLES(short\_name\_of\_object\_type)  
STARTCMD(command\_to\_start\_program)  
PARMLIST(list\_of\_program\_parameters)  
SHRTDESC(description\_of\_program)

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

<b>Keyword</b>	<b>Value</b>
<b>NAME</b>	The external name (up to 80 characters) of the program.
<b>UUICLASS</b>	A classifying property, such as Spreadsheet.
<b>UIQUAL1, 2, 3</b>	Additional identifying properties.
<b>UIIDENT</b>	Additional identifying property.
<b>HANDLES</b>	The short name of the object type that this Programs object handles. This property is required.
<b>STARTCMD</b>	The command that is required to start the program. This property is required. You do not need to start the command processor (command.exe) as part of the value of the STARTCMD keyword.
<b>PARMLIST</b>	The parameters you want to start the program with.
<b>SHRTDESC</b>	A short description of the program.

Figure 10 shows an example of tag language that sets up a program to handle spreadsheet objects. The example assumes that you have the object type SPRDSHET in your information catalog.

```
ACTION.OBJINST(ADD)
OBJECT.TYPE(PROGRAMS)
INSTANCE.NAME(Lotus 1-2-3 for Windows)
  UUICLASS(SPRDSHET)
  UIQUAL1(Lotus 1-2-3)
  UIQUAL2(Windows)
  UIIDENT(123w.exe)
  HANDLES(SPRDSHET)
  STARTCMD(start /f /win 123w.exe)
  PARMLIST(%LISTSRCE%)
  SHRTDESC(Lotus 1-2-3 for Windows)
```

*Figure 10. Setting up a program to handle spreadsheet objects*

## Associating a program with object types

### Copying a program that is associated with an object type

You can create a program association that is based on values of an existing association:

1. Right-click the **Object types** icon in the Information Catalog window.
2. Click **Open as** → **Icon list**.
3. Click the icon of the object type from which you want to copy the program association.
4. Click **Associate programs**.  
The Programs window displays a list of programs currently associated with the selected object type.
5. Select the program you want to copy.
6. Click **Copy**.  
The Copy Program window opens.
7. From the **Properties/Values** list, click a property.
8. In the **Enter value for selected property** field, edit the value for the property.  
If you want to erase the existing value in the **Enter value for selected property** field, click **Clear**.
9. Click **Enter** to move the changed value to the **Value** column in the **Properties/Values** list.
10. Click **Parameters** to update the list of properties whose values you want to use as program parameters.
11. Click **Copy**.  
To close the window without copying the Programs object, click **Cancel**.

### Updating a program association for an object type

You can change values for an existing association between a program and objects of a specified object type by using the Information Catalog Manager windows or tag language.

#### Updating a program association for an object type by using the Information Catalog Manager windows

1. Right-click **Object types** icon in the Information Catalog window.
2. Click the **Open as** → **Icon list**.
3. Right-click the icon of the object type for which you want to update the program association.
4. Click **Associate programs**.  
The Programs window displays a list of programs currently associated with the selected object type.
5. Click the program you want to update.
6. Click **Update**.

## Associating a program with object types

The Update Program window opens.

7. From the **Properties/Values** list, click a property.
8. In the **Enter value for selected property** field, edit the value for the property.  
If you want to erase the existing value in the **Enter value for selected property** field, click **Clear**.
9. Click **Enter** to move the changed value to the **Value** column in the **Properties/Values** list.
10. Click **Parameters** to update the list of properties whose values you want to use as program parameters.
11. Click **Update** when you finish changing values.  
To close the window without updating the Programs object, click **Cancel**.

### Updating a program association by using tag language

You can update programs that handle objects by using the Information Catalog Manager tag language. You do this the same way you update other objects with tag language. See “Updating an object” on page 46 for information.

### Disassociating a program from an object type

You can delete the association between a program and objects of a specified object type by using the Information Catalog Manager windows or tag language.

#### Disassociating a program from an object type by using the Information Catalog Manager windows

1. Right-click the **Object types** icon in the Information Catalog window.
2. Click **Open as → Icon list**.
3. Right-click the icon of the object type for which you want to delete the program association.
4. Click **Associate programs**.

The Programs window displays a list of programs currently associated with the selected object type.

5. From the list, click the program you want to delete.
6. Click **Delete**.

#### Disassociating a program from an object type by using tag language

To delete the association between a Programs object and an object type by using tag language, delete the Programs object that handles the particular object type:

1. Enter the following lines in your tag language file:  
ACTION.OBJINST(DELETE)  
OBJECT.TYPE(PROGRAMS)

## Associating a program with object types

2. Enter the following lines, filling in the UII properties and property values of the object you want to delete:

```
INSTANCE.SOURCEKEY(UIICLASS(class_of_program)
    UIIQUAL1(identifier)
    UIIQUAL2(identifier)
    UIIQUAL3(identifier))
```

Enter only UII properties for which you have existing values in the information catalog. Completely enclose in parentheses all the properties and values after the SOURCEKEY keyword.

---

## Creating a dictionary facility

You can set up an icon where users can quickly find definitions or synonyms of the business terms that you use in your information catalog, such as a dictionary, glossary, thesaurus, or synonym finder.

When you set up this icon for the first time, it is displayed in every user's Information Catalog window as a saved search. Whenever users want to know the name for an item in the information catalog, they can double-click the saved search icon to view the glossary.

The Information Catalog Manager comes with a Glossary object type.

To create a dictionary facility:

1. Create a Dictionary category object type. See "Creating your own object types" on page 28 for information on creating object types.
2. Create new objects of this object type.
3. Close the information catalog.

When you log on to the Information Catalog Manager again, the new dictionary facility is displayed in your Information Catalog window as a saved search. The new dictionary facility is also displayed in every user's Information Catalog window.

---

## Creating a support facility

You can set up an icon to provide users with any kind of support or helpful information about your information catalog.

You might use this icon to inform users of new items in the information catalog or announce planned updates to the information catalog.



When you set up this icon for the first time, it is displayed in every user's Information Catalog window as a saved search. Whenever users want to view current entries in the support facility, they can double-click on the saved search icon to view a list of items.

The Information Catalog Manager comes with a News object type.

To create a support facility:

1. Create a Support category object type. See "Creating your own object types" on page 28 for information on creating object types.
2. Create new objects of this object type.
3. Close the information catalog.

When you log on to the Information Catalog Manager again, the new support facility is displayed in your Information Catalog window as a saved search. The new support facility is also displayed in every user's Information Catalog window.

## Creating a support facility

---

## Chapter 5. Expanding and automating your information catalog

As you build your information catalog, you will probably find that you need ways to expand it and automate your processes for maintaining it. You might want to exchange objects with other information catalogs or combine the contents of one information catalog with another. For example, you might decide to centralize some of your metadata, or make the contents of your information catalog available to another organization. You might want to exchange Information Catalog Manager metadata with metadata from other products.

The Information Catalog Manager provides tag language for working with large amounts of descriptive data at one time and for exchanging objects among information catalogs to coordinate multiple information catalogs.

- A tag language file can contain descriptive data that you:
  - Export from another information catalog (see “Exporting metadata” on page 79)
  - Extract from another source (see “Extracting descriptive data from other sources” on page 72)
  - Write using any word processing program (see “Performing Information Catalog Manager tasks with the user interface or tag language” on page ix)
  - Exchange with products that generate Information Catalog Manager tag language (see “Publishing and synchronizing metadata” on page 85)
  - Exchange with another product that produces metadata that conforms to the Metadata Interchange Specification (MDIS) (see “Exchanging MDIS-conforming metadata with other products” on page 96)
- A tag language file can comprise deletions you logged in one information catalog that you want to include in other information catalogs (see “Logging deletions from your information catalog” on page 77).

You can import a tag language file that contains descriptive data that you want to include in an information catalog. See “Importing tag language files” on page 78 for more information.

## Extracting descriptive data from other sources

---

### Extracting descriptive data from other sources

The easiest way to fill your information catalog with descriptive data is to use existing descriptions of your organization's information. Many databases and desktop applications already contain valuable descriptive data that you can transfer to your information catalog.

Using the Information Catalog Manager tag language, you can:

- 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

Extracting descriptive data also makes updating or refreshing your information catalog easier. You can edit the tag language files by using any word processing program that can import and export ASCII text files.

### Extracting descriptive data with the information catalog extract programs

The Information Catalog Manager comes with a set of programs that can extract descriptive data from various sources. You can install these extract programs when you install the Information Catalog Manager or at any time. The Information Catalog Manager for Windows places them in `\SQLLIB\SAMPLES\DGEXTxxx` subdirectories.

To use the extract programs, you must be familiar with the specific operating environments in which they run. See "Appendix A. Information Catalog Manager extract programs" on page 113 for information on running the specific extract programs you need.

### Writing customized descriptive data extract programs

You might need to write an extract program if you want to copy data from existing catalogs other than the ones for which the Information Catalog Manager provides extract programs. Your extract program must translate this descriptive data into the Information Catalog Manager tag language. You can then import your descriptive data directly into the Information Catalog Manager as object types and objects.

### Planning your extract program input and output

The format of your existing data is the default input format. Your extract program must produce a tag language file as output. However, the properties you decide to include about your information source determine the tag language output your extract program needs to produce.

This output tag language file can contain some or all of the following tags:

**:DISKCNTRL.** Identifies the sequential number of the current

## Extracting descriptive data from other sources

	diskette, and specifies whether this file continues on more diskettes
<b>:ACTION.</b>	Specifies that an action (add, update, delete, append, or merge) occurs involving an object type, object, or relationship
<b>:OBJECT.</b>	Identifies an object type and its properties
<b>:PROPERTY.</b>	Identifies a property for the object type that you are defining
<b>:INSTANCE.</b>	Identifies an object or relationship
<b>:RELTYPE.</b>	Identifies the type of relationship that you are adding or deleting
<b>:COMMIT.</b>	Identifies an information catalog database commit point
<b>:COMMENT.</b>	Allows you to add comments to the tag language file
<b>:NL.</b>	Allows you to include multi-line property values (for non-UII properties)
<b>:TAB.</b>	Allows you to insert tabs in non-UII property values

For detailed information about the Information Catalog Manager tag language, see “Appendix D. Tag language” on page 147 and “Appendix E. What a tag language file should look like” on page 181.

### Formatting your output tag language file

If you store your tag language file on one or more diskettes, your extract program must place a `:DISKCNTL.` tag at the beginning of the tag language file so that the Information Catalog Manager knows how many diskettes contain your file.

For example, on the first diskette, specify:

```
:DISKCNTL.SEQUENCE(1, +)
```

The `:DISKCNTL.` tag must be the first tag in the file on each diskette.

### Creating object types and objects with an extract program

In the Information Catalog Manager, descriptive data is stored as properties of an object. An object type describes a set of properties that each object has. If you extract descriptive data that includes a set of undefined properties, your extract program must produce tag language that creates an object type.

## Extracting descriptive data from other sources

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 tags that are shown in Figure 11.

```
: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)
```

Figure 11. Tag language output from an extract program to create an object type

When you generate a new object type, you must specify at least one property that has the UUISEQ option with a value of 1. You can specify up to four 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 UUI that uniquely identifies an object to an information catalog.

The 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, and write the tag language file to generate three objects of the MYTABLE object type.

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

## Extracting descriptive data from other sources

Your extract program must produce the tags that are shown in Figure 12 , which you must insert 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)
```

Figure 12. Tag language output for the object type MYTABLE

**Reusing existing object type definitions:** The tag language used to produce the object type definitions shipped with the Information Catalog Manager is available for you to copy or include in your own extract programs. The tag language is in the `SQLLIB\DGWIN\TYPES` directory located on the drive where you installed the DB2 Universal Database. For information on other metadata templates included with the Data Warehouse Center, see the *Data Warehouse Center Application Integration Guide*.

### Merging object types and objects

You will probably use your extract program many times to extract descriptive data for refreshing your information catalog. In this case, it is important to merge your object types and objects so that you do not add new objects every time you import the tag language file.

If two information catalogs contain the same object types, you must merge the object types before you can merge any objects. You should also merge object types if you are not sure whether the information catalogs contain the same object types.

For example, you want to import the contents of an information catalog (the source) that contains a Table object type into another information catalog (the target) that also contains a Table object type. You are not sure if the object types have the same properties. Object-type definitions must be alike for you to merge objects. If the object types have the same properties, you can merge them without problems. If they don't, the UUI properties must be identical,

## Extracting descriptive data from other sources

and all the properties of the object type in the source information catalog must match properties of the object type in the target information catalog.

If the object type in the source information catalog has more properties than the object type in the target information catalog, you can update the object type in the target information catalog before you merge the two.

Your extract program must create tags to merge object types and objects, as shown in Figure 13. The Information Catalog Manager does not update an object type's external name or its icon in the process of merging.

```
:ACTION.OBJTYPE(MERGE)
:OBJECT.TYPE(short_name_of_object_type)
    CATEGORY(category_of_object_type)
    EXTNAME(extended_name_of_object_type)
    ICOFILE(OS/2_icon_file_name)
    ICWFILE(Windows_icon_file_name)
:PROPERTY.SHRTNAME(short_name) DT(data_type) DL(size)
    UUISEQ(position_in_UUI) NULLS(y_or_n) EXTNAME(extended_name)
```

Figure 13. Tag language output for merging object types

### Restriction:

You cannot merge the Programs or the Comments object types.

To merge objects, your extract program must create the tags that are shown in Figure 14.

```
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(short_name_of_object_type)
:INSTANCE.NAME(extended_name_of_object)
    UUI_short_name(value_for_property)
    .
    .
    short_name(value_for_property)
    .
    .
```

Figure 14. Tag language output for merging objects

### Committing changes to the information catalog database

When the Information Catalog Manager imports a tag language file, it stores the descriptive data defined in the tag language file in the information catalog database. However, the Information Catalog Manager does not automatically commit these additions to its database until it reaches the end of a tag language file. The Information Catalog Manager looks for a checkpoint tag (:COMMIT.), which tells it to issue a commit call to the database. If the tag



## Extracting descriptive data from other sources

language file is long, your extract program should place `:COMMIT.` tags at regular intervals in the tag language file. Each checkpoint tag should have an identifier that is unique in your tag language file.

For example, you might want your extract program to include a `:COMMIT.` tag after the complete specification of an `:ACTION.` tag (following all the information that must be specified for an action). Your program can place a `:COMMIT.` tag in the tag language file after it generates all the tags to create an object type or an object.

If the Information Catalog Manager fails to import part of your tag language file, you need to import the rest of your file, starting at a successful checkpoint. The Information Catalog Manager issues a rollback to the database to the point of the last checkpoint. When you try to import this file again, the Information Catalog Manager starts from the checkpoint that matches the ID of the last committed checkpoint. For example, you might want your program to put the following checkpoint into the tag language file after the tags that create the object type:

```
:COMMIT.CHKPID(objtype1)
```

### Restrictions on extract programs

When your extract program generates values for the properties, the Information Catalog Manager 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 Manager 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 extract program generates tag language for values containing 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 the Information Catalog Manager imports a tag language file, it ignores any characters with a hexadecimal value less than `X'20'`. Your descriptive data can contain only alphanumeric characters or timestamps.

---

## Logging deletions from your information catalog

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, to shadow information catalogs in a distributed environment.

## Logging deletions from your information catalog

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, especially if you delete Grouping objects.

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

1. Click **Catalog** → **Manage delete history**.

The Manage Delete History window opens.

2. Select the **Record delete history** check box to begin or continue logging deletions.

To stop logging deletions, clear the **Record delete history** check box.

3. Optional: To copy the existing log of deletions to a tag language file:

- a. Select the **Transfer to a tag language file** check box.

- 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.

4. Optional: To erase the current log of deletions, select the **Reset the delete history** check box.

5. Click **OK** to save your selections.

To close the window without changing delete history selections, click **Cancel**.

---

## Importing and exporting tag language files

You can import the Information Catalog Manager tag language files into your information catalog.

You can also export the Information Catalog Manager objects and object types in the form of tag language files from your information catalog.

For information about importing and exporting tag language files that conform to the Metadata Interchange Specification (MDIS), see “Exchanging MDIS-conforming metadata with other products” on page 96.

### Importing tag language files

#### Important information about importing:

- You can import actions, such as deletions from an information catalog in the form of a delete history tag language file. 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, especially if you delete Grouping objects or object types. If you plan to import a delete history tag language file, ensure that you are not currently logging deletions. The deletions will be recorded during import

## Importing and exporting tag language files

and affect import performance. For information about delete history, see “Logging deletions from your information catalog” on page 77.

- Importing a tag file that was created using an OS/2® editor and that contains accented characters creates unreadable data in the information catalog.

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

1. Click **Catalog** → **Import**.

The Import window opens.

2. In the **Import path and filename** field, type the directory path and file name of the tag language file you want to import.

If you type only the file name, the Information Catalog Manager assumes the tag language file is on the drive and path pointed to by the DGWPATH environment variable.

3. In the **Icon path** field, type the directory path from which you want to import icon files.

4. To change the file destination for messages that are generated during import, in the **Log path and filename** field, type a new directory path and file name. If you type only the file name, the Information Catalog Manager places the log file on the drive and path pointed to by the DGWPATH environment variable.

5. Indicate from where you want to begin importing the tag language file .

- Click **Start at beginning** to start at the beginning.
- Click **Start at checkpoint** to start at the last point at which the Information Catalog Manager successfully committed changes to the information catalog.

6. Click **Import** 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**.

### Exporting metadata

To export the Information Catalog Manager objects from your information catalog to a tag language file, start from one of the following windows:

Search Results  
Collection  
Found In  
Contacts  
Subjects  
Tree View  
Attachments  
Linked With

## Importing and exporting tag language files

1. Right-click on the objects you want to export. On Windows systems, hold down the Shift key to select multiple objects.
2. Right-click on any one of the selected objects.
3. Click **Export**.

The Export window opens.

4. In the **Export path and filename** field, type the directory path and file name of the tag language file to which you want to export the selected objects.

If you type only the file name, the Information Catalog Manager assumes that the tag language file is on the drive and path pointed to by the DGWPATH environment variable.

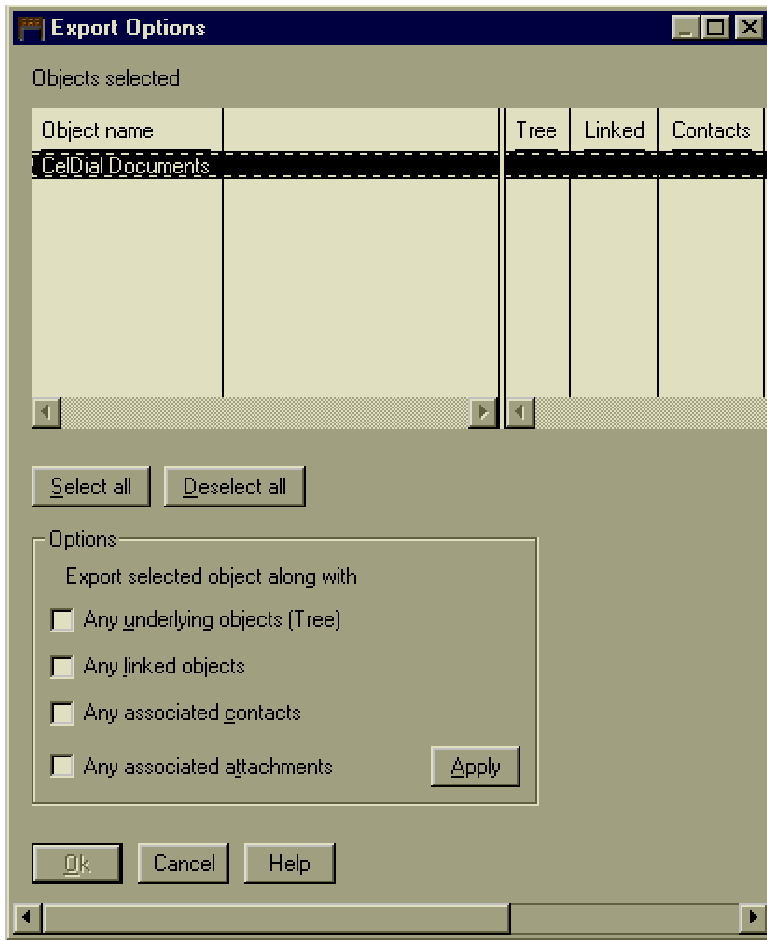
Give a new name to the export tag language file each time you export the Information Catalog Manager objects. The Information Catalog Manager does not append to, or write over, export tag language files.

5. In the **Icon path** field, type the directory path to which you want to export icon files.

If icon files already exist in the path, the Information Catalog Manager replaces them with the new icons.

6. To change the file destination for messages that are generated during export, type a new directory path and file name in the **Log path and filename** field. If you type only the file name, the Information Catalog Manager places the log file on the drive and path pointed to by the DGWPATH environment variable.

7. (Optional) You specified the default export options for all objects on the **Export** page of the Information Catalog Manager Settings notebook. To specify individual export options for selected objects, click on **Options** to open the Export Options window.



Use this window to identify additional related objects to export:

- From the **Objects selected** list, click one or more objects.
- In the **Options** list, select one or more options for the set of selected objects:

- |                                      |  |
|--------------------------------------|--|
| <b>All underlying objects (Tree)</b> | Exports a Grouping object and all the objects it contains.             |
| <b>Any linked objects</b>            | Exports an object and all objects linked to it.                        |
| <b>Any associated contacts</b>       | Exports an object and all Contact objects that are associated with it. |
| <b>Any associated attachments</b>    | Exports an object and all comments attached to it.                     |

## Importing and exporting tag language files

- c. To set the export options for the selected objects, click **Apply**.
  - d. Optional: Repeat steps 7a on page 81 through 7c for different objects.
  - e. Click **OK**.
8. Click **Export** to begin exporting the specified objects.
- The Export window remains open with a progress indicator. A message indicates when the export process is complete.
- To close the window without exporting objects, click **Cancel**.

### Solving import and export problems

Sometimes errors in your tag language files can stop the import process. When this happens, you can look in two files, the echo and log files, that the Information Catalog Manager creates during the process, to see what caused the problem.

The echo file records the tag language lines the Information Catalog Manager has processed. The log file records what happens during the import or export processes.

#### Reading the echo file

The Information Catalog Manager gives the echo file the name of the tag language file, plus an extension of ECH. For example, if you are importing TABLEOBJ.TAG (a tag language file), TABLEOBJ.ECH is the name of the echo file. The Information Catalog Manager automatically places the echo file on the drive and path pointed to by the DGWPATH environment variable.

The echo file contains uncommitted changes to the information catalog, so the tags of an echo file tell you which line in your tag language file stopped the import process. Figure 15 on page 83 shows an example of an echo file.

## Importing and exporting tag language files

```
:COMMENT. -----  
:COMMENT. Create Lotus Notes™ Object  
:COMMENT. -----  
:ACTION.OBJTYPE(MERGE)  
:OBJECT.TYPE(LOTNOTE)  
        CATEGORY(GROUPING)  
        EXTNAME(Lotus Notes Database)  
        PHYNAM(LOTUSDB)  
        ICOFILE(lotusico)  
:PROPERTY.SHRTNAME(LNSERVER)  
        DT(C) DL(30) UUISEQ(1) NULLS(N) EXTNAME(Lotus Notes Server Name)  
:PROPERTY.SHRTNAME(DBNAME)  
        DT(C) DL(15) UUISEQ(2) NULLS(N) EXTNAME(Database filename)  
:PROPERTY.SHRTNAME(MANAGERS)  
        DT(T) DL(50) UUISEQ(0) NULLS(Y) EXTNAME(Managers)  
:PROPERTY.SHRTNAME(POLICY)  
        DT(V) DL(4100) UUISEQ(0) NULLS(Y) EXTNAME(Policy Information)  
:PROPERTY.SHRTNAME(NAME)  
        UUISEQ(3)  
:COMMIT.CKPID(Add Instances)
```

Figure 15. Echo file showing tag language lines that the Information Catalog Manager processed

In Figure 15, the object type LOTNOTE was created with five properties. The import process ended after the checkpoint. From here, you can look in the log file to find error messages about the cause.

### Reading the log file

The log file includes the times and dates when the process started and stopped. It also includes any error messages for problems that occurred during the process.

You can specify a name for the log file or allow the Information Catalog Manager to give it the name of your tag language file plus an extension of LOG. For example, if you are importing TABLEOBJ.TAG (a tag language file), TABLEOBJ.LOG is the name of the log file.

You can specify a drive and path where the Information Catalog Manager places the log file. If you type only the file name, the Information Catalog Manager places the log file on the drive and path pointed to by the DGWPATH environment variable.

Figure 16 on page 84 shows an example of a log file.

## Importing and exporting tag language files

```
*****
Import started: Tag language file -- h:\rxlnotes\dg2lot.tag
1996/9/28 16:10:30
*****
FLG0505: Unable to create object type LOTNOTE. Reason code is 34508.
Extended code is 8.
Import terminated with error(s). The database has been rolled back to
either the last COMMIT tag executed or the beginning of the tag language file
*****
Import ended: Tag language file -- h:\rxlnotes\dg2lot.tag
1996/9/28 16:10:39
*****
```

*Figure 16. Log file showing reason and extended codes for problems the Information Catalog Manager encountered while importing*

In this example, the Information Catalog Manager could not create the object type `LOTNOTE`, as indicated by reason code 34508. The explanation for the reason code (in the *DB2 Universal Database Message Reference*) is as follows:

The length value is invalid for the indicated property in the definition area because of the defined data type.

The book also states that the extended code indicates the sequence number of the property that caused the error.

The property specified by the extended code (property 8) has a length that is not valid for the data type. Because the Information Catalog Manager generates the first five properties of all object types (`OBJTYPID`, `INSTIDNT`, `NAME`, `UPDATIME`, and `UPDATEBY`), the eighth property is the `MANAGERS` property.

In the echo file, you can see that the length for this property is 50, but the type is T (for `TIMESTAMP`). The property called `Managers` cannot have a data type of `TIMESTAMP`, so the correct data type is C (for `CHAR`). Edit your tag language file to correct the data type and restart the import from the last checkpoint.

Place frequent commit checkpoints in the file so that the Information Catalog Manager only rolls back to the last checkpoint. See “Committing changes to the information catalog database” on page 76 for more information about commit checkpoints.



---

## 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 process for the following products:

- The Data Warehouse Center
- The DB2 OLAP Server
- The DB2 OLAP Integration 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.

This chapter also describes how to exchange metadata with any product that conforms to the Metadata Interchange Specification (MDIS).

---

### Publishing and synchronizing 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. For step-by-step information on how to use the Data Warehouse Center user interface to publish metadata, see the online help for the Publish Metadata window. You use a command interface to publish DB2 OLAP Integration Server metadata; see “Preparing to publish OLAP server metadata” on page 87 for more information.

To publish and synchronize metadata in the information catalog, you must complete the following tasks, which are discussed later in this chapter:

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.

#### How metadata is synchronized

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.

## Publishing and synchronizing metadata

You use a command interface to publish DB2 OLAP Integration Server metadata. However, you cannot synchronize updates to the metadata. You must use the command interface to publish metadata again if you want to change it in the information catalog.

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 new object is created in the Data Warehouse Center
- When a new object is created in the Essbase outline
- When the name of an object that you previously published to the information catalog changes

If you plan to synchronize metadata, you must use the Data Warehouse Center warehouse control database as your information catalog database.

### **Before you begin: Establishing the environment for publishing metadata**

Before you begin the steps for publishing and synchronizing metadata, you need to ensure that you established the correct environment.

1. Ensure that you installed and configured the necessary warehouse components on the correct workstations.

*For publishing Data Warehouse Center metadata to the Information Catalog Manager:*

- The Information Catalog Manager 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.
- The Data Warehouse Center administrative interface must have DB2 connectivity to the information catalog APIs.

*For publishing OLAP server metadata to the Information Catalog Manager:*

- Both the Information Catalog Manager administrator function and Essbase client must be installed locally on the warehouse agent site.
  - The Windows NT 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 NT Control Panel.
2. Ensure that both the administrator user IDs for the Information Catalog Manager and Data Warehouse Center have Windows NT administrator privileges.

## Preparing to publish OLAP server metadata

You can use either the Data Warehouse Center user interface or a command 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.

With the DB2 OLAP Integration Server, you use a command interface to publish the metadata. You cannot synchronize updates to the metadata. You must publish the metadata again to change it in the information catalog.

### Identifying OLAP objects to publish

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

Table 12 provides the mapping between OLAP server and information catalog object types when objects are published to the information catalog. See “Appendix C. Metadata mappings” on page 133 for a detailed mapping of object types and object type properties. When you publish DB2 OLAP Integration Server metadata, a linked relationship is created between a “dimensions within a multi-dimensional database” object type in the information catalog and a table object in the DB2 OLAP Integration Server.

*Table 12. Mapping between object types*

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

Complete the following steps from a workstation where the Information Catalog Manager administrator function is installed.

1. Edit the control file in `X:\Program Files\SQLLIB\EXCHANGE\DG2OLAP.CTL` to identify the OLAP objects whose metadata you want to publish to the information catalog (where `X` is the drive where the DB2 Universal Database is installed). Identify each object separately as follows:

*servername.applicationname.databasename.outlinename*

*servername*

The name of the OLAP server.

*applicationname*

The name of the OLAP server application that contains the database that is identified by *databasename*.

## Publishing and synchronizing metadata

*databasename*

The name of the OLAP server database that contains the outline that is identified by *outlinename*.

*outlinename*

The name of the OLAP server outline whose metadata you want to publish.

For example:

```
st11w71.sample.basic.basic
st11w71.sample.internatl.internatl
st11w71.demo.basic.basic
```

2. From an MS-DOS command prompt, run the `flgnxoln` program. The DB2 OLAP Integration Server parameters are required for publishing metadata from the DB2 OLAP Integration Server to the Information Catalog Manager. If you want to publish to the DB2 OLAP Integration Server, you must specify all the DB2 OLAP Integration Server parameters. Enter the required parameters on the same line, separated by blanks:

```
flgnxoln ouuserid opassword oc_filename ic_userid ic_password ic_name generate_names
delete max_levels max_dim -ff fb_filename -hi OIS_name -hu OIS_userid -hp
OIS_password-hm OIS_model
```

*ouuserid*

The DB2 OLAP Server or Essbase supervisor user ID.

*opassword*

The password for the DB2 OLAP Server or Essbase supervisor user ID.

*filename*

The full path and file name of the control file where you identified the DB2 OLAP Server/Essbase metadata to publish.

*ic\_userid*

The Information Catalog Manager user ID for the information catalog to use for publishing metadata. The user ID can be the ID of the information catalog administrator or that of an information catalog user who has certain administrative privileges.

*ic\_password*

The password for the information catalog user ID.

*ic\_name*

The name of the information catalog to use for exchanging metadata.

*generate\_names*

Enter:

**Y** Specifies that you want to generate the object names and descriptions from the OLAP outline in the information catalog when the objects are updated. The Name and Short Description properties will be updated for objects in the information catalog.

- N** Specifies that you want to preserve the object names and descriptions, if they exist, in the information catalog when the metadata updated.

You might specify **N** if you added the object to the information catalog and specified additional business information, but you do not want the information overwritten by the updated values.

### *delete*

Values:

- Y** Specifies that if objects already exist in the information catalog, they should be deleted and then added when the objects are updated. The Information Catalog Manager deletes all objects related to the deleted object, such as associated database and table objects. The Tree view is updated so that there are no objects unrelated to other objects.
- N** Specifies that if objects already exist in the information catalog, they should be kept in the information catalog and updated.

### *max\_levels*

A value that specifies the maximum number of levels of Essbase objects that are published and displayed in the Tree view. If you do not specify this value, all objects and underlying objects that you identify are published. If you limit the number of levels, the tree structure that is displayed in Tree view will be less detailed.

This parameter is optional if you are publishing DB2 OLAP Server metadata. It is required if you are publishing DB2 OLAP Integration Server metadata.

### *max\_dimen*

A value that specifies the maximum number of dimensions and members for each level that are published and displayed in the Tree view. If you do not specify a value, for each level of the tree view, all objects that represent dimensions and all objects that represent members within those dimensions in the Essbase outline are published. If you limit the number of dimensions and members, the structure displayed in Tree view will be less detailed.

This parameter is optional if you are publishing DB2 OLAP Server metadata. It is required if you are publishing DB2 OLAP Integration Server metadata.

### *feedback file name*

The name of the feedback file. This parameter is ignored, however you must specify if you specify the DB2 OLAP Integration Server parameters.

## Publishing and synchronizing metadata

*OIS\_name*

The name of the DB2 OLAP Integration Server catalog that contains the metadata that you want to publish.

*OIS\_userid*

The DB2 OLAP Integration Server supervisor user ID.

*OIS\_password*

The password for the DB2 OLAP Integration Server supervisor user ID.

*OIS\_model*

The name of the OLAP model whose metadata you want to publish.

For example (do not enter the line breaks in this example):

```
flgnxoln olapadm olapass x:\Program Files\sqllib\logging\dg2olap.ct1 icadmin icpass  
ICMSAMP Y Y 20 20 -ff x:\Program Files\sqllib\logging\db2olap.ff oiscat -hu oisadm  
-hp oispass -hm oismod
```

You can verify that the flgnxoln program ran successfully by viewing the log file. The log file is located in the directory 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.LOG (located on the drive where you installed the DB2 Universal Database) to check the results.

Figure 17 on page 91 shows how OLAP server metadata is displayed in the Information Catalog Manager Administration Tree view.

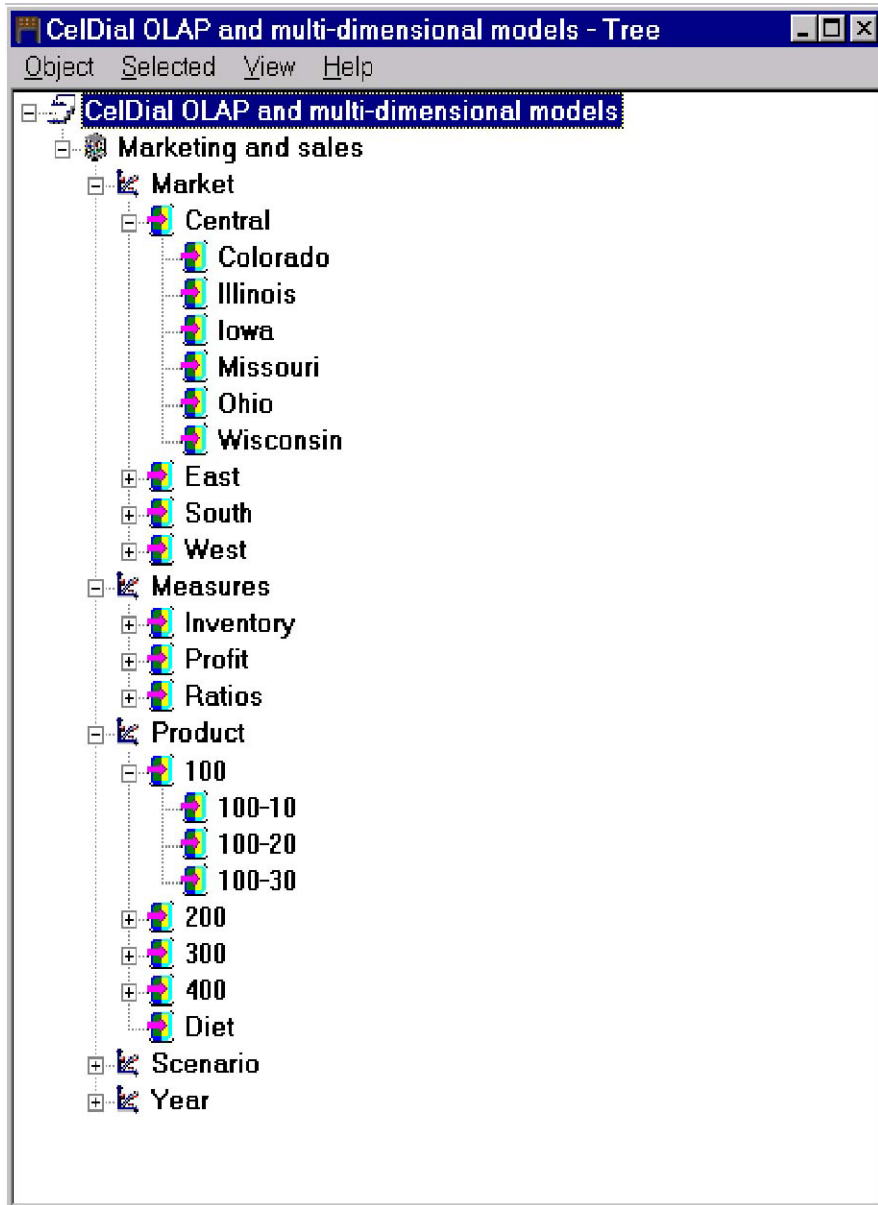


Figure 17. OLAP metadata in the Information Catalog Manager Tree view

### Setting up and scheduling 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).

## Publishing and synchronizing metadata

You can create a schedule for the publication to run on a regular basis. For step-by-step information on using the user interface, see the task "Updating published metadata" in the online help for the Publish Metadata window.

When the publication is updated, the registered metadata is checked for updates since the last time metadata was published to the information catalog.

After the publication runs, the objects that you identified in "Identifying OLAP objects to publish" on page 87 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.

### Preparing to publish Data Warehouse Center metadata

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

#### Identifying Data Warehouse Center metadata to exchange

Table 13 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. See "Appendix C. Metadata mappings" on page 133 for a detailed mapping of object types and object type properties.

*Table 13. 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, files, IMS™ database definitions
Subject	Business subject areas
Table, file or segment	IMS segments, relational tables and views
Warehouse schema	Star Schema



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.

### How Data Warehouse Center metadata is displayed in the information catalog

Metadata lineage describes the path from target data to source data. In the Data Warehouse Center, users begin working 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 Manager displays Data Warehouse Center metadata beginning with the end-result of a transformation (for example, a table or a file). You can expand the Tree view 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. Some data sources can also contain transformations. For example, Figure 18 shows a conceptual view of how metadata in the Data Warehouse Center compares to its structure when it is published to the information catalog.

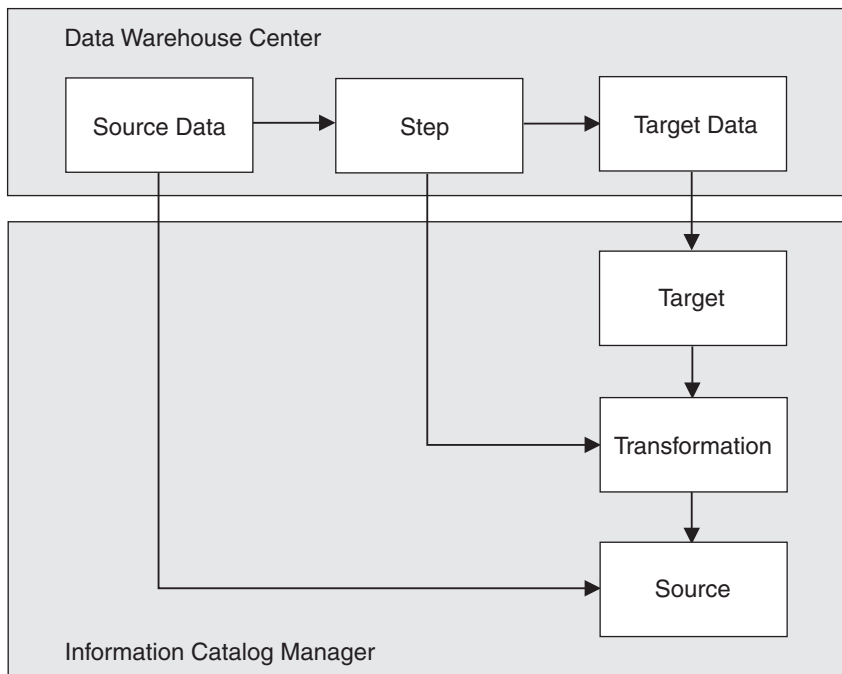


Figure 18. Path of source to target data in the Data Warehouse Center and Information Catalog Manager

## Publishing and synchronizing metadata

In the Data Warehouse Center, you use the Process Model window to map warehouse sources to warehouse targets and to define transformations (processes or steps). Figure 19 shows an example of how a process, source, and target are displayed in the Data Warehouse Center and the Information Catalog Manager Tree view.

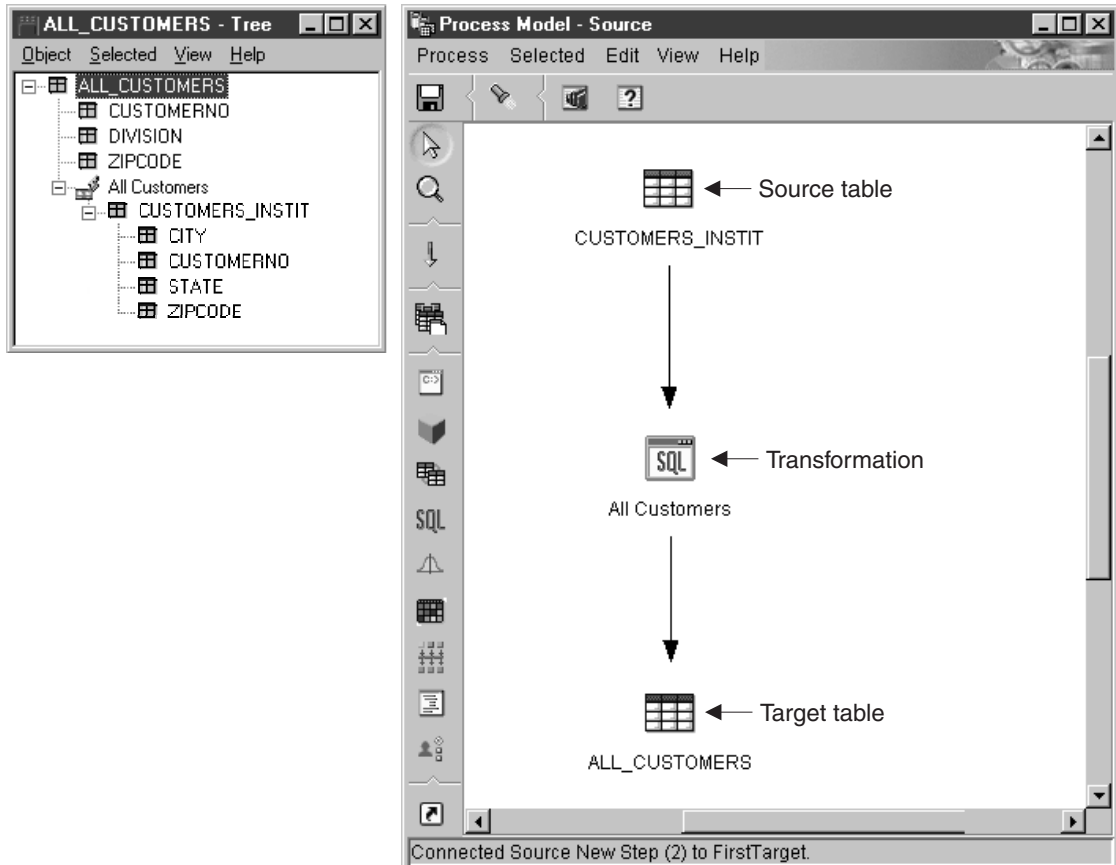


Figure 19. The Information Catalog Manager Tree view and the Data Warehouse Center Process Model window

When you publish metadata for Data Warehouse Center source and target tables, you can show transformations either at the table level or at the column level. Figure 19 shows a transformation that is mapped at the table level. The actual SQL transformation is also shown in the Description view for the transformation object.

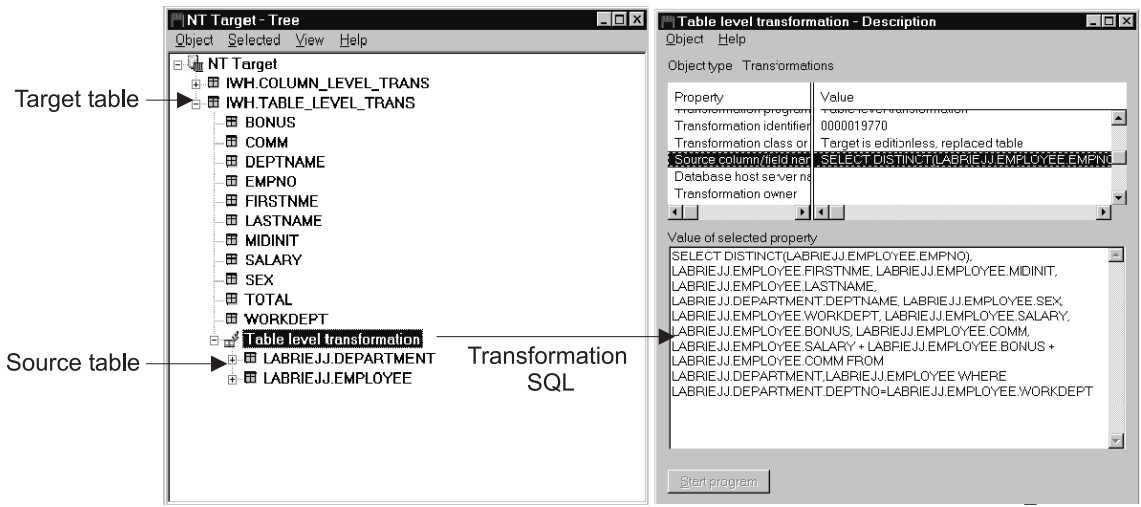


Figure 20. A transformation mapped at the table level and the Description view

## Maintaining published objects in the Data Warehouse Center

When an object is deleted in Data Warehouse Center, information about the deleted object is stored in the warehouse control database. When the metadata is updated during the synchronization process, Data Warehouse Center propagates these deletions to the information catalog before importing other changes into the information catalog. When metadata synchronization completes successfully, the Data Warehouse Center removes the entries in the warehouse control database. Because Data Warehouse Center removes the entries, Data Warehouse Center can propagate deletions to only one information catalog. If you need to make the deletions to a second information catalog, you must delete those items manually.

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. The object with the old name is not overwritten, so both objects exist in the information catalog after metadata synchronization.

## Setting up and scheduling 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. For step-by-step information, see the task "Updating published metadata," in the online help for the Publish Metadata window.

## Publishing and synchronizing metadata

When the publication is updated, the registered metadata is checked for updates since the last time metadata was published to the information catalog.

The first time that you publish Data Warehouse Center metadata to the information catalog, two publication objects are created; one publication contains control metadata, the other definition metadata. *Control metadata* is metadata that describes changes to objects in the warehouse. Examples of control metadata are the date and time that a table is updated by the processing of a step. *Definition metadata* is metadata that describes the format of objects in the warehouse, the sources of data, and the transformations that are applied to the data. Examples of definition metadata are column names, table names, and database names. The first time that you publish metadata, both control and definition metadata is registered in the information catalog. When you synchronize metadata, you can choose to update the control metadata, the definition metadata, or both types.

Table 14 shows where you can check the processing status of the types of publications after they run. The 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 NT is \SQLLIB\LOGGING, (located on the drive where you installed the DB2 Universal Database).

Table 14. Metadata synchronization status files

Type of publication	Processing status file	When there is new processing status:
Update warehouse definition metadata in the information catalog	\SQLLIB\LOGGING\ICMXCHG.OUT	The OUT file is replaced
Update warehouse control metadata in the information catalog	\SQLLIB\LOGGING\ICMDWCD.OUT	It is appended to the existing OUT file.

---

## Exchanging MDIS-conforming metadata with other products

The Information Catalog Manager includes predefined object types that can be exchanged with metadata from other Data Warehouse Center components and other MDIS-conforming products from IBM and other companies. This section describes how to use the MDIS conversion utility to convert MDIS-conforming metadata from another product into an Information Catalog Manager tag

language file. It also explains how to use the utility to convert the Information Catalog Manager metadata into an MDIS-conforming tag language file.

This section also describes how to import MDIS metadata directly into, and export MDIS metadata directly from, your information catalog.

For information about the Metadata Interchange Specification, including complete MDIS object type definitions, visit the Meta Data Coalition's Web site at <http://www.MDCinfo.com>.

### Exchanging metadata by using the MDIS conversion utilities

You can use the MDISDGC command to convert metadata that conforms to the Metadata Interchange Specification (MDIS) into an Information Catalog Manager tag language file. You can then import the tag language file into Information Catalog Manager by using the methods that are described in "Importing tag language files" on page 78.

You can export metadata from the Information Catalog Manager as described under "Exporting metadata" on page 79. You can convert the exported tag language file into an MDIS-conforming tag language file by using the DGMDISC command.

### Converting MDIS-conforming metadata into a tag language file

Use the MDISDGC command from the MS-DOS command prompt to convert MDIS-conforming metadata into an Information Catalog Manager tag language file, which you can then import into your information catalog. All MDISDGC command variables are required.

**MDISDGC** *userid password input\_tagfile output\_tagfile logfile*

*userid*

Your Windows NT user ID. User IDs for Windows NT are case-sensitive; you must enter them exactly as specified.

*password*

The password for *userid*. Passwords for Windows NT are case-sensitive; you must enter them exactly as specified.

*input\_tagfile*

Full path and file name of the MDIS-conforming tag language file that you want to convert into a tag language file. If you type only the file name, the Information Catalog Manager assumes the file exists on the drive and path that is pointed to by the DGWPATH environment variable.

*output\_tagfile*

Full path and file name for the tag language file that is created by

## Exchanging MDIS-conforming metadata

MDISDGC. If you type only the file name, the Information Catalog Manager places the file on the drive and path pointed to by the DGWPATH environment variable.

### *logfile*

Full path and file name destination for messages that are generated during conversion by the Information Catalog Manager. If you type only the file name, the Information Catalog Manager places the file on the drive and path pointed to by the DGWPATH environment variable.

### **Converting a tag language file into MDIS-conforming metadata**

Use the DGMDISC command from the MS-DOS command prompt to convert an Information Catalog Manager tag language file into MDIS-conforming metadata. You can then use this metadata with other MDIS-compliant products. All DGMDISC command variables are required.

**DGMDISC** *userid password input\_tagfile output\_tagfile logfile*

### *userid*

Your Windows NT user ID. User IDs for Windows NT are case-sensitive; you must enter them exactly as specified.

### *password*

The password for *userid*. Passwords for Windows NT are case-sensitive; you must enter them exactly as specified.

### *input\_tagfile*

Full path and file name of the tag language file that you want to convert into an MDIS-conforming tag language file. If you type only the file name, the Information Catalog Manager assumes the file exists on the drive and path that is pointed to by the DGWPATH environment variable.

### *output\_tagfile*

Full path and file name for the MDIS-conforming tag language file that is created by DGMDISC. If you type only the file name, the Information Catalog Manager places the file on the drive and path pointed to by the DGWPATH environment variable.

### *logfile*

Full path and file name destination for messages that are generated during conversion by the Information Catalog Manager. If you type only the file name, the Information Catalog Manager places the file on the drive and path pointed to by the DGWPATH environment variable.

## **Importing MDIS-conforming tag language files**

To import an MDIS tag language file directly into your information catalog, enter the Information Catalog Manager command from a an MS-DOS command prompt. Adhere to the following rules for the command syntax:

- All the parts, except where specified, are case-insensitive.

## Exchanging MDIS-conforming metadata

- Either a slash (/) or a hyphen must precede each keyword (-).
- All keywords that follow the DGUIDE command are required. All keywords that follow the /MDIS\_IMPORT keyword are required.
- Underlined choices are defaults.

```
DGUIDE /USERID userid /PASSWORD password /DGNAME dname /MDIS_IMPORT filename  
/LOGFILE filename name/ADMIN
```

Optional keywords:

```
/TRACE 0|1|2|3|4
```

For example, to import MDIS metadata into your information catalog, type the following command (do not enter a line break):

```
DGUIDE /USERID longods /PASSWORD secret /DGNAME ICMSAMP /ADMIN  
/MDIS_IMPORT c:\mdis.tag /LOGFILE c:\mdis.log
```

### **/ADMIN**

Specifies that you are logging on as an administrator. You must log on as an administrator to import metadata.

### **/DGNAME**

Your information catalog name.

If the information catalog is local, specify the database name. If the information catalog is remote, specify the alias under which it was cataloged.

Example:

```
/DGNAME ICMSAMP
```

### **/LOGFILE**

This parameter is required.

Specifies the file destination for messages that the Information Catalog Manager generates during MDIS import or MDIS export. Unless you specify a full drive, path, and file name, the Information Catalog Manager places the file in the path specified on the DGWPATH environment variable. You must specify a fixed drive.

Example:

```
/LOGFILE d:\tagfile.log
```

### **/MDIS\_IMPORT**

Imports the MDIS-conforming tag language file that you specify. Unless you specify the full drive, path, and file name, the Information Catalog Manager assumes that the file is in the path specified on the DGWPATH environment variable.

Example:

```
/MDIS_IMPORT d:\tagfile.tag
```

## Exchanging MDIS-conforming metadata

The information catalog into which you import MDIS metadata must include, but is not limited to, valid MDIS object type definitions.

### **/PASSWORD**

Your password for this user ID.

Example:

```
/PASSWORD secret
```

Passwords are case-sensitive for accessing databases on the following operating systems, you must type them exactly as specified.

- AIX
- Windows NT and Windows 2000
- Solaris Operating Environment

### **/TRACE**

The level of trace information to send to the Information Catalog Manager trace file. Each higher level includes the functions of the levels below it (for example 3 includes the functions of levels 0, 1, 2, and 3). You might need to specify a higher level if you call IBM Software Support to diagnose the Information Catalog Manager problems.

- |   |   |
|---|---|
| 0 | The default. Includes all messages and warning, error, and severe error conditions.                     |
| 1 | Includes entry and exit records of the highest level Information Catalog Manager functions.             |
| 2 | Includes extremely granular entry and exit records of the Information Catalog Manager functions.        |
| 3 | Includes input and output parameters (that exclude input or output structures).                         |
| 4 | Includes all input or output structures that are passed to and used by the Information Catalog Manager. |

### **/USERID**

Your information catalog user ID. Type the user ID required by the database where the information catalog resides. For example, the user ID might be your local, LAN, AS/400, AIX, or OS/390 TSO user ID.

Example:

```
/USERID longods
```

## Exporting MDIS-conforming tag language files

To export an MDIS tag language file directly from your information catalog, enter the DGUIDE command from an MS-DOS command prompt. Adhere to the following rules for the command syntax:

- All the parts, except where specified, are case-insensitive.



## Exchanging MDIS-conforming metadata

- Either a slash (/) or a hyphen must precede each keyword (-).
- All keywords that follow the DGUIDE command are required. All keywords that follow the /MDIS\_EXPORT keyword are required.

```
DGUIDE /USERID userid /PASSWORD password /DGNAME dname /MDIS_EXPORT filename  
/LOGFILE filename /OBJTYPE object_type /OBJECTS name
```

Optional keywords:

```
/ADMIN  
/TRACE 0|1|2|3|4
```

For example, to export MDIS metadata from your information catalog to a file, type the following command (do not enter line breaks):

```
DGUIDE /USERID longods /PASSWORD secret /DGNAME ICMSAMP /ADMIN  
/MDIS_EXPORT c:\mdis.tag /LOGFILE c:\mdis.log  
/OBJTYPE database /OBJECTS server01.payroll.valdezma
```

### **/ADMIN**

Specifies that you are logging on as an administrator. If you do not specify this optional keyword for the DGUIDE command, you log on as a user. You can export metadata as a user; however, you cannot perform all administrator tasks.

### **/DGNAME**

Your information catalog name.

If the information catalog is local, specify the database name. If the information catalog is remote, specify the alias under which it was cataloged.

Example:

```
/DGNAME ICMSAMP
```

### **/LOGFILE**

Specifies the file destination for messages that the Information Catalog Manager generates during MDIS import or MDIS export.

Unless you specify a full drive, path, and file name, the Information Catalog Manager places the file in the path specified on the DGWPATH environment variable. You must specify a fixed drive.

Example:

```
/LOGFILE d:\tagfile.log
```

### **/MDIS\_EXPORT**

Exports MDIS-conforming metadata into an MDIS-conforming tag language file with the name that you specify. Unless you specify the full drive, path, and file name, the Information Catalog Manager places the file in the path specified on the DGWPATH environment variable.

## Exchanging MDIS-conforming metadata

Example:

```
/MDIS_EXPORT d:\tagfile.tag
```

The information catalog from which you export MDIS metadata can contain metadata other than MDIS metadata, but `/MDIS_EXPORT` exports only metadata that conforms to MDIS.

### **/OBJECTS**

This parameter is required.

Specifies the objects you want to export. Depending on the object type that you specified on the `/OBJTYPE` keyword, the *name* value is from three to five property values, separated by periods.

<b>/OBJTYPE</b>	<b>/OBJECTS</b>
<b>Database</b>	<i>ServerName.DatabaseName.OwnerName</i>
<b>Dimension</b>	<i>ServerName.DatabaseName.OwnerName.DimensionName</i>
<b>Subschema</b>	<i>ServerName.DatabaseName.OwnerName.SubschemaName</i>
<b>Record</b>	<i>ServerName.DatabaseName.OwnerName.RecordName</i>
<b>Element</b>	<i>ServerName.DatabaseName.OwnerName.RecordName.ElementName</i>

In this list, the parts of the name are represented with their MDIS name. To find the equivalent information catalog names, see *Data Warehouse Center Application Integration Guide*, available from the Data Warehouse Center Web site at <http://www.software.ibm.com/data/vw/>.

### **/OBJTYPE**

This is a required parameter.

Specifies one of the following MDIS object types that you want to export:

- Database
- Dimension
- Subschema
- Record
- Element

The object type name is not case sensitive.

Example:

```
/MDIS_EXPORT d:\tagfile.tag /OBJTYPE record
```

### **/PASSWORD**

Your password for this user ID.

Example:

```
/PASSWORD secret
```

Passwords are case-sensitive for accessing databases on the following operating systems, you must type them exactly as specified.

- AIX
- Windows NT and Windows 2000
- Solaris Operating Environment

### **/TRACE**

The level of trace information to send to the Information Catalog Manager trace file. Each higher level includes the functions of the levels below it (for example 3 includes the functions of levels 0, 1, 2, and 3). You might need to specify a higher level if you call IBM Software Support to diagnose the Information Catalog Manager problems.

- 0** The default. Level includes all messages and warning, error, and severe error conditions.
- 1** Includes entry and exit records of the highest level Information Catalog Manager functions.
- 2** Includes extremely granular entry and exit records of the Information Catalog Manager functions.
- 3** Includes input and output parameters (that exclude input or output structures).
- 4** Includes all input or output structures that are passed to and used by the Information Catalog Manager.

### **/USERID**

Your information catalog user ID. Type the user ID required by the database where the information catalog resides. For example, the user ID might be your local, LAN, AS/400, AIX, or OS/390 TSO user ID.

Example:

```
/USERID 1ongods
```

## Exchanging MDIS-conforming metadata

---

## Chapter 7. Maintaining the Information Catalog Manager

The Information Catalog Manager administration tasks fall into two categories: Preventing problems and solving problems.

---

### Preventing problems

You can keep the Information Catalog Manager running smoothly by:

- Monitoring available disk space.
- Ensuring your LAN configuration provides enough resources for the Information Catalog Manager.
- Ensuring users can access the information catalog concurrently.
- Backing up your information catalog databases and supporting software regularly.
- Reminding your users to back up their personal files regularly.

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

### 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 Manager can fail, and users will not be able to access the information catalog.

Also monitor the drive on the user's workstation that contains the Windows paging file. On Windows NT, you can view or edit this file:

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 UDB for OS/2 database:*

Monitor the space available on the drive where DB2 UDB for OS/2 writes the log files it creates as it runs. If the drive does not have enough space for the log file, DB2 UDB for OS/2 can fail, closing the Information Catalog Manager. You might need to increase the number or size of these log files. To find out about or change the DB2 UDB for OS/2 log files, use the DB2 UDB for OS/2 tools.

## Preventing problems

*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.

### Ensuring that users can access the information catalog concurrently

When you use the Information Catalog Manager with a DB2 database, each person who accesses your information catalog is considered to be an agent that is using DB2. For information catalogs stored in DB2, the maximum number of idle agents must be higher than the default, which is 3. You should also increase the maximum concurrent agents.

To determine the current setting for idle and concurrent agents, enter the following command from the DB2 Command Line Processor:

```
get database manager configuration
```

To change the number of idle agents, enter the following command from the DB2 Command Line Processor: :

```
update database manager configuration use max_idleagents num
```

where *num* is the new number of idle agents.

To change the number of concurrent agents, enter the following command from the DB2 Command Line Processor: :

```
update database manager configuration use maxagents num
```

where *num* is the new number of concurrent agents.

### Backing up information catalog databases and configuration information

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.

If your information catalog database is on DB2 UDB for OS/390, the Information Catalog Manager uses two table spaces to store the data. You must back up both table spaces at the same time, because the data stored in these table spaces is interrelated.

- Backing up the Information Catalog Manager configuration information.

- 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.

### **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.

You can use either of two methods to back up the information catalog databases:

- Use the DB2 Universal Database BACKUP utility. See the administration guide for your DB2 database system for the more information.
- Export all of the data into tag language files. See “Exporting metadata” on page 79 for information.

**Additional Maintenance tips for information catalog databases:** 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.

### **Information Catalog Manager configuration information**

The Information Catalog Manager creates some configuration information on administrators’ and users’ workstations. The stored information consists of user-specific data, such as collections and saved searches for a particular information catalog. The Information Catalog Manager for Windows stores this information in the system registry.

Enter REGEDIT at an MS-DOS prompt to view the Windows system registry in the Windows Registry Editor.

To locate the working directory for the Information Catalog Manager for Windows, check the DGWPATH environment variable. (On Windows NT, you can find the environment variable on the Control Panel under Environment

## Preventing problems

Variables. On Windows 95, you can find the environment variable in your AUTOEXEC.BAT file.) Then, search for the working directory name in the Registry Editor.

One branch, or folder, in the registry has the suffix INI for each information catalog that you access. One folder contains wildcard settings, the last user ID that logged on, and the last information catalog that was used. Your administrator folder also includes default export settings.

The online help for the Registry Editor explains how to import and export selected branches or how to restore the entire registry.

Whenever there are any major changes or additions to your information catalog, you should back up your, and all your users', Information Catalog Manager configuration information.

---

## Solving problems

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

- Online information and messages.
- A utility for resetting a logged-on administrator user ID.
- A procedure for restoring your information catalog using backed up data.
- Information Catalog Manager trace file.

### Using online information and messages

The Information Catalog Manager 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 Manager messages and explanations, and extended codes for Information Catalog Manager reason codes in the *DB2 Universal Database Message Reference*. The extended codes are also described in the *Information Catalog Manager Programming Guide and Reference*.

### Resolving administrator logon problems

If the Information Catalog Manager closes unexpectedly while you are working on your information catalog, you must reset any administrator user ID that was logged on when the problem occurred. If you try to log on without resetting the administrator ID, you will get an error message telling you that another administrator is already logged on.

From an MS-DOS command prompt, enter the command:

```
X:\Program Files\SQLLIB\BIN\CLEARKA
```



where X is the drive where the DB2 Universal Database is installed.

When prompted, enter the name, user ID, and password for the information catalog, separated by blanks. For example:

```
ICMSAMP longods secret
```

Then open your information catalog again.

If your information catalog is stored in a DB2 UDB for OS/390 Version 4.1 database and the Information Catalog Manager is forced to close because communication with the database is lost, you might need to increase the value of the DB2 UDB for OS/390 system parameter IDTHTOIN.

## Recovering Information Catalog Manager 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 Manager, and data.

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 Manager, as necessary.
2. Restore the information catalog databases by using your backup files.
  - If your backup file is a database image copy:
    - a. See the documentation for your database management system for information on restoring your information catalog information catalog database.
    - b. Bind the restored databases to the Information Catalog Manager by issuing the SQLBIND command from a command prompt. Table 15 indicates the bind files that you should use for the Information Catalog Manager component that you are restoring.

*Table 15. Information Catalog Manager Version 5.2 bind files*

<b>If you are using:</b>	<b>Use these bind files:</b>
Information Catalog Manager for Windows Administrator	FLGNJKA1.BND FLGNJKW1.BND FLGNJKX1.BND
Information Catalog Manager for Windows User	FLGNJKW1.BND FLGNJKX1.BND

Specify the following parameter values for the bind files:

## Solving problems

<b>Datetime format</b>	ISO
<b>Grant execute privilege</b>	PUBLIC
<b>Row blocking</b>	ALL (does not apply to FLGNJKA0.BND or FLGNJKA1.BND)

- If your backup file is an Information Catalog Manager tag language file that is produced by the Information Catalog Manager export process:
  - a. Define your information catalog again using the Initialize Information Catalog window.
  - b. Import the tag language file to restore your data.

### Using trace files for problem diagnosis

The Information Catalog Manager automatically creates a file, called a *trace file*, that can help you diagnose problems.

The trace file is created each time you log on to the Information Catalog Manager. It includes the time and date you started the Information Catalog Manager, product version, service level, and information about what is happening in the Information Catalog Manager while it is running.

The Information Catalog Manager gives the trace file the name of the information catalog you are using plus an extension of TRC. The Information Catalog Manager for Windows places the trace file on the drive and path that are specified by the DGWPATH environment variable. (On Windows NT, you can find this variable on the Control Panel under Environment Variables. On Windows 95, you can find this variable in your AUTOEXEC.BAT file.) For example, if you are using the DGIVP information catalog, the trace file is named DGIVP.TRC on the drive and path that are specified by the DGWPATH environment variable.

The Information Catalog Manager overwrites the trace file each time you log on.

#### **Important**

If the Information Catalog Manager unexpectedly closes, immediately go to a command prompt and rename the trace file before you call IBM Software Support. Otherwise, when you restart the Information Catalog Manager you will overwrite the trace file and clear the records that you need to explain your problem.

You can trace Information Catalog Manager activities at five levels. The default level is 0. At this level, the Information Catalog Manager records only error messages that you see on the screen. If the Information Catalog Manager

encounters a severe error, it automatically increases the trace level to record the error number and any associated extended codes. You can look up the error numbers and some extended codes in the online book *DB2 Universal Database Message Reference*. An *extended code* is an additional explanatory code that provides more information about an error in a specific situation. However, the trace file can contain extended codes for the Information Catalog Manager or Structured Query Language (SQL) statements. You might have to look in the associated message reference for additional information.

You might need higher trace levels to help IBM Software Support diagnose a problem. To do so, you must start the Information Catalog Manager from a command prompt and use the /TRACE option. See “Importing a tag language file from the command line” on page 185 for more information.

Figure 21 shows an example of a trace file.

***What to do with trace files:***

```
*****
* Information Catalog Manager Trace File
*
* Date and Time : Fri Jan 21 10:15:38 2000
* Product Version : V7.1
*
*****
Starting Trace... >> PID/TID = 319 / 320"
Starting Trace Level: 0
FLGGETReg >> PID/TID = 319 / 320"
New Reason Code: 1008
-----
```

*Figure 21. Trace file showing Information Catalog Manager reason and extended codes*

1. Look up error messages in the *DB2 Universal Database Message Reference*.
2. Follow the actions that are recommended in the book.
3. If you see an extended code, look it up in the following documentation:  
For SQL codes and Information Catalog Manager extended codes see the *DB2 Universal Database Message Reference*. The extended codes are also described in the *Information Catalog Manager Programming Guide and Reference*.

In Figure 21, the New Reason Code: 1008 specifies that the FLGGETReg API could not retrieve an icon file.

## Solving problems

---

## Appendix A. Information Catalog Manager extract programs

The Information Catalog Manager gives you a set of extract programs that can extract descriptive data from several database management systems and desktop applications. This appendix briefly describes how to begin using these extractors.

---

### Extract programs supplied with the Information Catalog Manager

You can extract descriptive data from the following sources.

- BACHMAN Database Administrator
- Quattro Pro
- CorelDRAW!
- Harvard Graphics
- IBM DB2 Universal Database family
- IBM DataAtlas (IMS and relational definitions)
- IBM DataJoiner
- Lotus 1–2–3
- Lotus Approach
- Lotus Freelance Graphics
- Microsoft Excel
- Microsoft Word
- ODBC extractor
- Oracle
- Texas Instruments Information Engineering Facility (TI/IEF)
- WordPerfect

To use these extract programs, you must be familiar with the specific operating environments in which they run.

---

### Planning to run extract programs

To effectively use the extract programs to populate an information catalog, complete the following preparatory steps:

1. Identify descriptive data for extraction.

The Information Catalog Manager extract programs for relational databases extract descriptive data for Table and Column types of objects.

See “Appendix B. Predefined Information Catalog Manager object types” on page 115 for a description of these object types. 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

## Planning to run extract programs

properties that the catalogs contain. You can fill in the missing information by editing the tag language file. For complete information on writing tag language files, see “Appendix D. Tag language” on page 147. The extract programs for desktop applications create their own object types specifically for these applications.

### **Important:**

This step is especially important if you plan to extract data for storage in a DB2 UDB for AS/400 information catalog. After you create object types in a DB2 UDB for AS/400 information catalog, you cannot update them to add properties. You must create exactly the object types you want to fit the data you plan to extract. You might want to create separate object types that are specifically tailored to the extracted data. For example, you might already have an object type called Columns, so you can create an object type Columns2 for extracted data.

2. Select the appropriate extract program.

When you know what descriptive data you want to extract, choose the appropriate extract program and decide if you need to change it.

The DGWEXT program for the IBM DB2 family of relational databases needs no modification. IBM supports and services the DGWEXT program. The other extract programs that come with the Information Catalog Manager are only sample programs. You might need to change them to work with your own environment and databases.

3. Read the documentation about the extract program you want to use.

Each extractor exists in its own subdirectory located in the \SQLLIB\SAMPLES path on the drive where the DB2 Universal Database is installed. Information about, and steps for using, each of the extractors is supplied in a README file in that extractor’s subdirectory.

---

## Appendix B. Predefined Information Catalog Manager object types

The Information Catalog Manager includes predefined object types that you can exchange with metadata from other Data Warehouse Center components and other MDIS-conforming products from IBM and other companies. This appendix describes all of the predefined Information Catalog Manager object types and describes how the object type properties map to MDIS object types. For information about the Metadata Interchange Specification, including complete MDIS object type definitions, visit the Meta Data Coalition's Web site at <http://www.MDCinfo.com>

The Information Catalog Manager provides both the predefined object types and sample objects of each type within the sample information catalog. The sample information catalog includes at least one object type for each of the seven Information Catalog Manager categories. This appendix describes how to create the sample information catalog. For details of Information Catalog Manager object type capabilities, see "Creating your own object types" on page 28.

---

### Accessing the Information Catalog Manager sample data

This section describes how you can create a sample information catalog that contains both the predefined object types and objects of those types. It also describes how to import only the predefined object types.

#### Creating a sample information catalog

You can help your users learn to use the Information Catalog Manager by setting up a sample DB2 UDB for Windows NT information catalog with which they can experiment. The sample information catalog includes all the predefined Information Catalog Manager object types, and objects of those types. The objects describe information that is typically required in a business environment. The *Information Catalog Manager User's Guide* uses the CelDial business scenario to introduce the Information Catalog Manager by using examples from this sample information catalog.

You create the DB2 database on the platform you want and then use the Create Information Catalog utility to create the sample information catalog. The sample information catalog utility populates your newly created information catalog with the sample data.

To create the sample information catalog:

## Creating a sample information catalog

1. To define a new information catalog, follow the steps for your database system in “Chapter 1. Setting up an information catalog” on page 1 or in the online help. When you create it, name your sample information catalog ICMSAMP. If an information catalog named ICMSAMP already exists, name the new information catalog something else, but be sure to notify your users of the new name.
2. At an MS-DOS command prompt, enter:

```
X:\Program Files\SQLLIB\SAMPLES\SAMPDATA\DGWDEMO
```

where X is the drive where you installed the DB2 Universal Database.

To get information about additional parameters that you can specify with this command, type DGWDEMO ? at the MS-DOS command prompt.

3. Press Enter. You can change any, or all, of the following four default options.
  - 1 The fully qualified path for the Web server on which you installed the Information Catalog Manager for the Web (for example, dgntserv2.stl.ibm.com) if applicable.
  - 2 The administrator user ID for the information catalog.
  - 3 Passwords are case-sensitive for accessing databases on the following operating systems, you must type them exactly as specified.
    - AIX
    - Windows NT and Windows 2000
    - Solaris Operating Environment
  - 4 The name of the information catalog that you created for the sample data. This is ICMSAMP, unless you named it differently because you have an existing catalog named ICMSAMP.

After you press Enter the last time, the utility proceeds to load the sample data. The utility notifies you when it finishes loading the sample data. The sample information catalog is now ready to use.

**Note for Information Catalog Manager for the Web users:** The sample data in the \SQLLIB\SAMPLES\SAMPDATA directory must be copied to the Information Catalog Manager for the Web server. You must copy it to a directory named ICMSAMP under the root directory. For example:

- For Lotus Domino Go Webserver, create the directory WEBSERVE\HTML\ICMSAMPLE.
- For Microsoft Internet Information Server (IIS), create the directory INETPUB\WWWROOT\ICMSAMPLE.
- For Apache Web Server on AIX, create the directory /usr/local/apache/htdocs/icmsample.



### Initializing your information catalog with the predefined object types

If you were unable to import the common object types when you created your information catalog, and you do not want to import the entire sample tag language file, you can import the object types using the following command at an MS-DOS command prompt:

```
X:Program Files\SQLLIB\SAMPLES\SAMPDATA\DGWDEMO /T userid password dname
```

*X* The drive where you installed the DB2 Universal Database.

*userid*

The administrator user ID for the information catalog.

*password*

The password for *userid*.

Passwords are case-sensitive for accessing databases on the following operating systems, you must type them exactly as specified.

- AIX
- Windows NT and Windows 2000
- Solaris Operating Environment

*dname*

The name of the information catalog that you want to initialize with the predefined object types.

---

### Predefined object type models

Information Catalog Manager predefined object types follow the six data models shown in Figures 22 through 27.

Figure 22 shows the object types that participate in the relational model.

## Predefined object type models

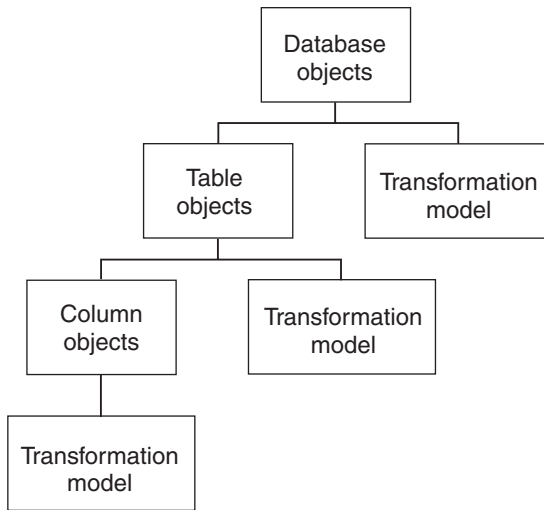


Figure 22. Relational model and the predefined object types

Figure 23 shows the object types that participate in the hierarchical models.

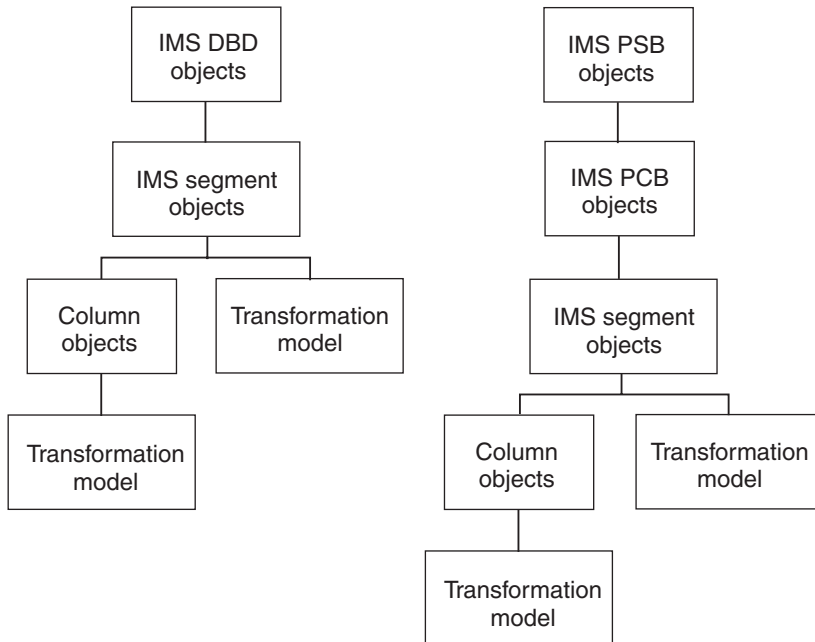


Figure 23. Hierarchical models and the predefined object types

Figure 24 shows the object types that participate in the file models.

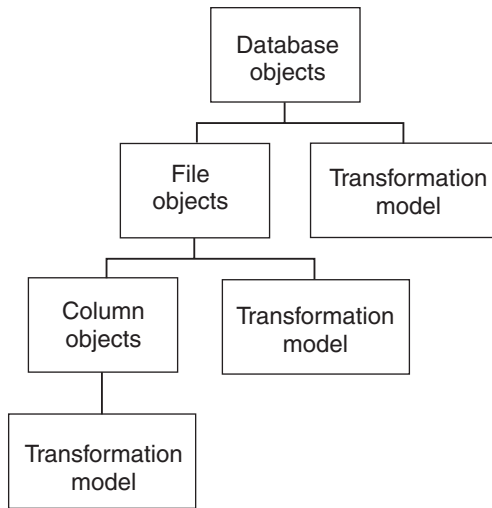


Figure 24. File models and the predefined object types

Figure 25 shows the object types that participate in the multi-dimensional (OLAP) model.

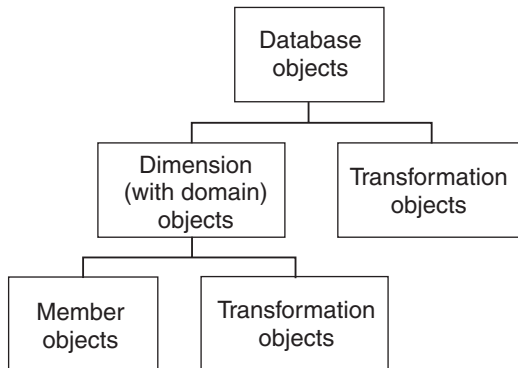


Figure 25. Multi-dimensional model and the predefined object types

Figure 26 shows the object types that participate in the transformation models.

## Predefined object type models

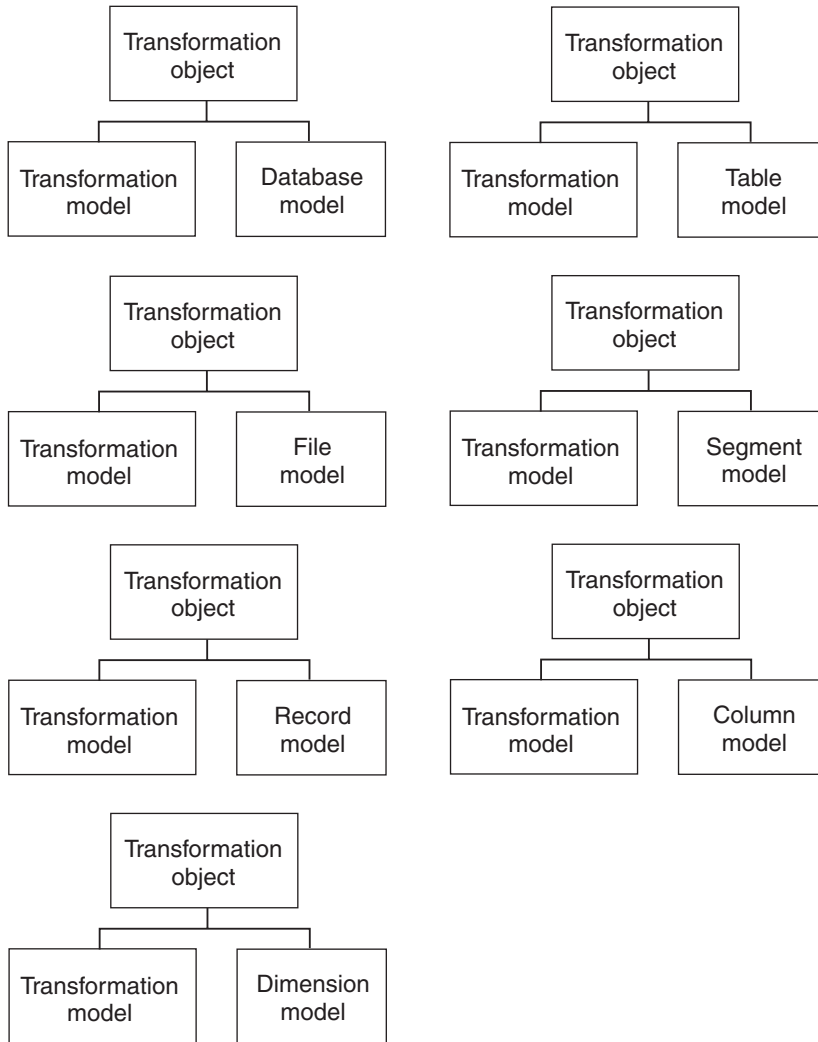


Figure 26. Transformation model and the predefined object types

Figure 27 shows the object types that participate in the subject area model.

## Descriptions of predefined Information Catalog Manager object types

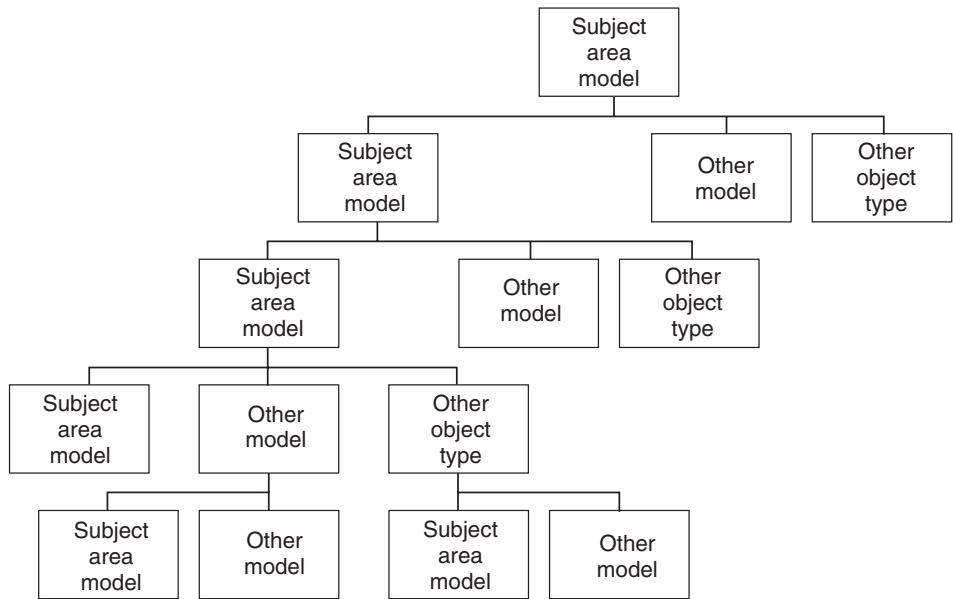


Figure 27. Subject area model and the predefined object types

---

### Predefined object type descriptions

Sample Information Catalog Manager object types are organized by category. They are described beginning on page 122.

For comprehensive tables that describe each object type and its property specifications see *Data Warehouse Center Application Integration Guide*, available from the Data Warehouse Center Web site at <http://www.software.ibm.com/data/vw/>.

The tag language files for each object type described in the following sections are located in \SQLLIB\DGWIN\TYPES on the drive where the DB2 Universal Database is installed.

#### Grouping category

The Grouping category contains the following object types:

- Application data
- Business subject areas
- Columns or fields
- Databases
- DWC (Data Warehouse Center) process
- Dimensions within a multidimensional database
- Elements

## Descriptions of predefined Information Catalog Manager object types

- Files
- IMS database definitions (DBD)
- IMS program control blocks (PCB)
- IMS program specification blocks (PSB)
- IMS segments
- Members within a multidimensional database
- Multidimensional databases
- Records
- Relational tables and views
- Star schemas
- Subschemas
- Transformations

### **“Application data” object type**

The Information Catalog Manager uses the “Application data” object type internally for some MDIS metadata exchanges. Objects of this object type might appear in your information catalog. However, you will not use this object type to create objects.

The tag language for defining the “Application data” object type is in the file FLGNYAPL.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Business subject areas” object type**

The “Business subject areas” object type represents logical groupings of objects.

The tag language for defining the “Business subject areas” object type is in the file FLGNYINF.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Columns or fields” object type**

The “Columns or fields” object type represents columns within a relational table, fields within a file, or fields within an IMS segment.

The tag language for defining the “Columns or fields” object type is in the file FLGNYCOL.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Databases” object type**

The “Databases” object type represents relational databases.

The tag language for defining the “Databases” object type is in the file FLGNYDAT.TYP in the \SQLLIB\DGWIN\TYPES directory.

## Descriptions of predefined Information Catalog Manager object types

### **“Dimensions within a multidimensional database” object type**

The “Dimensions within a multidimensional database” object type represents dimensions within a multi-dimensional database. A dimension consists of members.

The tag language for defining the “Dimensions within a multidimensional database” object type is in the file FLGNYDIM.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **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.

The tag language for defining the a “DWC” object type is in the file FLGNYZD02.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Elements” object type**

The “Elements” object type represents MDIS Element objects that do not map directly to the “Columns or fields” object type.

The tag language for defining the “Elements” object type is in the file FLGNYELE.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Files” object type**

The “Files” object type represents a file within a file system.

The tag language for defining the “Files” object type is in the file FLGNYFIL.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“IMS database definitions (DBD)” object type**

The “IMS database definitions (DBD)” object type represents IMS database definitions.

The tag language for defining the “IMS database definitions (DBD)” object type is in the file FLGNYDBD.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“IMS program control blocks (PCB)” object type**

The “IMS program control blocks (PCB)” object type represents IMS program control blocks.

The tag language for defining the “IMS program control blocks (PCB)” object type is in the file FLGNYPCB.TYP in the \SQLLIB\DGWIN\TYPES directory.

## Descriptions of predefined Information Catalog Manager object types

### **“IMS program specification blocks (PSB)” object type**

The “IMS program specification blocks (PSB)” object type represents IMS program specification blocks.

The tag language for defining the “IMS program specification blocks (PSB)” object type is in the file FLGNYPBSB.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“IMS segments” object type**

The “IMS segments” object type represents IMS segments.

The tag language for defining the “IMS segments” object type is in the file FLGNYSEG.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Members within a multidimensional database” object type**

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.

The tag language for defining the “Members within a multidimensional database” object type is in the file FLGNYMEM.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Multidimensional databases” object type**

The “Multidimensional databases” object type represents multidimensional databases.

The tag language for defining the “Multi-dimensional databases” object type is in the file FLGNYOLA.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Records” object type**

The “Records” object type represents MDIS Record objects that do not map directly to the “Files” or “Relational tables or views” object types. Records consist of Elements.

The tag language for defining the “Records” object type is in the file FLGNYREC.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Relational tables and views” object type**

The “Relational tables and views” object type represents tables or views of relational databases.

The tag language for defining the “Relational tables and views” object type is in the file FLGNYTAB.TYP in the \SQLLIB\DGWIN\TYPES directory.



## Descriptions of predefined Information Catalog Manager object types

### **Star Schemas**

This object type represents OLAP server relational data. A star schema contains a fact table and one or more dimension tables.

The tag language for defining the “star schemas” object type is in the file FLGNYZ01.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Subschemas” object type**

The “Subschemas” object type represents logical groupings of records within a database.

The tag language for defining the “Subschemas” object type is in the file FLGNYSUB.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Transformations” object type**

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.

The tag language for defining the “Transformations” object type is in the file FLGNYFLT.TYP in the \SQLLIB\DGWIN\TYPES directory.

## **Elemental category**

The Elemental category contains the following object types:

- Audio clips
- Charts
- Documents
- Images or graphics
- Internet documents
- Lotus Approach<sup>®</sup> queries
- Presentations
- Spreadsheets
- Text-based reports
- Video clips

### **“Audio clips” object type**

The “Audio clips” object type represents files that contain audio information. These objects might represent electronic (AUD files) or physical (for example, CDs, tapes) audio information.

The tag language for defining the “Audio clips” object type is in the file FLGNYAUD.TYP in the \SQLLIB\DGWIN\TYPES directory.

## Descriptions of predefined Information Catalog Manager object types

### **“Charts” object type**

The “Charts” object type represents either hardcopy or electronic charts.

The tag language for defining the “Charts” object type is in the file FLGNYCHA.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Documents” object type**

The “Documents” object type represents books, manuals, and technical papers. These publications might be printed or electronic, found locally or within a library.

The tag language for defining the “Documents” object type is in the file FLGNYDOC.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Images or graphics” object type**

The “Images or graphics” object type represents graphic images, such as bitmaps.

The tag language for defining the “Images or graphics” object type is in the file FLGNYIMA.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Internet documents” object type**

The “Internet documents” object type represents Web sites and other documents on the Internet that might be of interest.

The tag language for defining the “Internet documents” object type is in the file FLGNYINT.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Lotus Approach queries” object type**

The “Lotus Approach queries” object type represents available Lotus Approach queries for use with your organization’s data.

The tag language for defining the “Lotus Approach queries” object type is in the file FLGNYAPR.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Presentations” object type**

The “Presentations” object type represents various hardcopy or electronic presentations. These presentations might include product, customer, quality, and status presentations.

The tag language for defining the “Presentations” object type is in the file FLGNYPRE.TYP in the \SQLLIB\DGWIN\TYPES directory.

### **“Spreadsheets” object type**

The “Spreadsheets” object type represents desktop spreadsheets (for example, Lotus 1-2-3 or Microsoft Excel spreadsheets).

## Descriptions of predefined Information Catalog Manager object types

The tag language for defining the “Spreadsheets” object type is in the file FLGNYSSH.TYP in the \SQLLIB\DGWIN\TYPES directory.

### “Text-based reports” object type

The “Text-based reports” object type represents either hardcopy or electronic reports.

The tag language for defining the “Text-based reports” object type is in the file FLGNYREP.TYP in the \SQLLIB\DGWIN\TYPES directory.

### “Video clips” object type

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.

The tag language for defining the “Video clips” object type is in the file FLGNYVID.TYP in the \SQLLIB\DGWIN\TYPES directory.

## Contact category

The Contact category contains the “People to contact” object type.

### “People to contact” object type

The “People to contact” object type identifies a person or group that is responsible for single or multiple objects within the information catalog.

The tag language for defining the “People to contact” object type is in the file FLGNYCON.TYP in the \SQLLIB\DGWIN\TYPES directory.

## Dictionary category

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.

The tag language for defining the “Glossary entries” object type is in the file FLGNYGLO.TYP in the \SQLLIB\DGWIN\TYPES directory.

## Support category

The Support category contains the following object types:

- Information catalog news
- Online news services
- Online publications

### “Information Catalog Manager news” object type

The “Information Catalog Manager news” object type conveys information to end users about changes to the information catalog.

## Descriptions of predefined Information Catalog Manager object types

The tag language for defining the “Information Catalog Manager news” object type is in the file FLGNYDGN.TYP in the \SQLLIB\DGWIN\TYPES directory.

### “Online news services” object type

The “Online news services” object type represents news and information services that can be accessed online.

The tag language for defining the “Online news services” object type is in the file FLGNYOLN.TYP in the \SQLLIB\DGWIN\TYPES directory.

### “Online publications” object type

The “Online publications” object type represents publications and other documents that can be accessed through online services.

The tag language for defining the “Online publications” object type is in the file FLGNYOLP.TYP in the \SQLLIB\DGWIN\TYPES directory.

## Program category

The Program category can contain only the Programs object type. The Programs object type is created when an information catalog is created and is used to define an application capable of processing a particular object type.

In the sample information catalog, ICMSAMP, the Programs object type is named “Programs that can be invoked from information catalog objects”.

## Attachment category

The Attachment category can contain only the “Comments” object type. The “Comments” object type is created when an information catalog is created.

The Comments object type holds comments about other objects in the information catalog.

## Predefined program objects

Program object types shown in Table 16 are provided in the sample information catalog. The table also shows the property name that you use to associate with the Information Catalog Manager program object when launching a program.

Table 16. Generic predefined program objects in the sample information catalog

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

## Descriptions of predefined Information Catalog Manager object types

Table 16. Generic predefined program objects in the sample information catalog (continued)

Type of information	Program name	Object type	Property name
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 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 17 lists specific IBM Business Partners who have applications that are integrated with the Information Catalog Manager. The information in this table similar to that in Table 16 on page 128.

Table 17. Predefined program objects in sample information catalog — IBM Business Partners

Type of information	Program name	Object type	Property name
Lotus	Approach	Lotus Approach	Approach object filename
	Freelance Graphics	Presentations	Presentation object filename
Hyperion	Lotus 1-2-3 with Essbase Spreadsheet add-in	Spreadsheets	Spreadsheet filename
	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

## Descriptions of predefined Information Catalog Manager object types

Table 17. Predefined program objects in sample information catalog — IBM Business Partners (continued)

Type of information	Program name	Object type	Property name
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 Impromptu 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

## Descriptions of predefined Information Catalog Manager object types

Table 17. Predefined program objects in sample information catalog — IBM Business Partners (continued)

Type of information	Program name	Object type	Property name
	Microsoft PowerPoint Viewer within Netscape	Text based reports	URL to access data
	Microsoft PowerPoint Viewer within Microsoft Internet Explorer	Text based reports	URL to access data

## Descriptions of predefined Information Catalog Manager object types



---

## Appendix C. Metadata mappings

This appendix lists object types and object type properties for the following metadata:

- Information Catalog Manager metadata to Data Warehouse Center metadata, described in “Metadata Mappings between the Information Catalog Manager and the Data Warehouse Center” below.
- Information Catalog Manager metadata to OLAP server metadata, described in “Metadata mappings between the Information Catalog Manager and OLAP server” on page 143.

---

### Metadata Mappings between the Information Catalog Manager and the Data Warehouse Center

The following tables show the metadata mappings between the Information Catalog Manager and the Data Warehouse Center for each object type. The Information Catalog Manager 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 (such as processing time stamps for steps) is taken from the Work in Progress window.

*Table 18. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for database objects*

Information Catalog Manager 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

*Table 18. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for database objects (continued)*

Database location	N/A
Database host server name	System name
System code page	This metadata is internal to the Data Warehouse Center. <sup>1</sup>
Database server type and database extended type.	The mapping is derived from the warehouse source or warehouse target type.  For example, if your warehouse target is a DB2 Universal Database for Windows NT database, the database server type is DB2 Family. The database extended type is DB2 NT.
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. <sup>1</sup>
Database status	N/A
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 Universal Database for Windows NT database, the database extended type is DB2 NT.
Timestamp source definition created	N/A
For further information	Administrator
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

*Table 19. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for IMS DBD (database description definition) objects*

Information Catalog Manager metadata	Data Warehouse Center metadata
Name	Warehouse source name
Short description	Description
Long description	Notes

Table 19. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for IMS DBD (database description definition) objects (continued)

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	This property is set to HIERARCHICAL.  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. <sup>1</sup>
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

Table 20. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for relational table or view objects

Information Catalog Manager metadata	Data Warehouse Center metadata
Name	Table name

Table 20. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for relational table or view objects (continued)

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
Table data last refreshed	<p>Last completed time stamp for a step that ran and used the table as a target table.</p> <p>This information is displayed in the Work in Progress window.</p>
Transformation program type	<p>The value for this property is Data Warehouse Center.</p> <p>There is no specific metadata in the Data Warehouse Center for this property.</p>
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	<p>Last update time stamp for a table definition.</p> <p>This metadata is internal to the Data Warehouse Center.<sup>1</sup></p>
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 20. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for relational table or view objects (continued)

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 Universal Database for Windows NT database, the database table type is DB2 NT.
Definition represents a view	N/A
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.
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

Table 21. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for column or field objects

Information Catalog Manager 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

Table 21. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for column or field objects (continued)

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. <sup>1</sup>
Is column or field part of a key	N/A
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. <sup>1</sup>
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.

Table 21. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for column or field objects (continued)

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
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

Table 22. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for file objects

Information Catalog Manager metadata	Data Warehouse Center metadata
Name	The value for this property is derived from the file name.
Short description	Description
Long description	N/A
URL to access data	N/A
Actions	N/A
Information last refreshed	Last completed time stamp for a step that ran and used the file as a target file.
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

Table 22. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for file objects (continued)

Source definition last changed	Last update time stamp for a file definition.  This metadata is internal to the Data Warehouse Center. <sup>1</sup>
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.
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

Table 23. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for IMS segment objects

Information Catalog Manager 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
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



Table 23. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for IMS segment objects (continued)

Timestamp source definition last changed	Last update time stamp for a segment definition.  This metadata is internal to the Data Warehouse Center. <sup>1</sup>
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.
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

Table 24. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for transformation objects

Information Catalog Manager 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. <sup>1</sup>
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. <sup>1</sup>

Table 24. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for transformation objects (continued)

Database host server name	Target database system name
Transformation owner	N/A
Source sequence	N/A
Transformation ordinality	N/A
Transformation bidirectionality	N/A
Timestamp source definition created	N/A
For further information	Administrator
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

Table 25. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for business subject area objects

Information Catalog Manager 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 26. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for star schema objects

Information Catalog Manager 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

Table 26. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for star schema objects (continued)

Timestamp source definition last changed	Last update time stamp for warehouse schema definition.  This metadata is internal to the Data Warehouse Center. <sup>1</sup>
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

Table 27. Metadata mappings between the Information Catalog Manager and the Data Warehouse Center for Data Warehouse Center process objects

Information Catalog Manager 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
Timestamp source definition last changed	Last update time stamp for the process definition.  This metadata is internal to the Data Warehouse Center. <sup>1</sup>
<b>Note:</b>	
1. If a property is specified as internal to the Data Warehouse Center, it is not displayed.	

---

## Metadata mappings between the Information Catalog Manager and OLAP server

Table 28 on page 144 shows the mapping of OLAP server metadata to the Information Catalog Manager common object types. OLAP server metadata refers to metadata for DB2 OLAP Server, DB2 OLAP Integration Server, or Hyperion Essbase Server.

When you publish DB2 OLAP Integration Server metadata to the information catalog, a linked relationship is created between a "dimensions within a multi-dimensional database" object type and a table object.

The left column of the table shows the name of the Essbase API structure. The right column shows the Information Catalog Manager object and object type

properties.

Table 28. Mapping of OLAP server metadata to the Information Catalog Manager common object types

OLAP server metadata	Information Catalog Manager metadata
<i>Outline</i>	<i>Multidimensional databases</i>
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.
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.
<i>Dimensions in an outline</i>	<i>Dimension within a multidimensional database</i>
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
<i>Members in a dimension</i>	<i>Members within a multidimensional database</i>
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

*Table 28. Mapping of OLAP server metadata to the Information Catalog Manager common object types (continued)*

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).



---

## Appendix D. Tag language

The Information Catalog Manager tag language allows you to format your descriptive data so that you can import it into your information catalog. The tag language tells the Information Catalog Manager what to do with the descriptive data that it imports. The Information Catalog Manager also exports descriptive data into tag language files so that you can back up your information catalog or transfer data from one information catalog to another.

By formatting descriptive data with the tag language, you can move descriptive data from one information catalog to another and define Information Catalog Manager object types and objects. You can also write and use extract programs to extract descriptive data from other sources, such as a relational database catalog, that you can import to your information catalog. Table 29 shows the tags in the tag language and the actions that these tags perform.

*Table 29. Information Catalog Manager tags*

<b>Task</b>	<b>Tag names</b>	<b>For details</b>
Record diskette sequence	DISKCNTL	see page 162
Identify action to be taken on input data	ACTION.OBJINST	see page 150
	ACTION.OBJTYPE	see page 155
	ACTION.RELATION	see page 159
Describe data to the information catalog	OBJECT	see page 169
	PROPERTY	see page 174
	INSTANCE	see page 163
	RELTYPE	see page 178
Identify when changes are committed and where check point occurs	COMMIT	see page 161
Identify user comments	COMMENT	see page 160
Format data	NL	see page 168
	TAB	see page 179

---

### 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 Table 29 on page 147.

- 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 OBJTYPID, INSTIDNT, UPDATIME, or UPDATEBY as property short names (*short\_names*) with the PROPERTY or INSTANCE tags.
- These property names are reserved by Information Catalog Manager:  
OBJTYPID  
INSTIDNT  
NAME  
UPDATIME  
UPDATEBY

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

:PROPERTY.SHRTNAME(NAME) UUISEQ(1)

---

## How the Information Catalog Manager reads tag language files

When you code a tag language file, consider how the Information Catalog Manager reads tag language files:

- The Information Catalog Manager reads the entire tag language file as a continuous data stream.
- The Information Catalog Manager 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.
- The Information Catalog Manager considers a tag complete when it encounters the next tag in the tag language file.
- Tags and keywords are not translated into national languages.
- Only the values for the keywords in Table 30 on page 149 are enabled for double-byte character set (DBCS) support.



Table 30. Keyword values enabled for DBCS

Tag name	Keywords	Variable value
OBJECT	EXTNAME	<i>ext_name</i>
	ICWFILE	<i>Windows_ICON_file_name</i>
PROPERTY	EXTNAME	<i>ext_name</i>
COMMIT	CHKPID	<i>checkpt_id</i>
INSTANCE	<i>UUI_short_name</i>	<i>UUI_property_value</i>
	<i>or</i>	<i>or</i>
	<i>short_name</i>	<i>property_value</i>

All user-defined property values can use DBCS characters.

- The Information Catalog Manager accepts DBCS blanks only in the keyword values that are shown in Table 31. If DBCS blanks appear anywhere else in the tag language file, errors can occur.

Table 31. Keyword values enabled for DBCS blank characters

Tag name	Keywords
ACTION	OBJTYPE
	OBJINST
	RELATION
OBJECT	All keywords
PROPERTY	All keywords
RELTYPE	All keywords
COMMIT	CHKPID
INSTANCE	<i>UUI_short_name</i>
	<i>or</i>
	<i>short_name</i>

## Valid data types for Information Catalog Manager descriptive data

Table 32 shows the valid data types for Information Catalog Manager descriptive data.

Table 32. Valid data types for Information Catalog Manager descriptive data

Data type	Description
CHAR	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	26-character timestamp in the following format: <i>yyyy-mm-dd-hh.mm.ss.nnnnnn</i>

Table 32. Valid data types for Information Catalog Manager descriptive data (continued)

Data type	Description
LONG VARCHAR	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 UII property.
VARCHAR	Varying-length character string between 1 and 4 000 bytes long.

The Information Catalog Manager automatically removes trailing blanks from variable values and adjusts their length accordingly before validating and accepting the request.

If a required value is not specified or contains all blanks, the Information Catalog Manager inserts the values that are shown in Table 33.

Table 33. Information Catalog Manager-supplied values

Data type	Supplied value
CHAR	A not-applicable symbol as the first character and padded with trailing blanks to fill the defined length.
TIMESTAMP	9999-12-31-24.00.00.000000
LONG VARCHAR	A not-applicable symbol
VARCHAR	A not-applicable symbol

---

## How to read the tag language syntax diagrams

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).

---

## 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 Manager 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()
:INSTANCE.short_name()
:INSTANCE.short_name()

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

*Figure 28. 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.

## ACTION.OBJINST

- 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()
:INSTANCE.SOURCEKEY(UII_short_name()...)

:OBJECT.TYPE()
:INSTANCE.SOURCEKEY(UII_short_name()...)
```

*Figure 29. 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 UII 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.
- If the object to delete is a Grouping object, it cannot contain another object. If it does, the delete fails. Use ACTION.OBJINST(DELETE\_TREE\_ALL) instead.

### ACTION.OBJINST(DELETE\_TREE\_ALL)

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()
:INSTANCE.SOURCEKEY(UUI_short_name()...)

:OBJECT.TYPE()
:INSTANCE.SOURCEKEY(UUI_short_name()...)
```

*Figure 30. 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)**

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()
:INSTANCE.SOURCEKEY(UUI_short_name()...)

:OBJECT.TYPE()
:INSTANCE.SOURCEKEY(UUI_short_name()...)
```

*Figure 31. 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.

## ACTION.OBJINST

- The OBJECT tag identifies the object type for the object being deleted.
- The INSTANCE tag specifies the UUI 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 UUI in the information catalog to see whether the input object exists.

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

#### Context:

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

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

Figure 32. Using the ACTION.OBJINST tag when merging objects

#### Rules:

- If the object exists, the Information Catalog Manager updates the property values of the object in the information catalog. If the object does not exist, the Information Catalog Manager 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.
- You must have an ACTION.OBJTYPE(MERGE) tag for a given object type earlier in the tag language file than the ACTION.OBJINST(MERGE) tag for the same object type. This ensures that the object type exists in the information catalog that you are importing to before the Information Catalog Manager can add or update (merge) objects.

You cannot use ACTION.OBJTYPE(MERGE) for an object type that belongs to the Program or Attachment categories, because you cannot create new

Program or Attachment object types. However, you can use ACTION.OBJINST(MERGE) with Program objects, without specifying the ACTION.OBJTYPE(MERGE) first.

### **ACTION.OBJINST(UPDATE)**

Updates the value of an object.

#### **Context:**

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

*Figure 33. 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 UUI 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.

---

## **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 Manager 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
```

## ACTION.OBJTYPE

DELETE\_EXT  
MERGE  
UPDATE

### **ACTION.OBJTYPE(ADD)**

Creates the object type.

#### **Context:**

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

*Figure 34. 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. The Information Catalog Manager automatically defines the following required properties for every object type:  
OBJTYPID  
INSTIDNT  
NAME  
UPDATIME  
UPDATEBY
- You cannot add object types that belong to the Program or Attachment categories.

### **ACTION.OBJTYPE(APPEND)**

Appends a property to an existing object type.

#### **Context:**

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

*Figure 35. 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 UUI.
- 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.
- You cannot append to object types that belong to the Attachment category.

**ACTION.OBJTYPE(DELETE)**

Deletes the object type.

**Context:**

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

*Figure 36. 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.
- You cannot delete object types that belong to the Program or Attachment categories.

**ACTION.OBJTYPE(DELETE\_EXT)**

Deletes the object type and objects of that object type.

**Context:**

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

*Figure 37. 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

- You cannot delete object types that belong to the Program or Attachment categories.

### **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 Manager 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 Manager creates a new object type.

#### **Context:**

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

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

*Figure 38. 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.
- Before you can merge objects, you must merge object types to ensure that a valid object type exists in the target information catalog. Therefore, an ACTION.OBJTYPE(MERGE) tag must appear before an ACTION.OBJINST(MERGE) tag in the tag language file.
- You cannot merge object types that belong to the Program or Attachment categories.

### **ACTION.OBJTYPE(UPDATE)**

Changes an object-type external name and ICON file information.

**Context:**

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

*Figure 39. 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.

**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, CONTAIN, or LINK relationship.

**Context:**

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE() SOURCETYPE() TARGETTYPE()
:INSTANCE.SOURCEKEY(UII_short_name()...) TARGETKEY(UII_short_name()...)
```

*Figure 40. 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 Manager writes an informational message and continues processing.

## ACTION.RELATION

- 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 UUI property values that identify the two objects that are being associated.

### ACTION.RELATION(DELETE)

Deletes a relationship.

#### Context:

```
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE() SOURCETYPE() TARGETTYPE()
:INSTANCE.SOURCEKEY(UUI_short_name()...) TARGETKEY(UUI_short_name()...)
```

*Figure 41. Using the ACTION.RELATION tag when deleting relationships*

#### Rules:

- The relationship is deleted if it exists; otherwise, the Information Catalog Manager 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 UUI property values that identify the two associated objects.

---

## COMMENT

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

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

### Syntax

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

*Figure 42. 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 Manager tags (for example :ACTION.), because each tag ends either at the end of the file or at the beginning of the next valid tag.

---

## COMMIT

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

If the Information Catalog Manager 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 Manager 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 Manager cancels when it rolls back the information catalog.

Frequent COMMIT checkpoints make the echo file easier to read if there are errors in the tag language file. When the COMMIT tag is processed successfully, the Information Catalog Manager clears the echo file of the tags that were processed before the COMMIT tag. The echo file then contains only tags that describe uncommitted changes.

## 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 43. Example of a COMMIT tag*

# COMMIT

## Keywords

### CHKPID

Required keyword.

### *checkpt\_id*

An identifier that the Information Catalog Manager 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 Manager 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.

---

## DISKCNTL

Identifies the diskette sequence number when the tag language file is stored on one or more diskettes.

### Context

When one tag language file is stored on one or more diskettes, DISKCNTL is the first tag on each diskette.

### Syntax

```
:DISKCNTL.SEQUENCE(nn, + | -)
```

```
:DISKCNTL.SEQUENCE(01,+)
```

*Figure 44. Example of a DISKCNTL tag for the first of a sequence of diskettes*

## Keywords

### SEQUENCE

Required keyword

*nn* A one-digit or two-digit number that indicates the number of the diskette in sequence.

The first number for any sequence of disks must be 1 or 01. This value increases by 1 for subsequent diskettes. The numbers for a set of three diskettes are 1, 2, 3, or 01, 02, 03.

- + Additional diskettes containing the tag language file follow this one.
- The last or only diskette that contains the tag language file.

## Rules

If this tag is specified, it must be the first tag in each tag language file. If the tag is missing and the tag language file is on diskette, the import program assumes that the tag language file is contained on one diskette.

If a tag language file is stored on the hard disk, this tag is not applicable. If the tag is present, it is ignored.

---

## INSTANCE

Defines or identifies objects or relationships to be acted on.

### Context

This tag is required following:

- |                         |   |
|-------------------------|---|
| <b>:ACTION.OBJINST</b>  | The INSTANCE tag follows an OBJECT tag. |
| <b>:ACTION.RELATION</b> | 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*) . . .

# INSTANCE

## Context:

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

*Figure 45. Using the INSTANCE tag when adding objects*

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

*Figure 46. Using the INSTANCE tag when merging objects*

## Keywords:

### *short\_name*

Identifies each property by its 8-character short name. This value is not case sensitive; you can specify this value by using uppercase or lowercase characters. If an INSTANCE tag has multiple short names associated with it, use only one INSTANCE tag followed the short names as shown in Figure 46.

### *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 UII 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. However, if an omitted property is a required property with a CHAR, VARCHAR, or LONG VARCHAR data type, a not-applicable symbol is generated and stored in the information catalog. If an omitted required property has a TIMESTAMP data type, then the Information Catalog Manager generates and stores the value 9999-12-31-24.00.00.000000.
- When merging an object:
  - You must specify all UII 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. However, if the defined object does not exist, and the omitted property is required, then a not-applicable symbol is generated and stored in the information catalog.

## **ACTION.OBJINST(DELETE) or ACTION.OBJINST(DELETE\_TREE\_ALL) or ACTION.OBJINST(DELETE\_TREE\_REL)**

Deleting an object

```
:INSTANCE.SOURCEKEY(UUI_short_name (UUI_property_value) . . . )
```

### **Context:**

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

*Figure 47. Using the INSTANCE tag when deleting objects*

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

*Figure 48. Using the INSTANCE tag when deleting Grouping category objects and contained objects*

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

*Figure 49. Using the INSTANCE tag when deleting Grouping category objects and relationships*

### **Keywords:**

#### **SOURCEKEY**

Specifies the UUI property values that identify a particular object.

SOURCEKEY must be the first keyword of the INSTANCE tag.

#### *UUI\_short\_name*

Identifies a UUI property name by its 8-character short name. Specify all of the *UUI\_short\_name(UUI\_property\_value)* combinations. The *UUI\_short\_name* is not case sensitive; you can specify this value by using uppercase or lowercase characters.

#### *UUI\_property\_value*

Specifies the value of a UUI property for a particular object. This value is case sensitive.

## INSTANCE

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

### **ACTION.OBJINST(UPDATE)**

Updating property values for an object

```
:INSTANCE.SOURCEKEY(UII_short_name (UII_property_value) . . . )  
                          short_name (property_value) . . .
```

### **Context:**

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

*Figure 50. Using the INSTANCE tag when updating objects*

### **Keywords:**

#### **SOURCEKEY**

Specifies the UII property values that identify a particular object.

SOURCEKEY must be the first keyword of the INSTANCE tag.

#### *UII\_short\_name*

Identifies a UII property by its 8-character short name. The *UII\_short\_name* is not case sensitive; you can specify this value by using uppercase or lowercase characters.

#### *UII\_property\_value*

This value is case sensitive. With *UII\_short\_name*, specifies the value of a UII property for a particular object.

#### *short\_name*

Identifies the property to be updated by its 8-character short name. The *short\_name* is not case sensitive; you can specify this value by using uppercase or lowercase characters.

You cannot specify the following property short names because you cannot update these properties: OBJTYPID, INSTIDNT, UPDATIME, UPDATEBY.

#### *property\_value*

With *short\_name*, specifies the new value of the property for the given object. This value is case sensitive.

**Rules:** You must specify one *UII\_short\_name(value)* combination for each property that is defined as a UII property for the object type. Each object

type has one or more properties defined as UUI 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(UUI_short_name (UUI_property_value)...)
TARGETKEY(UUI_short_name (UUI_property_value)...)

```

**Context:**

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE() SOURCETYPE() TARGETTYPE()
:INSTANCE.SOURCEKEY(UUI_short_name()...) TARGETKEY(UUI_short_name()...)

```

*Figure 51. Using the INSTANCE tag when adding relationships*

```
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE() SOURCETYPE() TARGETTYPE()
:INSTANCE.SOURCEKEY(UUI_short_name()...) TARGETKEY(UUI_short_name()...)

```

*Figure 52. Using the INSTANCE tag when deleting relationships*

**Keywords:**

**SOURCEKEY**

Specifies the UUI property values that identify the first object in a relationship.

<b>When the relationship is:</b>	<b>The SOURCEKEY identifies:</b>
<b>Contains</b>	The Grouping category object
<b>Contact</b>	The object the contact is for
<b>Attachment</b>	The object the comment is for
<b>Link</b>	Either object to link

SOURCEKEY must be the first keyword of the INSTANCE tag.

**TARGETKEY**

Specifies the UUI property values that identify the second object in a relationship.

<b>When the relationship is:</b>	<b>The TARGETKEY identifies:</b>
<b>Contains</b>	The Elemental category object

## INSTANCE

<b>Contact</b>	The Contact category object
<b>Attachment</b>	The Attachment category object
<b>Link</b>	Either object to link

TARGETKEY must be the second keyword of the INSTANCE tag.

### *UUI\_short\_name*

Identifies a UUI property name by its 8-character short name. This value is not case sensitive; you can specify this value by using uppercase or lowercase characters.

### *UUI\_property\_value*

Specifies the value of a UUI property for a particular object. This value is case sensitive.

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

You must separate each *UUI\_short\_name(value)* and *short\_name(value)* pair with a blank, as shown in Figure 53.

```
:INSTANCE.SOURCEKEY(UUIname1(value1) UUIname2(value2)) sname3(value3) sname4(value4)
```

Figure 53. 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 Manager 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.

---

## NL

Specifies a new line within a property value.

The Information Catalog Manager reads only NL tags that are specified within non-UUI property values and ignores all others.

### Syntax

```
:NL.
```

## Rules

Use NL tags only within the specification of *property\_values* in INSTANCE tags.

---

## 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(type) CATEGORY(category) EXTNAME(ext_name)  
    PHYNAME(table_name) ICOFIELD( )  
    ICWFILE(Windows_ICON_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() CATEGORY() EXTNAME() PHYNAME() ICOFIELD() ICWFILE()  
:PROPERTY.EXTNAME() DT() DL() SHRTNAME() NULLS() UUISEQ()
```

*Figure 54. Using the OBJECT tag when adding object types*

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

*Figure 55. Using the OBJECT tag when merging object types*

#### Keywords:

##### **TYPE**

Specifies the name of an object type.

Required keyword.

##### *type*

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

## OBJECT

The value of *type* 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 *type* already exists, it is used as a search argument.

The maximum length for the value is 8 characters. The value is stored in uppercase 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 *type*.

### CATEGORY

Specifies which category this object type belongs to.

Required keyword.

#### *category*

Specifies an Information Catalog Manager 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.

#### *ext\_name*

Specifies an extended, descriptive name for the object type. The maximum length for *ext\_name* is 80 characters.

This name must be unique within related information catalogs.

The value of *ext\_name* is stored in mixed case.

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.

*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 Manager 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 *type* 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.

*Windows\_ICON\_file\_name*

Specifies the name of the Windows icon file to associate with the object type. The maximum length of *Windows\_ICON\_file\_name* is 254 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 cannot specify the drive and path information that identifies where the icon file resides using this keyword. You must specify this information as an input parameter for the FLGImport API call (see the *Information Catalog Manager Programming Guide and Reference*), the import function on the user interface (see "Importing a tag language file from the command line" on page 185), or the IMPORT option of the DGUIDE command (see "Importing a tag language file from the command line" on page 185).

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.

# OBJECT

## **ACTION.OBJTYPE(APPEND)**

### **Context:**

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

*Figure 56. Using the OBJECT tag when adding properties to object types*

### **Keywords:**

#### **TYPE**

Specifies the name (*type*) of an object type.

Required keyword.

*type*

Identifies a specific object type by its 8-character short name.

## **ACTION.OBJTYPE(DELETE) or ACTION.OBJTYPE(DELETE\_EXT)**

Deleting an existing object type.

### **Context:**

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

*Figure 57. Using the OBJECT tag when deleting object types*

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

*Figure 58. Using the OBJECT tag when deleting object types and all objects of that type*

### **Keywords:**

#### **TYPE**

Specifies the name (*type*) of an object type.

Required keyword.

*type*

Identifies a specific object type by its 8-character short name.

## **ACTION.OBJTYPE(UPDATE)**

Updating object type information.



**Context:**

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

Figure 59. Using the *OBJECT* tag when updating object types

**Keywords:****TYPE**

Specifies the name (*type*) of an object type.

Required keyword.

*type*

Identifies a specific object type by its 8-character short name. You cannot update this value.

**EXTNAME**

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

*ext\_name*

Specifies an extended, descriptive name for the object type. The maximum length for *ext\_name* is 80 characters.

You can update this value.

This name must be unique within related information catalogs.

The value of *ext\_name* is stored in mixed case.

**ICWFILE**

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

Optional keyword.

*Windows\_ICON\_file\_name*

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

You can update this value.

The maximum length of *Windows\_ICON\_file\_name* is 254 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 FLGImport API call, the import function on the user interface, or the IMPORT option of the Information Catalog Manager command.

**ACTION.OBJINST**

Adding, updating, deleting, or merging objects

## OBJECT

### Context:

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

*Figure 60. Using the OBJECT tag when adding objects*

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

*Figure 61. Using the OBJECT tag when merging objects*

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

*Figure 62. Using the OBJECT tag when updating objects*

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

*Figure 63. Using the OBJECT tag when deleting objects*

### Keywords:

#### TYPE

Specifies the name (*type*) of an object type.

Required keyword.

*type*

Identifies a specific object type by its 8-character short name.

---

## 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(ext_name) DT(data_type) DL(data_length)  
                  SHRTNAME(short_name) NULLS(Y | N) UUISEQ(UUI_number)
```

## Context

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

Figure 64. Using the *PROPERTY* tag when adding object types

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

Figure 65. Using the *PROPERTY* tag when merging object types

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

Figure 66. Using the *PROPERTY* tag when adding properties to object types

## Keywords

### EXTNAME

Specifies a descriptive name for the property.

Required keyword.

#### *ext\_name*

Specifies an extended descriptive name.

The maximum length of *ext\_name* is 80 characters. The *ext\_name* must be unique within the object type. *ext\_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 value in either uppercase or lowercase. Valid values are:

- C**     Character
- V**     Variable character

## PROPERTY

**L** Long variable character

**T** Timestamp

### 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:

<b>data_type</b>	<b>Maximum value for data_length</b>
<b>C (character)</b>	Maximum length is 254
<b>V (variable character)</b>	Maximum length is 4000
<b>L (long variable character)</b>	Maximum length is 32700
<b>T (timestamp)</b>	Always 26 characters

### SHRTNAME

Specifies the property short name.

Required keyword.

#### *short\_name*

The short name for the property. The *short\_name* value can be up to 8 characters long. This value can contain only SBCS characters.

This value is stored as uppercase characters; any lowercase characters are converted to uppercase.

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. Do not specify the property short names of the following required properties for every Information Catalog Manager object type: OBJTYPID, INSTIDNT, UPDATIME, or UPDATEBY.

### 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. If no data exists for a required property when an object is added to the information catalog, a not-applicable symbol is entered for the required value for data types of CHAR, VARCHAR, and LONG VARCHAR. For a required value with a data type of TIMESTAMP, the following value is entered:  
9999-12-31-24.00.00.000000

### UUISEQ

Identifies the properties that are used in the UUI.

Optional keyword; the default value is 0. The UUISEQ keyword is optional for properties that are not part of the UUI. The UUI is a set of properties that are defined by the administrator as the key that uniquely identifies each object.

### *UUI\_number*

Specifies the position of the property in the UUI sequence. Valid values are 0, 1, 2, 3, 4, and 5. The value 0 means that the property is not part of the UUI. A nonzero value for *UUI\_number* indicates that the property is part of the UUI.

All object types defined in the tag language file must have at least one property that is part of the UUI. The UUI can consist of up to 5 properties.

At least one property must be defined as part of the UUI.

When assigning *UUI\_number* values to more than one property, the numbers of the UUI properties must range from 1 to the number of properties in the UUI. For example, if three properties are defined as part of the UUI, the *UUI\_number* values must be 1, 2, and 3. You cannot skip numbers in the sequence. The *UUI\_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 UUI when you add a new object type or merge object types. Figure 67 shows the general syntax for identifying NAME as a UUI property.

```
:ACTION.OBJTYPE(ADD)
:OBJECT.TYPE() CATEGORY() EXTNAME() PHYNAME() ICOFILE() ICWFILE()
:PROPERTY.SHRTNAME(NAME) UUISEQ()
```

Figure 67. Example of specifying the NAME property as part of the UUI

Empty parentheses in this figure denote values that you must provide in a tag language file.

- The maximum length of the UUI fields is 254 bytes.

## RELTYPE

---

### RELTYPE

Identifies the type of relationship that 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(CONTAIN | CONTACT | ATTACHMENT | LINK)
                SOURCETYPE(source_type) TARGETTYPE(target_type)
```

#### Context

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE() SOURCETYPE() TARGETTYPE()
:INSTANCE.SOURCEKEY(UII_short_name()...) TARGETKEY(UII_short_name()...)
```

*Figure 68. Using the RELTYPE tag when adding relationships*

```
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE() SOURCETYPE() TARGETTYPE()
:INSTANCE.SOURCEKEY(UII_short_name()...) TARGETKEY(UII_short_name()...)
```

*Figure 69. 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.

##### CONTAIN

Contains relationship: source object contains the target object.

**LINK** Link relationship: source object is linked with the target object.

**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 8 characters. This value is not case sensitive; you can specify this value with uppercase or lowercase characters.

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.

**TARGETTYPE**

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 8 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.

---

**TAB**

Specifies a tab within a property value.

The Information Catalog Manager reads only TAB tags that are specified within non-UII property values and ignores all others.

**Syntax**

**:TAB.**

## TAB

### Rules

Use TAB tags only within the specification of *property\_values* in INSTANCE tags.



---

## Appendix E. What a tag language file should look like

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

---

### Start your tag language file with DISKCNTL

Start the tag language file with a DISKCNTL tag if the file is on a removable disk, such as a diskette. For example:

```
:DISKCNTL.SEQUENCE(01,+)
```

If the tag language file is on more than one diskette, then DISKCNTL must be the first tag in each section of the tag language file on each diskette. If the tag language file is on a fixed disk, then DISKCNTL is ignored.

---

### 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 Manager what you want to do. The keyword tells the Information Catalog Manager what kind of information you want to maintain. The option tells the Information Catalog Manager what task you want to perform.

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

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

```
:ACTION.RELTYPE(option)  
    Maintaining object relationships.
```

#### 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:

Existing object type  
Object type to be merged  
New object type  
New properties for an object type

##### Use these tags:

OBJECT  
OBJECT and PROPERTY  
OBJECT and PROPERTY  
OBJECT and PROPERTY

**To define:**

New or existing object  
 New or existing object relationship

**Use these tags:**

OBJECT and INSTANCE  
 RELTYPE and INSTANCE

**Putting it all together**

The keywords and values that are required for OBJECT, INSTANCE, and PROPERTY 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()
:INSTANCE.short_name() ...

:ACTION.OBJINST(DELETE)
:OBJECT.TYPE()
:INSTANCE.SOURCEKEY(UII_short_name()...)

:ACTION.OBJINST(DELETE_TREE_ALL)
:OBJECT.TYPE()
:INSTANCE.SOURCEKEY(UII_short_name()...)

:ACTION.OBJINST(DELETE_TREE_REL)
:OBJECT.TYPE()
:INSTANCE.SOURCEKEY(UII_short_name()...)

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

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

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

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

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

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

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

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

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

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

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(CONTAIN | CONTACT | ATTACHMENT | LINK) SOURCETYPE(type)
TARGETTYPE(type)
:INSTANCE.SOURCEKEY(UII_short_name()...) TARGETKEY(UII_short_name()...)
:ACTION.RELATION(DELETE)
:RELTYPE.TYPE(CONTAIN | CONTACT | ATTACHMENT | LINK) SOURCETYPE(type)
TARGETTYPE(type)
:INSTANCE.SOURCEKEY(UII_short_name()...) TARGETKEY(UII_short_name()...)
```

For specific information about the format of the `INSTANCE`, `OBJECT`, and `PROPERTY` tags, see “`INSTANCE`” on page 163, “`OBJECT`” on page 169, or “`PROPERTY`” on page 174.

---

## 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 Manager 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
```



---

## Appendix F. Performing Information Catalog Manager functions from the command line

You can perform some Information Catalog Manager functions from an MS-DOS command prompt.

To:	See:
Open an information catalog	"Starting the Information Catalog Manager from the command line"
Import a tag language file into your information catalog	"Importing a tag language file from the command line"
Export metadata from your information catalog	"Exporting metadata from the information catalog" on page 188
Import MDIS metadata into your information catalog	"Importing MDIS-conforming tag language files" on page 98
Export MDIS metadata from your information catalog	"Importing MDIS-conforming tag language files" on page 98
Create an information catalog	"Creating an information catalog from the command line" on page 191

---

### Starting the Information Catalog Manager from the command line

Use the DGUIDE command to open an information catalog from the MS-DOS command line. You can add parameters to the command to import or export a tag language file. "Importing a tag language file from the command line" and "Exporting metadata from the information catalog" on page 188 describe the parameters that you can add to the DGUIDE command.

#### Importing a tag language file from the command line

Use the DGUIDE command to open an information catalog and import a tag language file from an MS-DOS command prompt. When you use the DGUIDE command, keep in mind the following rules for the command syntax:

- None of the parts, except where specified, are case sensitive.
- Each keyword must be preceded by either a slash (/) or hyphen (-) character.
- All keywords that follow /IMPORT as shown in Figure 70 on page 186 are required if you choose to import.
- Underlined choices are defaults.

**DGUIDE** /**USERID** *userid* /**PASSWORD** *password* /**DGNAME** *dgname*

Optional keywords:

**/ADMIN**  
**/TRACE** 0|1|2|3|4  
**/IMPORT** *filename* /**LOGFILE** *filename* /**RESTART** B|C

Optional import keyword:

**/ICOPATH** *iconpath*

*Figure 70. DGUIDE command parameters for opening an information catalog and importing metadata*

The following example shows the required parameters you specify to open the sample information catalog as an administrator.

```
DGUIDE /USERID longods /PASSWORD secret /DGNAME ICMSAMP /ADMIN
```

The following list shows the parameters you can add to the DGUIDE command. Optional and required keywords for importing a tag language file are noted.

#### **/ADMIN**

Specifies that you are logging on as an administrator. If you do not specify this optional keyword for the DGUIDE command, you are logged on as a user, and you cannot perform administrator tasks.

#### **/DGNAME**

Your information catalog name.

If the information catalog is local, give the database name. If the information catalog is remote, give the alias under which it was cataloged.

Example:

```
/DGNAME ICMSAMP
```

#### **/ICOPATH**

Valid only with /IMPORT; optional.

Indicates that you are importing icons and specifies the icon path that the import function will use. The Information Catalog Manager assumes that the path is the same as the one where you installed the Information Catalog Manager unless you specify a full drive and path. You must specify a fixed drive.

Example:

```
/ICOPATH d:\icons\
```

#### **/IMPORT**

Imports the tag language file you specify. Unless you specify the full

drive, path, and file name, the Information Catalog Manager assumes that the file is in the path specified on the DGWPATH environment variable.

Example:

```
/IMPORT d:\tagfile.tag
```

This keyword bypasses the Information Catalog Manager user interface and performs the import function as a batch process.

### **/LOGFILE**

Valid only with /IMPORT; required with /IMPORT.

Specifies the file destination for messages the Information Catalog Manager generates during import. Unless you specify a full drive, path, and file name, the Information Catalog Manager places the file in the path specified on the DGWPATH environment variable. You must specify a fixed drive.

Example:

```
/LOGFILE d:\tagfile.log
```

### **/PASSWORD**

Your password for this user ID.

Example:

```
/PASSWORD secret
```

Passwords are case-sensitive for accessing databases on the following operating systems, you must type them exactly as specified.

- AIX
- Windows NT and Windows 2000
- Solaris Operating Environment

### **/RESTART**

Valid only with /IMPORT; required with /IMPORT.

Indicates which option the import function uses. The valid options are:

- B** Imports the tag language file from the beginning.
- C** The default. Imports the tag language file from the last point at which the Information Catalog Manager successfully committed changes to the information catalog.

### **/TRACE**

The level of trace information to send to the trace file. Each higher level includes the functions of the levels below it (3 includes the functions of levels 0, 1, 2, and 3). You might have to specify a higher level if you call IBM Software Support to diagnose Information Catalog Manager problems.

- 0 The default. Includes all messages and warning, error, and severe error conditions.
- 1 Includes entry and exit records of the highest level Information Catalog Manager functions.
- 2 Includes extremely granular entry and exit records of the Information Catalog Manager functions.
- 3 Includes input and output parameters (excluding input or output structure).
- 4 Includes all input or output structures that are passed to and used by the Information Catalog Manager.

**/USERID**

Your information catalog user ID. Depending on the database location of the information catalog you are opening, type the user ID required by the database. For example, the user ID might be your local, LAN, AS/400, AIX, or OS/390 TSO user ID.

Example:

`/USERID tongods`

**Exporting metadata from the information catalog**

Use the DGUIDE command to export metadata into a tag language file which you can import into another information catalog.

When you use the DGUIDE command to export metadata, keep in mind the following rules for command syntax:

- None of the parts, except where specified, are case sensitive.
- Each keyword must be preceded by either a slash (/) or hyphen (-) character.
- All keywords that follow /EXPORT as shown in Figure 71 are required.
- Underlined choices are defaults.

```
DGUIDE /DGNAME ic_name /USERID userid /PASSWORD password /ADMIN
/EXPORT filename /LOGFILE filename /OBJTYPE
object_type /OBJECTS name
```

Optional keywords:

```
/ADMIN
/A /C /L /T
```

*Figure 71. DGUIDE command parameters for exporting metadata*

**/ADMIN**

Specifies that you are logging on as an information catalog administrator. If you do not specify this optional keyword for the DGUIDE command,



you log on as a user. You can export metadata as a user if your information catalog administrator has given you authority; however, you cannot perform all administrator tasks.

This keyword is optional.

- /A** Specifies that you want to export all comments associated with an exported object.

This keyword is optional.

- /C** Specifies that you want to export all objects that are contained in an exported object.

This keywords is optional.

#### **/DGNAME**

The name of the information catalog from which you want to export.

If the information catalog is local, give the database name. If the information catalog is remote, give the alias under which it was cataloged.

Example:

```
/DGNAME ICMSAMP
```

#### **/EXPORT**

Exports objects to the tag language file you specify. Unless you specify the full drive, path, and file name, the Information Catalog Manager assumes that the file is in the path specified on the DGWPATH environment variable.

Example:

```
/EXPORT d:\tagfile.tag
```

- /L** Specifies that you want to export all objects that have a linked relationship with an exported object.

This keyword is optional.

#### **/LOGFILE**

Specifies the file destination for messages the Information Catalog Manager generates during the export process. Unless you specify a full drive, path, and file name, the Information Catalog Manager places the file in the path specified on the DGWPATH environment variable. You must specify a fixed drive.

Example:

```
/LOGFILE d:\tagfile.log
```

## **/OBJECTS**

Specifies the universal unique identifier (UUI) for each object you want to export. You can specify up to five UUI values separated by periods and enclosed in quotation marks.

Example:

```
/OBJECTS "DWCTARGET.DWCADMIN.CUSTOMER"
```

## **/OBJTYPE**

Specifies the DP NAME (short name) of the object that you want to export.

Example:

```
/OBJTYPE TABLES
```

## **/PASSWORD**

Your password for this user ID.

Example:

```
/PASSWORD secret
```

Passwords are case-sensitive for accessing databases on the following operating systems, you must type them exactly as specified.

- AIX
- Windows NT and Windows 2000
- Solaris Operating Environment

**/T** Specifies that you want to export all contacts that are associated with an exported object.

This keyword is optional.

## **/USERID**

Your information catalog user ID. Depending on the database location of the information catalog you are opening, type the user ID required by the database. For example, the user ID might be your local, LAN, AS/400, AIX, or OS/390<sup>®</sup> TSO user ID.

Example:

```
/USERID hchan
```

In the following example, the administrator hchan exports a relational table object with a UUI of DBNAME(DWCTARGET) OWNER(DWCADMIN) TABLES(CUSTOMER) from the ICMSAMP information catalog. The table object and all columns associated with the object will be exported as well as all contact objects and objects that have linked relationships with the table object. The metadata for the exported objects will be written to a file called x:\mypath\tagfile.tag.

```
DGUIDE /DGNAME ICMSAMP /USERID hchan /PASSWORD mypass /ADMIN /EXPORT
x:\mypath\tagfile.tag /LOGFILE x:\mypath\tagfile.log
/OBJTYPE TABLES /OBJECTS "DWCTARGET.DWCADMIN.CUSTOMER" /c /t /1
```

---

## Creating an information catalog from the command line

To create an information catalog from an MS-DOS command prompt, enter the CREATEIC command. Keep in mind the following rules for the command syntax:

- None of the parts, except where specified, are case sensitive.
- Each keyword must be preceded by either a slash (/) or hyphen (-) character.
- All keywords that follow CREATEIC as shown in Figure 72 are required.
- If plan to store your information catalog in a DB2 UDB for OS/390 database, you must include the required OS/390 keywords.
- Underlined choices are defaults.

```
CREATEIC /DBTYPE dbtype /DGNAME dname /USERID userid /PASSWORD
password /KA1 primary_admin
```

Optional keywords:

```
/NAS -|symbol
/KA2 backup_admin
/MVSDB dbname /TSTORGP table_stor /XSTORGP index_stor
```

Optional OS/390 keyword:

```
/MVSUPPER Y|N
```

*Figure 72. Example of using the DGUIDE command to create an information catalog*

For example, to create an information catalog on a remote DB2 UDB for Windows NT database, type:

```
CREATEIC /DBTYPE DB2NT /DGNAME ICMSAMP /USERID longods /PASSWORD secret /KA1 longods
```

### **/DBTYPE**

Specifies the type of DB2 database in which you want to store your information catalog. Valid choices are:

- |             |  |
|-------------|--|
| <b>DB22</b> | Specifies a DB2 UDB for OS/2 database.   |
| <b>DB2</b>  | Specifies a DB2 UDB for OS/390 or DB2 for OS/390 database. You must have the DB2 Connect product installed on your workstations to use DB2 UDB for OS/390. |

- DB2400** Specifies a DB2 UDB for AS/400 database. You must have either the DB2 Connect product on your workstations to use DB2 UDB for AS/400.
- DB2NT** Specifies a DB2 UDB for Windows NT database or a DB2 UDB for Windows 2000 database. You must have TCP/IP or NetBIOS installed on your workstations to use the remote database.
- DB2 Family** Specifies a DB2 UDB database on other operating systems, for example:
- **DB2 UDB for AIX or DB2 UDB for Solaris Operating Environment**  
You must have TCP/IP installed on your workstation to use the remote database.
  - **DB2 UDB EEE**  
Use the requirements for the operating system on which your database resides.

**/DGNAME**

Specifies the database name. If the database is local, specify the database name. If the database is remote, specify the alias name for the remote database that is cataloged on your local workstation.

**/USERID**

Specifies the user ID required by the database that stores your information catalog:

**DB2 UDB for OS/2 (local)**

Local user ID, specified with UPM on your workstation

**DB2 UDB for OS/2 (remote)**

LAN user ID, specified with UPM on the remote workstation

**DB2 UDB for OS/390**

RACF user ID

**DB2 UDB for AS/400**

AS/400 user ID

**DB2 UDB for AIX**

AIX user ID

**DB2 UDB EEE**

Use the user ID required for the operating system on which your database resides.

**DB2 UDB for Windows NT or DB2 UDB for Windows 2000 (local)**

Windows NT user ID

**DB2 UDB for Windows NT or DB2 UDB for Windows 2000 (remote)**  
LAN user ID, specified with User Manager on the remote workstation

**/PASSWORD**

Specifies the password for the user ID that you entered on the /USERID keyword.

Passwords are case-sensitive for accessing databases on the following operating systems, you must type them exactly as specified.

- AIX
- Windows NT and Windows 2000
- Solaris Operating Environment

**/NAS**

Specifies the character you want to use to indicate property values that are not applicable. You can choose from the following special characters:

!	;	#	\$	%	*	(
)	+	,	-	.	/	:
{	}	=	?	@	[	]
-						

The default is a hyphen (-).

**/KA1**

Specifies the user ID of the person who will be the primary administrator of the Information Catalog Manager. This user ID must have SYSADM (or ALLOBJ authority if your information catalog is stored in a DB2 UDB for AS/400 database) authority.

**/KA2**

Specifies the user ID of the person who will back up the primary administrator. This user ID must have database administrator authority.

**/MVSDB**

Valid only with /DBTYPE DB2; required with /DBTYPE DB2.

Specifies the name of the DB2 UDB for OS/390 database.

**/TSTORGP**

Valid only with /DBTYPE DB2; required with /DBTYPE DB2.

Specifies the name of the storage group that you will use for tables.

**/XSTORGP**

Valid only with /DBTYPE DB2; required with /DBTYPE DB2.

Specifies the name of the storage group that you will use for indexes.

**/MVSUPPER**

Valid only with /DBTYPE DB2; optional with /DBTYPE DB2.

Indicates whether you want to save the property values of each object in uppercase.

- Y The default. Specifies that the values are stored in the OS/390 database in uppercase, but you can enter the values in lowercase when you search for them.
- N Specifies that the values are stored in the OS/390 database exactly as you enter them and that all the information catalog searches are case sensitive.

### **Importing common object types into the information catalog**

After you create an information catalog, you can use the DGUIDE command to import common object types that you can use to exchange metadata with other conforming products.

Before you import the common object types, you must first append all the Information Catalog Manager common object types to a tag language file. From the \SQLLIB\DGWIN\TYPES directory located on the drive where the DB2 Universal Database is installed, enter the following copy command at MS-DOS command prompt.

```
copy *.typ + *.app mytag.tag
```

The metadata for the common object types is copied to mytag.tag. Next, use the DGUIDE command to import mytag.tag into your information catalog as in the following example (do not enter the line breaks):

```
DGUIDE /USERID valdezma /PASSWORD secret /DGNAME ICMSAMP /IMPORT  
d:\Program Files\sqllib\dgwin\types\mytag.tag /LOGFILE  
d:\Program Files\sqllib\dgwin\mytag.log  
/ICOPATH d:\Program Files\sqllib\dgwin\types /RESTART B
```

---

## 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 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 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.

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

IBM Canada Limited  
Office of the Lab Director  
1150 Eglinton Ave. East  
North York, Ontario  
M3C 1H7  
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 information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

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



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

This information 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 illustrates 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, which may be denoted by an asterisk(\*), are trademarks of International Business Machines Corporation in the United States, other countries, or both.

ACF/VTAM	IBM
AISPO	IMS
AIX	IMS/ESA
AIX/6000	LAN DistanceMVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	OS/2
BookManager	OS/390
CICS	OS/400
C Set++	PowerPC
C/370	QBIC
DATABASE 2	QMF
DataHub	RACF
DataJoiner	RISC System/6000
DataPropagator	RS/6000
DataRefresher	S/370
DB2	SP
DB2 Connect	SQL/DS
DB2 Extenders	SQL/400
DB2 OLAP Server	System/370
DB2 Universal Database	System/390
Distributed Relational Database Architecture	SystemView
DRDA	VisualAge
eNetwork	VM/ESA
Extended Services	VSE/ESA
FFST	VTAM
First Failure Support Technology	WebExplorer
	WIN-OS/2

The following terms are trademarks or registered trademarks of other companies:

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

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

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, other countries, or both.

UNIX is a registered trademark in the United States, other countries or both and is licensed exclusively through X/Open Company Limited.

Other company, product, or service names, which may be denoted by a double asterisk(\*\*) may be trademarks or service marks of others.



---

## Glossary

### A

**administrator.** A person responsible for managing the content and use of the Information Catalog Manager.

**anchor.** A Grouping object that contains other objects, but is not contained by another Grouping object.

**Attachment.** The category for object types used to attach additional information to another Information Catalog Manager object. For example, you can attach comments to an object.

### B

**browse.** To display information catalog objects that are grouped by subject. Contrast with *search*.

### C

**catalog.** See *information catalog* and *database catalog*.

**category.** A classification for Information Catalog Manager object types. The category designates the:

- Actions available to object types
- Relationships allowed between object types in the same or different categories.

Object types belong to one of the following categories:

- Attachment
- Contact
- Dictionary
- Elemental
- Grouping
- Program
- Support

**CellDial sample data.** A sample information catalog (ICMSAMP) available when you install the Information Catalog Manager that can be

used for installation verification. This sample information catalog is also used in the exercises in the *Information Catalog Manager User's Guide*.

**collection.** A container for objects. A collection can be used to gather objects of interest for easy access.

**Comments.** A classification for objects that annotate another object in the Information Catalog Manager. For example, you may want to attach a Comments object to a chart object that contains notes about the data in the chart.

The Comments object type is with the Information Catalog Manager. You cannot add properties to it.

**commit.** To make changes to information catalog database permanent. Contrast with *roll back*.

**contact.** A reference for more information about an object. Further information might include the person who created the information that the object represents, or the department responsible for maintaining the information.

**Contact.** A category for the Contact object type and other object types that identify contacts.

**Contact object type.** A classification for objects that identify contacts.

### D

**database catalog.** A collection of tables that contains descriptions of database objects such as tables, views, and indexes.

**DBCS.** 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 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 Manager 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.** Data that identifies and describes an object, for example, the name of a table, the location of a spreadsheet, or the creator of a document. Also called metadata.

**Description view.** A view that lists the properties and property values for an object.

**Dictionary.** The category for object types that can be used to define terminology (for example, the “Glossary entries” object type in the sample information catalog).

**dictionary facility.** A collection of definitions or synonyms for the business terms you use in the information catalog. After it is created, the dictionary facility appears in every user’s Catalog window as a saved search icon.

**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. Contrast with *single-byte character set*.

**DP NAME.** An identification for an object type that uniquely identifies it for import operations. Also called the short name of an object type.

## E

**echo file.** A file produced by the Information Catalog Manager when it imports a tag language file. This file contains all the tags that have been processed since either the beginning of the tag language file or the point when the last COMMIT tag was processed.

**Elemental.** The category for non-Grouping object types that are the building blocks for other Information Catalog Manager object types. Elemental object types are at the bottom of object type hierarchies. “Columns in relational tables,” “Presentations {electronic and hardcopy},” and “Graphics and Images” are all examples of Elemental object types.

**export.** To copy metadata from the Information Catalog Manager, translate the metadata into tag language, and put this output in a tag language file for a subsequent import operation.

**external name.** The 80-byte name for an object type. Also called object type name.

**extract control file.** A file that contains statements that control the operation of an extractor utility program.

**extract program.** A utility program that copies 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 used to allocate space on a disk for a file and to locate the file.

**FLGID.** See *object identifier*.

## G

**Grouping.** The category for object types that can contain other object types. Examples of Grouping object types available in the sample information catalog included with the Information Catalog Manager are: “Tables or views in a relational database,” which contains the Elemental object type “Columns in relational tables”; and “Multi-dimensional model,” which contains another Grouping object type “Dimension.”

## H

**HPFS.** High-performance file system. In OS/2, an installable file system that uses high-speed buffer storage, known as a cache, to provide fast access to large disk volumes. File names used with the HPFS can have as many as 254 characters.

## I

**import.** To apply the contents of a tag language file to an Information Catalog Manager to initially populate the information catalog, change the information catalog contents, or copy the contents of another information catalog to the information catalog.

**information catalog.** The database managed by the Information Catalog Manager containing descriptive data that helps users identify and locate the data and information available to them in the organization.

**Information Catalog Manager application program interface (API).** The portion of the Information Catalog Manager that processes application program requests for the Information Catalog Manager services and functions.

**information source.** An item of data or information, such as a table or chart, that is represented by an Information Catalog Manager object.

**informational application.** A program or system that lets users retrieve and analyze their data.

**informational database.** A database that contains derived data and is intended for business decision making.

**input structure.** A self-defining data structure used to submit data to the Information Catalog Manager application program interface.

**instance.** See *object*.

**instance identifier.** A 10-digit numeric identifier generated by the Information Catalog Manager

for each object. The identifier is unique for that object within a given object type (an object of another object type may have the same identifier), and within a given information catalog database (an object in another information catalog database may have the same identifier).

**I/O structure.** See *input structure* and *output structure*.

## K

**keyword.** An element of the Information Catalog Manager tag language that identifies the meaning of a data value imported into or exported out of an information catalog.

**keyword search.** See *search*.

## L

**link.** A connection between two or more objects involved in a linked relationship.

**linked relationship.** A relationship between objects in an information catalog. Objects in a linked relationship are peers, rather than one an underlying object of the other.

For example, in the sample information catalog included with the Information Catalog Manager, the object called **CelDial Sales Information** is linked with various objects describing CelDial advertisements for the year.

**log file.** A file produced by the Information Catalog Manager when it imports a tag language file or exports objects in the information catalog. This file records the times and dates when the import or export started and stopped and any error information for the process.

## M

**metadata.** Data about information sources. See *descriptive data*.

**multiple character wildcard.** A character used to represent any series of characters of any

length. By default, the multiple character wildcard is an asterisk (\*). See also *wildcard* and *single character wildcard*.

## N

**not-applicable symbol.** A character that indicates that a value for a required property was not provided when an object was created. The not-applicable symbol is a hyphen (-) by default, but you could have identified a different symbol when you created the information catalog.

## O

**object.** An item that represents a unit or distinct grouping of information. Each Information Catalog Manager 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 identifier.** A 16-digit identifier for an object that is made up of its 6-digit object type identifier and its 10-digit instance identifier that is used with some API calls. See *object type identifier* and *instance identifier*.

**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 Manager provides a set of sample object types, which you can modify. You can also create additional object types to meet the needs of your organization.

**object type identifier.** A 6-digit numeric identifier generated by the Information Catalog Manager for each object type. The identifier is unique within the information catalog database.

**object type registration.** With the Information Catalog Manager application program interface, the basic information about an object type that you must define in the Information Catalog Manager before you can define the properties for the object type. This information includes the

category, the name, the icon, and the name of the table containing the object information.

**operational data.** Data used to run the day-to-day operations of an organization.

**option.** In Information Catalog Manager tag language, a parameter of the ACTION tag that defines the action to be performed on objects or object types in the information catalog database when the tag language file is imported.

**output structure.** A self-defining data structure produced by the Information Catalog Manager when returning data produced by an Information Catalog Manager API call.

## P

**physical type name.** The name of the table in the information catalog database that contains metadata for instances of a specific object type.

**populate.** To add object types, objects, or metadata to the Information Catalog Manager.

**Program category.** The category for the Programs object type.

**Programs object type.** A classification for objects that identify and describe applications capable of processing the actual information described by Information Catalog Manager objects.

The Programs object type is included with the Information Catalog Manager.

**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 are assigned to the properties.



**property name.** The 80-byte descriptive name of a property that is displayed in the Information Catalog Manager user interface. Contrast with *property short name*.

**property short name.** An 8-character name used by the Information Catalog Manager to uniquely identify a property of an object or object type.

**property value.** The value of a property.

**PT NAME.** See *physical type name*.

## R

**RDBMS.** Relational database management system.

**RDBMS catalog.** A set of tables that contain descriptions of SQL objects, such as tables, views, and indexes, maintained by an RDBMS.

**relational database management system.** A software system, such as DB2 UDB for OS/2, that manages and stores relational data.

**registration.** See *object type registration*.

**roll back.** To remove uncommitted changes to the information catalog database. Contrast with *commit*.

## S

**saved search.** A set of search criteria that is saved for subsequent use. Appears as an icon in the Catalog window.

**SBCS.** Single-byte character set.

**search.** To request the display of the Information Catalog Manager objects that meet specific criteria.

**search by subject.** See *browse*.

**search by term.** See *search*.

**search criteria.** Options and character strings used to specify how to perform a search. This 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*.

**single character wildcard.** A character used to represent any single character. By default, the single character wildcard is a question mark (?). See also *wildcard* and *multiple character wildcard*.

**subject search.** See *browse*.

**Support.** The category for object types that provide additional information about your information catalog or enterprise (for example, the "Information Catalog Manager News" object type in the sample information catalog).

**support facility.** A collection of information you consider helpful for users of your information catalog, such as announcements of changes or updates to the information catalog. After it is created, the support facility appears in every user's Catalog window as a saved search icon.

## 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. A tag language file is produced by exporting objects from the Data Warehouse Center or from the Information Catalog Manager.

In the Information Catalog Manager, a tag language file is also produced by:

- Transferring a delete history log.

- Extracting descriptive data from another database system using an extract program.

**Tree view.** A view that displays hierarchically an object and the objects it contains.

## U

**unit of work.** A recoverable sequence of operations within an application process. A unit of work is the basic building block a database management system uses to ensure that a database is in a consistent state. A unit of work is ended when changes to the database are committed or rolled back.

**universal unique identifier (UUI).** A key for an object. The key is comprised of up to five properties, which, when concatenated in a designated order, uniquely identify the object during import and export functions.

**user.** A person who accesses the information available in the information catalog but who is not an administrator.

Some Information Catalog Manager users, if they have been granted authority, can perform some object management tasks normally performed by administrators.

## W

**wildcard.** A special character that is used as a variable when specifying property values in a search. See also *single character wildcard* and *multiple character wildcard*.

---

## Bibliography

To get copies of the books listed here, or to get more information about a particular library, see your IBM representative.

### *Data Warehouse Center publications:*

*DB2 Warehouse Manager Installation Guide*  
(GC26-9998)

*Data Warehouse Center Administration*  
*Guide* (SC26-9993)

*Information Catalog Manager Programming*  
*Guide and Reference* (SC26-9997)

*Information Catalog Manager User's Guide*  
(SC26-9996-00)

*Data Warehouse Center Application*  
*Integration Guide* (SC26-9994-00)

See the *OLAP Setup and User's Guide* (SC27-0702-00) for information on the IBM OLAP server products.

You can find comprehensive descriptions of Information Catalog Manager messages and reason codes in *DB2 Universal Database Message Reference* (GC09-2978). In addition, see the Information Catalog Manager online help for descriptions of messages.

## Bibliography

---

# Index

## Special Characters

*Information Catalog Manager User's Guide*, creating sample data for 115

### A

access information catalog, users can't 106  
ACTION tag  
    OBJINST keyword 150, 165  
    OBJTYPE keyword 155  
    planning for extract program 72  
    RELATION keyword 159  
    sequence 182  
    tag language reference 150, 160  
    tips 181  
ADD option  
    ACTION.OBJINST 151  
    ACTION.OBJTYPE 156, 169  
    ACTION.RELATION 159  
administrator  
    logon problems, resolving 108  
    resetting with CLEARKA command 108  
ALTERKA command, changing  
    Information Catalog Manager administrators with 21  
APPEND option 156  
Application data sample object type 122  
ASCII text 72  
Attachment category  
    Comments object type defined 128  
    definition of 27  
    relationships  
        modifying 61  
        summary of 27  
ATTACHMENT keyword 178  
Audio clips sample object type 125

### B

backing up Information Catalog Manager database 106  
backup administrator, identifying 21  
blanks removed from variable values 150  
Business subject areas sample object type 122

### C

C (CHAR) 35  
category  
    Attachment  
        Comments object types defined 128  
        copying comments 59  
        creating comments 58  
        definition of 27  
        deleting comments 60  
        relationships with other categories 27  
        updating comments 59  
    Contact  
        definition of 26  
        object type in sample information catalog 127  
        People to contact object type in sample information catalog 127  
        relationships with other categories 27  
    definition of 26  
    Dictionary  
        creating dictionary facility 68  
        definition of 26  
        Glossary entries object type in sample information catalog 127  
        object type in sample information catalog 127  
        relationships with other categories 27  
    Elemental  
        Audio clips object type in sample information catalog 125  
        Charts object type in sample information catalog 126  
        definition of 26  
        Documents object type in sample information catalog 126  
        Images or graphics object type in sample information catalog 126  
        Internet documents object type in sample information catalog 126

category (*continued*)

    Elemental (*continued*)

        Lotus Approach queries object type in sample information catalog 126  
        object types in sample information catalog 125  
        Presentations object type in sample information catalog 126  
        relationships with other categories 27  
        Spreadsheets object type in sample information catalog 126  
        Text-based reports object type in sample information catalog 127  
        Video clips object type in sample information catalog 127

    Grouping

        Application data object type in sample information catalog 122  
        Business subject areas object type in sample information catalog 122  
        Columns or fields object type in sample information catalog 122  
        Databases object type in sample information catalog 122  
        definition of 26  
        Dimensions within a multi-dimensional database object type in sample information catalog 123  
        Elements object type in sample information catalog 123  
        Files object type in sample information catalog 123  
        IMS database definitions (DBD) object type in sample information catalog 123

- category (*continued*)
  - IMS program control blocks (PCB) object type in sample information catalog 123
  - IMS program specification blocks (PSB) object type in sample information catalog 124
  - IMS segments object type in sample information catalog 124
  - Members within a
    - multi-dimensional database object type in sample information catalog 124
  - Multi-dimensional databases
    - object type in sample information catalog 124
  - object types in sample information catalog 121
  - Records object type in sample information catalog 124
  - Relational tables and views
    - object type in sample information catalog 124
  - relationships with other categories 27
  - Subschemas object type in sample information catalog 125
  - Transformations object type in sample information catalog 125
- Program
  - definition of 26
  - relationships with other categories 27
- Program, Programs object type defined 128
- Support
  - creating support facility 68
  - definition of 26
  - Information Catalog Manager
    - news object type in sample information catalog 127
  - object types in sample information catalog 127
  - Online news services object type in sample information catalog 128
  - Online publications object type in sample information catalog 128
  - relationships with other categories 27
- CATEGORY keyword 35, 169
- CelDial business scenario
  - object type property specifications 128
  - predefined object type descriptions 121
  - sample information catalog provided with Information Catalog Manager, creating 115
- CHAR data type for object type property 32
- character data type
  - optional property 40
  - property of DL 35
- character data type, PROPERTY tag 175
- Charts sample object type 126
- checkpoint, commit 84
- checkpoint tags 76
- checkpt\_id identifier 162
- CHKPID keyword 162
- CLEARKA command for resetting
  - logged-on administrator user ID 108
- code
  - extended, finding what it means 84
  - reason, finding what it means 84
- Columns or fields sample object type 122
- command
  - ALTERKA, for changing Information Catalog Manager administrators 21
  - CLEARKA, for resetting logged-on administrator user ID 108
  - DGUIDE, for opening an information catalog 185
  - DGWDEMO, for creating sample DB2 UDB for Windows NT information catalog 115
  - IMPORT 35, 39
- comment
  - attaching to objects 61
  - copying for an object 59
  - creating for an object 58
  - deleting from an object 60
  - detaching from objects 61
  - updating for an object 59
- comment status list
  - setting values for users 22
  - shown in Create Comment window 22
- COMMENT tag
  - planning for extract programs 72
  - tag language reference 160
  - tips 183
- Comments object type 128
- commit call 76
- commit checkpoint 84, 161
- COMMIT tag
  - planning for extract program 72
  - tag language reference 161, 162
  - tips 183
- committing to database 76
- common object types
  - importing from command line 194
- concurrent access 106
- Contact category
  - definition of 26
  - object type
    - People to contact, provided in sample information catalog 127
    - sample information catalog, provided in 127
  - relationships
    - Attachment, adding 61
    - Attachment, removing 61
    - modifying 57
    - Program, adding 64
    - Program, removing 67
    - summary of 27
- CONTACT keyword 178
- CONTAIN keyword 178
- Contains relationship, modifying 53
- customized extract programs 72
- D**
- data
  - corrupt, what to do 107
  - inconsistent, what to do 107
  - recovering 109
- data type
  - C (CHAR) 35, 40
  - L (LONG VARCHAR) 35, 40
  - T (TIMESTAMP) 35, 40
  - V (VARCHAR) 35, 40
- data types 149, 175
- Data Warehouse Center
  - metadata
    - identifying for publishing 93
    - scheduling for update 95
    - viewing in the information catalog 93
- database
  - alias name 14

- database (*continued*)
    - backup 107
    - local, registering 14
    - maintenance tips 107
    - remote, registering 14
    - rollback 183
    - supported by Information Catalog Manager 3
  - Databases sample object type 122
  - DB2 Connect 3
  - DB2 UDB EEE information
    - catalog 2
  - DB2 UDB for AIX information
    - catalog, defining 9
  - DB2 UDB for AS/400 information
    - catalog, defining 7
  - DB2 UDB for OS/2 2
    - Directory Tool 14
    - information catalog, defining 4
    - log files 105
  - DB2 UDB for OS/390 information
    - catalog, defining 5
  - DB2 UDB for Solaris Operating Environment Solaris information
    - catalog, defining 9
  - DB2 Universal Database
    - Enterprise-Extended Edition information catalog 2
  - DB2 Universal Database for Windows 2000 information catalog, defining 11
  - DB2 Universal Database for Windows NT information catalog, defining 11
  - DBCS 148
  - DELETE\_EXT option on ACTION.OBJTYPE 157
  - delete history
    - creating 77
    - importing 78
  - DELETE option
    - ACTION.OBJINST 152
    - ACTION.OBJTYPE 157
    - ACTION.RELATION 160
    - OBJINST keyword 165
  - DELETE\_TREE\_ALL option
    - ACTION.OBJINST 152
    - OBJINST keyword 165
  - DELETE\_TREE\_REL option
    - ACTION.OBJINST 153
    - OBJINST keyword 165
  - descriptive data, extracting 72
  - desktop applications, extract programs for 113
  - DGMDISC command, for converting Information Catalog Manager tag language to MDIS metadata
    - , specifying 98
    - database user ID, specifying 98
    - input Information Catalog Manager tag language file, specifying 98
    - output MDIS file, specifying 98
    - password, specifying 98
    - syntax of 98
  - DGUIDE command
    - ADMIN keyword 186
    - DGNAME keyword, for specifying information catalog 186
    - IMPORT keywords
      - ICOPATH 186
      - LOGFILE 187
      - RESTART 187
    - PASSWORD keyword 187
    - TRACE keyword 187
    - USERID keyword 188
  - DGUIDE command for invoking Information Catalog Manager 98, 100
    - ADMIN keyword 99, 101
    - DGNAME keyword, for specifying information catalog 99, 101
    - LOGFILE, MDIS\_IMPORT keyword 99, 101
    - MDIS\_EXPORT keywords
      - LOGFILE 99, 101
      - OBJECT 102
      - OBJTYPE 102
    - PASSWORD keyword 100, 102
    - TRACE keyword 100, 103
    - USERID keyword 100, 103
  - Dictionary category
    - creating dictionary facility 68
    - definition of 26
    - object type
      - Glossary entries, provided in sample information catalog 127
      - sample information catalog, provided in 127
    - relationships
      - Attachment, adding 61
      - Attachment, removing 61
      - Program, adding 64
      - Program, removing 67
      - summary of 27
  - dictionary facility, creating 68
  - Dimensions within a multi-dimensional database sample object type 123
  - disk space, monitoring 105
  - DISKCNTRL tag
    - extract program 72
    - tag language reference 162
    - tips 181
  - DL keyword 175
    - defining optional properties 41
    - defining properties 35
  - Documents sample object type 126
  - double-byte character set (DBCS) 148
  - DT keyword 175
    - defining optional properties 40
    - defining properties 35
- ## E
- echo (ECH) file 82, 183
    - definition of 82
    - example of 82
    - problem diagnosis 110
    - reading 82
    - solving import problems with 82
  - Elemental category
    - definition of 26
    - object types
      - Audio clips, provided in sample information catalog 125
      - Charts, provided in sample information catalog 126
      - Documents, provided in sample information catalog 126
      - Images or graphics, provided in sample information catalog 126
      - Internet documents, provided in sample information catalog 126
      - Lotus Approach queries, provided in sample information catalog 126
      - Presentations, provided in sample information catalog 126
      - sample information catalog, provided in 125
      - Spreadsheets, provided in sample information catalog 126

- Elemental category (*continued*)
    - Text-based reports, provided in sample information catalog 127
    - Video clips, provided in sample information catalog 127
  - relationships
    - Attachment, adding 61
    - Attachment, removing 61
    - Contact, adding 57
    - Contact, removing 57
    - Contains, adding 53
    - Contains, removing 53
    - linked, adding 55
    - linked, removing 55
    - Program, adding 64
    - Program, removing 67
    - summary of 27
  - Elements sample object type 123
  - EUI ix
  - examples
    - combining two information catalogs 75
    - echo file 82
    - log file 83
    - messages with reason codes, using 84
    - Programs object, required for
      - Class 63
      - Identifier 63
      - Name 63
      - Qualifier 1, 2, 3 63
    - properties 74
    - tag language file 37
    - trace file 111
    - UUI 74
  - export
    - options
      - default in settings notebook 79
      - specifying during export 79
  - exporting
    - icon files from information catalog 79
    - metadata from information catalog 79
    - solving problems from 83
  - extended code, finding what it means 84
  - external name
    - changing 39
    - for object type, rules for specifying 29
  - external name (*continued*)
    - for property, rules for specifying 32
    - of object type 35
    - of object type property 36
  - external name property 30
  - EXTNAME keyword
    - creating object types 35, 36
    - on OBJECT tag 169, 173
    - on PROPERTY tag 175
    - optional property 41
    - updating object type 39
  - extract program
    - creating new objects 73
    - customized 72, 77
    - desktop applications 113
    - input and output 72
    - installing files 72
    - steps for running documented in README files 113
  - extracting descriptive data 72
- F**
- file
    - echo 82
    - insufficient space for, what to do 105
    - log 83
    - lost, what to do 106
    - trace 110
  - Files sample object type 123
  - filling your information catalog 43
  - formatting tag language files 73
- G**
- Glossary entries sample object type 127
  - Grouping category
    - definition of 26
    - object types
      - Application data, provided in sample information catalog 122
      - Business subject areas, provided in sample information catalog 122
      - Columns or fields, provided in sample information catalog 122
      - Databases, provided in sample information catalog 122
      - Dimensions within a multi-dimensional database, provided in sample information catalog 123
  - Grouping category (*continued*)
    - object types (*continued*)
      - Elements, provided in sample information catalog 123
      - Files, provided in sample information catalog 123
      - IMS database definitions (DBD), provided in sample information catalog 123
      - IMS program control blocks (PCB), provided in sample information catalog 123
      - IMS program specification blocks (PSB), provided in sample information catalog 124
      - IMS segments, provided in sample information catalog 124
      - Members within a multi-dimensional database, provided in sample information catalog 124
      - Multi-dimensional databases, provided in sample information catalog 124
      - Records, provided in sample information catalog 124
      - Relational tables and views, provided in sample information catalog 124
      - sample information catalog, provided in 121
      - Subschemas, provided in sample information catalog 125
      - Transformations, provided in sample information catalog 125
    - relationships
      - Attachment, adding 61
      - Attachment, removing 61
      - Contact, adding 57
      - Contact, removing 57
      - Contains, adding 53
      - Contains, removing 53
      - linked, adding 55
      - linked, removing 55
      - Program, adding 64
      - Program, removing 67
      - summary of 27
    - grouping objects by subject 51



## H

HANDLES keyword 65  
hide system generated properties 30  
history, delete 78  
home page, URL for Information Catalog Manager 113

## I

ICMSAMP sample information catalog  
  object types defined in 121  
ICMSAMP sample information catalog, creating 115  
ICOFIELD keyword 169  
  creating object types 35  
  tag language reference 173  
  updating object type 39  
icon representing object type  
  changing 38  
  changing for OS/2 38  
  changing for Windows 38  
  exporting 79  
  identifying 30  
ICWFILE keyword 169  
  creating object types 35  
  tag language reference 173  
  updating object type 39  
Images or graphics sample object type 126  
IMPORT command 35, 39  
importing  
  delete history 78  
  restarting from checkpoint 84  
  solving problems from 82, 83  
  tag language files 78  
IMS database definitions (DBD)  
  sample object type 123  
IMS program control blocks (PCB)  
  sample object type 123  
IMS program specification blocks (PSB) sample object type 124  
IMS segments sample object type 124  
information catalog  
  combining with another information catalog  
    merging object types before 75  
    with tag language files 71  
  corrupt data in, what to do 107  
  DB2 UDB for AIX, defining 9  
  DB2 UDB for AS/400, defining 7  
  DB2 UDB for OS/2, defining 4

information catalog (*continued*)  
  DB2 UDB for OS/390, defining 5  
  DB2 Universal Database for Windows 2000, defining 11  
  DB2 Universal Database for Windows NT, defining 11  
  defining 3  
  establishing object types in 27  
  exporting  
    icon files for objects 79  
    metadata from 79  
  exporting MDIS metadata from 98, 100  
  exporting metadata using command line 188  
  extract programs for populating 113  
  filling 72  
  importing  
    delete history tag language files 77, 78  
    restarting from checkpoint 84  
    tag language files 78  
  importing MDIS metadata into 98, 100  
  inconsistent data in, what to do 107  
  local, registering 14  
  maximum allowed object types 28  
  migrating 14  
  objects  
    copying in 45  
    creating in, using Information Catalog Manager tag language 44  
    creating in, using Information Catalog Manager windows 43  
    deleting from, using Information Catalog Manager tag language 49  
    deleting from, using Information Catalog Manager windows 48  
    updating in, using Information Catalog Manager tag language 47  
    updating in, using Information Catalog Manager windows 46  
  opening from command line 185

information catalog (*continued*)  
  opening from user interface 18  
  planning 25  
  populating with objects 43  
  refreshing 72  
  remote, registering 14  
  sample provided with Information Catalog Manager  
    creating 115  
    object types defined in 121  
    predefined program objects 128  
  setting up 1, 25  
  users can't access, what to do 106  
Information Catalog Manager  
  opening information catalog from the command line 185  
  opening information catalog from user interface 18  
Information Catalog Manager administrator  
  changing in Manage Information Catalog Manager Users window 21  
  changing with ALTERKA command 21  
Information Catalog Manager news sample object type 127  
input to customized extract program 72  
installing extract programs 72  
instance identifier property 30  
INSTANCE tag  
  ACTION.OBJINST  
    (ADD) 163  
    (DELETE) 165  
    (DELETE\_TREE\_ALL) 165  
    (DELETE\_TREE\_REL) 165  
    (MERGE) 163  
    (UPDATE) 166  
  ACTION.RELATION  
    (ADD) 167  
    (DELETE) 167  
  planning for extract program 72  
  tag language reference 163, 168  
INSTIDNT property 30  
Internet documents sample object type 126  
invocation parameters  
  opening the Invocation Parameters window to specify 62  
  recommended value for programs 64

- K**
- keyword
    - ATTACHMENT 178
    - CATEGORY 35, 169
    - CHKPID 162
    - CONTACT 178
    - CONTAIN 178
    - context-sensitive 182
    - DL
      - defining optional properties 41
      - defining properties 35
    - DL tag language reference 175
    - DT
      - defining optional properties 40
      - defining properties 35
    - DT tag language reference 175
    - EXTNAME 41
      - creating object types 35, 36
      - on OBJECT tag 169, 173
      - on PROPERTY tag 175
      - updating object type 39
    - HANDLES 65
    - ICOFIELD 35
      - updating object type 39
    - ICOFIELD, optional keyword on OBJECT 169, 173
    - ICWFIELD
      - creating object types 35
      - updating object type 39
    - ICWFIELD, optional keyword on OBJECT 169, 173
    - LINK 178
    - NAME 65
    - not supported for national languages 148
    - NULLS 36
    - OBJTYPE 155
    - PARMLIST 65
    - PHYNAME 35, 169
    - RELATION 159
    - RELTYPE 178, 179
    - SEQUENCE 163
    - SHRTDESC 65
    - SHRTNAME
      - as a property 45
      - creating object types 35
      - optional property 40
    - SOURCEKEY 165
      - ACTION.OBJINST 166
      - ACTION.RELATION 167
      - associating contacts 57
      - deleting object 68
      - using parentheses 53
  - keyword (*continued*)
    - SOURCETYPE 178
      - associating contacts 57
      - defining grouping 53
      - defining linked relationship 55
    - STARTCMD 65
    - TARGETKEY 167
      - associating contacts 57
      - using parentheses 53
    - TARGETTYPE 178
      - associating contacts 57
      - defining grouping 53
      - defining linked relationship 55
    - TYPE
      - creating object types 35
      - creating objects 45
      - creating optional property 40
      - deleting object types 42
      - deleting objects 49
      - OBJTYPE(ADD) 169
      - OBJTYPE(APPEND) 172
      - OBJTYPE(DELETE) 172, 174
      - OBJTYPE(MERGE) 169
      - OBJTYPE(UPDATE) 173, 174
      - RELTYPE 178
        - updating object type 39
      - UUI\_short\_name 53, 55, 57
      - UUICLASS 65
      - UUIIDENT 65
      - UUIQUAL1, 2, 3 65
      - UUISEQ 36, 175
  - L**
    - L (LONG VARCHAR) 35
    - last changed by 30
    - last changed date and time property 30
    - LINK keyword 178
    - linked relationship, creating 55
    - list, comment status 22
    - log (LOG) file
      - DB2 UDB for OS/2 105
      - definition of 82
      - example of 83
      - Information Catalog Manager 105
      - location of 83
      - reading 83
      - solving export problems with 83
      - solving import problems with 83
  - logging on to Information Catalog Manager
    - from user interface 18
  - logging on to the Information Catalog Manager
    - from the command line 185
  - LONG VARCHAR data type for object type property 32
  - long variable character data type
    - optional property 40
    - property of DL 35
  - long variable character data type, PROPERTY tag 175
  - Lotus Approach queries sample object type 126
  - M**
    - maximum
      - LONG VARCHAR properties for object type 32
      - object types for an information catalog 28
      - properties for an object type 30
      - recommended length of UUI 34
    - MDIS
      - described 96
      - predefined object types that map to 115
        - Columns or fields 122
        - Databases 122
        - Dimensions within a multi-dimensional database 123
        - Elements 123
        - Files 123
        - IMS database definitions (DBD) 123
        - IMS program control blocks (PCB) 123
        - IMS program specification blocks (PSB) 124
        - IMS segments 124
        - Members within a multi-dimensional database 124
        - Multi-dimensional databases 124
        - Records 124
        - Relational tables and views 124
        - Subschemas 125
        - Transformations 125
      - tag language files, exporting 100
      - tag language files, importing 98
      - URL for Web site 96

- MDISDGC command, for converting MDIS metadata to Information Catalog Manager tag language
  - database user ID, specifying 97
  - input MDIS file, specifying 97
  - log file, specifying 98
  - output Information Catalog Manager tag language file, specifying 97
  - password, specifying 97
  - syntax of 97
- Members within a multi-dimensional database sample object type 124
- MERGE option
  - ACTION.OBJINST 154
  - ACTION.OBJTYPE 158, 169
- messages, using 108
- metadata
  - exchanging with other products
    - non-IBM, MDIS-conforming products 96
  - exporting from Information Catalog Manager 79
  - extracting from other products 72
  - Information Catalog Manager, exchanging with other information catalogs 71
  - mapping Data Warehouse Center with Information Catalog Manager 133
  - mapping with Information Catalog Manager and OLAP server 143
  - MDIS 96
    - tag language files, converting from 97
    - tag language files, converting to 98
  - publishing
    - Data Warehouse Center 85
    - identifying Data Warehouse Center metadata to publish 93
    - OLAP server 85
  - publishing and exchanging with other products 85
  - publishing and synchronizing
    - establishing environment for 86
  - publishing OLAP server 87
  - publishing to the information catalog
    - identifying OLAP objects 87
- metadata (*continued*)
  - publishing to the information catalog (*continued*)
    - scheduling in Data Warehouse Center 95
  - synchronizing Data Warehouse Center 92
- Metadata Interchange Specification 96
- migrating an information catalog 14
- Multi-dimensional databases sample object type 124
- N**
- name
  - external
    - of object type 35
    - of object type property 36
  - short
    - of object type 35
    - of object type property 35
- NAME keyword 65
- NAME property 30
- names
  - rules for specifying
    - object type (external) name 29
    - property name 32
    - short, for object type 30
    - short, for property 32
- national language support (NLS) 148
- NetBIOS
  - node 14
- NL tag 168
  - planning for extract programs 72
- not-applicable symbol
  - default value of 34
  - for DB2 UDB for AIX information catalog, selecting 10
  - for DB2 UDB for AS/400 information catalog, selecting 8
  - for DB2 UDB for OS/2 information catalog, selecting 4
  - for DB2 UDB for OS/390 information catalog, selecting 7
  - for DB2 UDB for Windows 2000, information catalog, selecting 13
- not-applicable symbol (*continued*)
  - for DB2 UDB for Windows NT information catalog, selecting 13
  - identified when creating information catalog 34
  - specifying for use during MDIS export 102
- NULLS keyword 36, 175
- O**
- object
  - adding contacts
    - steps for, using Information Catalog Manager tag language 57
    - steps for, using Information Catalog Manager windows 56
  - adding to another information catalog 71
  - attaching comments to 61
  - copying, steps for 45
  - creating 73
    - steps for, using Information Catalog Manager tag language 44
    - steps for, using Information Catalog Manager windows 43
  - definition of 43
  - deleting
    - steps for, using Information Catalog Manager tag language 49
    - steps for, using Information Catalog Manager windows 48
  - detaching comments from 61
  - exchanging with object from another information catalog 71
  - grouping with other objects
    - preparation for 51
    - steps for, using Information Catalog Manager tag language 53
    - steps for, using Information Catalog Manager windows 51
  - linking with other objects
    - steps for, using Information Catalog Manager tag language 55
    - steps for, using Information Catalog Manager windows 54

- object (*continued*)
  - properties 73
  - relationship
    - Attachment 61
    - Contact 57
    - Contains 53
    - link 55
  - removing contacts
    - steps for, using Information Catalog Manager tag language 57
    - steps for, using Information Catalog Manager windows 56
  - updating
    - steps for, using Information Catalog Manager tag language 47
    - steps for, using Information Catalog Manager windows 46
- OBJECT tag
  - ACTION.OBJTYPE
    - (ADD) 169
    - (APPEND) 172
    - (DELETE) 172
    - (DELETE\_EXT) 172
    - (MERGE) 169
    - (UPDATE) 172
  - planning for extract program 72
  - tag language reference 169, 174
- object type
  - associating programs with
    - preparation for 61
    - steps for, using Information Catalog Manager tag language 64
    - steps for, using Information Catalog Manager windows 62
  - Attachment category
    - Comments object type defined 128
  - category
    - definition of 26
  - Comments
    - copying 59
    - creating 58
    - deleting 60
    - updating 59
  - common, importing from
    - command line 194

- object type (*continued*)
  - Contact category
    - People to contact, provided in sample information catalog 127
    - sample information catalog, provided in 127
  - copying program association
    - for 66
  - creating
    - preparation for 28
    - steps for, using Information Catalog Manager windows 29
    - steps for, using tag language 35
  - definition of 25
  - deleting
    - steps for, using Information Catalog Manager windows 41
    - steps for, using tag language 41
  - Dictionary category
    - creating dictionary facility with 68
    - Glossary entries, provided in sample information catalog 127
    - sample information catalog, provided in 127
  - disassociating programs with
    - steps for, using Information Catalog Manager tag language 67
    - steps for, using Information Catalog Manager windows 67
  - Elemental category
    - Audio clips, provided in sample information catalog 125
    - Charts, provided in sample information catalog 126
    - Documents, provided in sample information catalog 126
    - Images or graphics, provided in sample information catalog 126
    - Internet documents, provided in sample information catalog 126

- object type (*continued*)
  - Elemental category (*continued*)
    - Lotus Approach queries, provided in sample information catalog 126
    - Presentations, provided in sample information catalog 126
    - sample information catalog, provided in 125
    - Spreadsheets, provided in sample information catalog 126
    - Text-based reports, provided in sample information catalog 127
    - Video clips, provided in sample information catalog 127
  - establishing in information catalog 27
  - exporting 79
  - Grouping category
    - Application data, provided in sample information catalog 122
    - Business subject areas, provided in sample information catalog 122
    - Columns or fields, provided in sample information catalog 122
    - Databases, provided in sample information catalog 122
    - Dimensions within a multi-dimensional database, provided in sample information catalog 123
    - Elements, provided in sample information catalog 123
    - Files, provided in sample information catalog 123
    - IMS database definitions (DBD), provided in sample information catalog 123
    - IMS program control blocks (PCB), provided in sample information catalog 123
    - IMS program specification blocks (PSB), provided in sample information catalog 124

- object type *(continued)*
  - IMS segments, provided in sample information catalog 124
  - Members within a multi-dimensional database, provided in sample information catalog 124
  - Multi-dimensional databases, provided in sample information catalog 124
  - Records, provided in sample information catalog 124
  - Relational tables and views, provided in sample information catalog 124
  - sample information catalog, provided in 121
  - Subschemas, provided in sample information catalog 125
  - Transformations, provided in sample information catalog 125
- icon representing
  - changing 38
  - changing for OS/2 38
  - changing for Windows 38
  - identifying 30
- identifying descriptive data for extraction 113
- limits for an information catalog 28
- merging to facilitate adding to another information catalog 75
- name (external name), rules for specifying 29
- Program category, Programs object type defined 128
- property
  - adding 30
  - adding during update 38
  - data types for 32
  - five common properties defined by Information Catalog Manager 30
  - maximum allowed LONG VARCHAR properties 32
  - maximum recommended length of UUI 34
  - part of UUI 33
  - rules for UUI 34
  - steps for adding, using Information Catalog Manager tag language 35

- object type *(continued)*
  - property *(continued)*
    - steps for adding, using Information Catalog Manager windows 31
  - relationship
    - Program, adding 64
    - Program, removing 67
  - relationships between 27
  - short name, rules for specifying 30
  - Support category
    - creating support facility with 68
    - Information Catalog Manager news, provided in sample information catalog 127
    - Online news services, provided in sample information catalog 128
    - Online publications, provided in sample information catalog 128
    - sample information catalog, provided in 127
  - updating
    - steps for, using Information Catalog Manager windows 38
    - steps for, using tag language 39
  - updating program association for steps for, using Information Catalog Manager windows 66
- object type identifier property 30
- objects
  - exporting 79
  - importing 78
- OBJTYPID property 30
- OLAP server
  - metadata
    - scheduling updates 91
  - metadata mapping with Information Catalog Manager 143
- OLAP Server
  - metadata
    - identifying for publish 87
- online information and messages 108
- Online news services sample object type 128
- Online publications sample object type 128

- option
  - ACTION.RELATION 167
  - ADD
    - ACTION.OBJINST 151
    - ACTION.OBJTYPE 156, 169
    - ACTION.RELATION 159, 167
  - APPEND 156
  - DELETE 167
    - ACTION.OBJINST 152
    - ACTION.OBJTYPE 157
    - ACTION.RELATION 160
    - on OBJINST 165
  - DELETE\_EXT 157
  - DELETE\_TREE\_ALL
    - ACTION.OBJINST 152
    - on OBJINST 165
  - DELETE\_TREE\_REL
    - ACTION.OBJINST 153
    - on OBJINST 165
  - MERGE
    - ACTION.OBJINST 154
    - ACTION.OBJTYPE 158, 169
  - UPDATE
    - ACTION.OBJINST 155, 166
    - ACTION.OBJTYPE 158
- output from customized extract program 72

**P**

- parameters, invocation
  - opening the Invocation Parameters window to specify 62
  - recommended value for programs 64
- parentheses, use of 53, 57
- PARMLIST keyword 65
- People to contact sample object type 127
- PHYNAME keyword 35, 169
- populating your information catalog 43
- Presentations sample object type 126
- primary administrator, identifying 21
- problems with Information Catalog Manager
  - closes unexpectedly 110
  - preventing 105
  - recovering from system failure 109
  - solving 105, 111
  - tools for solving 108

- program, extract
  - available from Information Catalog Manager Web site 113
  - steps for preparing to run 113
  - steps for running documented in README files 113
  - supplied with Information Catalog Manager 113
- Program category
  - definition of 26
  - relationships
    - Attachment, adding 61
    - Attachment, removing 61
    - summary of 27
- Program category, Programs object type defined 128
- programs
  - associating with object types
    - using Information Catalog Manager tag language 64
    - using Information Catalog Manager windows 62
  - copying associating with object types 66
  - disassociating with object types
    - using Information Catalog Manager tag language 67
    - using Information Catalog Manager windows 67
  - starting from object types, invocation parameters for 64
  - updating association with object types, using Information Catalog Manager windows 66
- programs, starting from the Information Catalog Manager 19
- Programs object type 128
- programs that can be invoked from information catalog objects 128
- property 122
  - adding
    - overview 30
    - steps for, using Information Catalog Manager tag language 35
    - steps for, using Information Catalog Manager windows 31
  - data types
    - CHAR 32
    - LONG VARCHAR 32
    - TIMESTAMP 32
    - VARCHAR 32
  - definition of 26
  - external name (NAME) 30
- property 122 (*continued*)
  - five common properties defined by Information Catalog Manager
    - hiding 30
    - instance identifier 30
    - last changed by 30
    - last changed date and time 30
    - object type identifier 30
  - instance identifier (INSTIDNT) 30
  - last changed by (UPDATEBY) 30
  - last changed date and time (UPDATIME) 30
  - maximum recommended length of UUI 34
  - NAME 37
  - name (external), rules for specifying 32
  - object 73
  - object type identifier (OBJTYPID) 30
  - optional 40
  - part of UUI 33
  - Programs object, required for
    - Class 63
    - Identifier 63
    - Qualifier 1, 2, 3 63
  - rules for UUI 34
  - short name, rules for specifying 32
- PROPERTY tag 72, 174, 178
- R**
  - reading syntax diagrams 150
  - Records sample object type 124
  - recovery, data 109
  - register
    - information catalog 14
    - server node
      - using DB2 Control Center 14
  - related publications 207
  - Relational tables and views sample object type 124
  - relationship
    - adding with tag language 53
    - between object types 27
    - between objects, modifying
      - Attachment 61
      - Contact 57
      - Contains 53
      - linked 55
    - deleting with tag language 53
  - RELTYPE tag 72, 178, 179
  - reserved words 147
  - restarting the echo file 183
  - restrictions for extract program 77
  - rolling back data 183
- S**
  - sample information catalog
    - object types defined in 121
    - predefined program objects 128
  - sample information catalog, creating 115
  - SEQUENCE keyword 163
  - setting up information catalog 1, 25
  - settings notebook
    - default export options specified in 79
    - specifying whether to display 5 common properties 30
  - short name
    - for object type, rules for specifying 30
    - for property, rules for specifying 32
    - of object type 35
    - of object type property 35
  - short\_name 47
  - SHRTDESC keyword 65
  - SHRTNAME keyword 175
    - as a property 45
    - creating object types 35
    - optional property 40
  - SOURCEKEY keyword
    - ACTION.OBJINST (DELETE) 166
    - ACTION.RELATION 167
  - associating contacts 57
  - deleting object 68
  - tag language reference 165
    - using parentheses 53
  - SOURCETYPE keyword 178
    - associating contacts 57
    - defining grouping 53
    - defining linked relationship 55
  - space, monitoring 105
  - Spreadsheets sample object type 126
  - STARTCMD keyword 65
  - status list, comment 22
  - Subschemas sample object type 125
  - Support category
    - creating support facility 68
    - definition of 26
    - object types
      - Information Catalog Manager news, provided in sample information catalog 127

- Support category (*continued*)
  - Online news services, provided in sample information catalog 128
  - Online publications, provided in sample information catalog 128
  - sample information catalog, provided in 127
- relationships
  - Attachment, adding 61
  - Attachment, removing 61
  - Program, adding 64
  - Program, removing 67
  - summary of 27
- support facility, creating 68
- symptom
  - closes, unexpectedly 110
  - corrupt data 107
  - inconsistent data 107
  - Information Catalog Manager failing 105
  - lost files 106
  - users can't access information catalog 106
- syntax diagrams 150
- syntax rules for tag language 147
- system administration 2
- system failure, recovering from 109
- system-generated properties, hiding 30
- system registry information for Information Catalog Manager for Windows 107

## T

- T (TIMESTAMP) 35, 40
- TAB tag 179
  - planning for extract programs 72
- tag language
  - defining information 181
  - editing using word processor 72
  - extract programs for producing supplied with Information Catalog Manager 113
  - writing customized 72
- file
  - converting from MDIS 97
  - converting to MDIS 98
  - cutting and pasting online templates into ix
  - formatting 73
  - how Information Catalog Manager reads 148

- tag language (*continued*)
  - file (*continued*)
    - importing into information catalog 78
    - MDIS-conforming, importing and exporting 98, 100
    - for creating object types 35, 73
    - for deleting object types 41
    - for merging object types 75
    - for updating object types 39
    - merging object types 76
    - overview 147
    - performing Information Catalog Manager tasks with ix
    - reference 147, 179
    - syntax rules 147
    - templates to cut and paste provided online ix
- tags
  - ACTION
    - OBJINST keyword 165
    - planning for extract program 72
    - sequence 182
    - tag language reference 150, 160
    - tips 181
  - COMMENT
    - planning for extract programs 72
    - tag language reference 160
    - tips 183
  - COMMIT
    - during imports 76
    - planning for extract program 72
    - tag language reference 161, 162
    - tips 183
    - contextual use of 181
  - DISKCNTRL
    - extract program 72
    - tag language reference 162
    - tips 181
  - INSTANCE 72, 163, 168
  - NL 168
    - planning for extract programs 72
  - not supported for national languages 148
  - NULLS 175
  - OBJECT 72
  - PROPERTY 72, 174, 178
  - RELTYPE 72
  - TAB 179

- tags (*continued*)
  - planning for extract programs 72
  - to define information 181
- TARGETKEY keyword 167
  - associating contacts 57
  - using parentheses 53
- TARGETTYPE keyword 178
  - associating contacts 57
  - defining grouping 53
  - defining linked relationship 55
- Text-based reports sample object type 127
- timestamp data type
  - optional property 40
  - property of DL 35
- timestamp data type, PROPERTY tag 175
- TIMESTAMP data type for object type property 32
- trace (TRC) file
  - definition of 110
  - example of 111
  - interpreting 111
  - location of 110
  - rename after Information Catalog Manager closes unexpectedly 110
- Transformations sample object type 125
- troubleshooting Information Catalog Manager 105, 111
- TYPE keyword
  - creating
    - object types 35
    - objects 45
    - optional property 40
  - deleting object types 42
  - deleting objects 49
  - OBJTYPE(ADD) 169
  - OBJTYPE(APPEND) 172
  - OBJTYPE(DELETE) 172, 174
  - OBJTYPE(MERGE) 169
  - OBJTYPE(UPDATE) 173, 174
  - RELTYPE 178
  - updating object type 39

## U

- universal unique identifier
  - property values 166
- universal unique identifier (UII)
  - definition of 33
  - object type requirement 33
  - parts of 34
  - position of property in 74

- universal unique identifier (UUI)
  - (continued)*
  - property values 34
  - rules for properties 34
  - UUI\_short\_name
    - specifying when associating contacts 57
    - specifying when defining grouping 53
    - specifying when defining linked relationship 55
- unsupported tags and keywords 148
- UPDATE option
  - ACTION.OBJINST 155, 166
  - ACTION.OBJTYPE 158
- UPDATEBY property 30
- UPDATIME property 30
- URL, Information Catalog Manager
  - home page 113
- user configuration files 106
- user ID
  - authorizing to Information Catalog Manager for Windows 2
  - changing Information Catalog Manager administrator with ALTERKA command 21
  - resetting logged-on administrator with CLEARKA command 108
- user interface (EUI), performing Information Catalog Manager tasks with ix
- UUI
  - definition of 33
  - object type requirement 33
  - parts of 34
  - position of property in 74
  - property values 34, 166
  - rules for properties 34
  - short\_name 47
  - UUI\_short\_name
    - specifying when associating contacts 57
    - specifying when defining grouping 53
    - specifying when defining linked relationship 55
- UUI\_property\_value 165
- UUI\_short\_name keyword 57
- UUI\_short\_name value 165
- UUICLASS keyword 65
- UUIIDENT keyword 65
- UUIQUAL1, 2, 3 keywords 65
- UUISEQ keyword 36, 175
- UUISEQ option 74

## V

- V (VARCHAR) 35
- VARCHAR data type for object type property 32
- variable character data type
  - optional property 40
  - property of DL 35
- variable character data type, PROPERTY tag 175
- variable values 150
- Video clips sample object type 127

## W

- Web site, URL for Information Catalog Manager 113
- writing customized extract programs 72
- writing tag language files 147



---

## Contacting IBM

If you have a technical problem, please review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. This guide suggests information that you can gather to help DB2 Customer Support to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

---

### Product Information

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) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

**<http://www.ibm.com/software/data/>**

The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more.

**<http://www.ibm.com/software/data/db2/library/>**

The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information.

**Note:** This information may be in English only.

**<http://www.elink.ibm.com/pbl/pbl/>**

The International Publications ordering Web site provides information on how to order books.

**<http://www.ibm.com/education/certify/>**

The Professional Certification Program from the IBM Web site provides certification test information for a variety of IBM products, including DB2.

**ftp.software.ibm.com**

Log on as anonymous. In the directory /ps/products/db2, you can find demos, fixes, information, and tools relating to DB2 and many other products.

**comp.databases.ibm-db2, bit.listserv.db2-l**

These Internet newsgroups are available for users to discuss their experiences with DB2 products.

**On CompuServe: GO IBMDB2**

Enter this command to access the IBM DB2 Family forums. All DB2 products are supported through these forums.

For information on how to contact IBM outside of the United States, refer to Appendix A of the *IBM Software Support Handbook*. To access this document, go to the following Web page: <http://www.ibm.com/support/>, and then select the IBM Software Support Handbook link near the bottom of the page.

**Note:** In some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.





Part Number: CT60MNA  
Program Number: 5648-D35

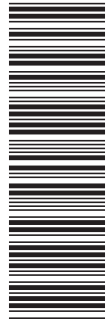


Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC26-9995-00



(1P) P/N: CT60MNA



Spine information:



IBM® DB2® Warehouse  
Manager

Information Catalog Manager Administration  
Guide

Version 7