IBM® DB2 Universal Database™

Data Warehouse Center Application Integration Guide

Version 8

SC27-1124-00

IBM® DB2 Universal Database™

Data Warehouse Center Application Integration Guide

Version 8

SC27-1124-00

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

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

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

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

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

Contents

Part 1. Integrating Applications

Chapter 1. Planning to integrate your applications

How partner applications can work with the Data Warehouse Center and the Information

A *partner application* is an application that runs independently from the Data Warehouse Center and provides some kind of support for a data warehousing solution. You can define the application to the Data Warehouse Center to include it in a warehouse-building process that can include multiple applications.

For example, assume that you want to unload operational data from an IMS™ database, clean the data, and load the cleansed data into a DB2® warehouse database. Users then query the cleansed data. You have three partner applications:

- Partner application 1 unloads data from a database, performs simple transformations, such as joining tables, and writes the transformed data to a warehouse database.
- Partner application 2 cleans the data to prepare the data for the warehouse.
- Partner application 3 queries and reports on the data in the warehouse. It contains metadata about the tables in the warehouse that users can search for specific attributes. Users use the metadata to determine which tables have the data that they need.

You use these three applications together in the following process:

1. Partner application 1 extracts data from multiple segments in a source IMS database.
2. Partner application 1 joins the data from the source segments, and writes the joined data to file 1.
3. Partner application 1 writes the joined data to file 1.
4. Partner application 2 reads the data from file 1.
5. Partner application 2 cleans the data by matching names and by using other data cleansing techniques.
6. Partner application 2 writes the cleansed data to file 2.
7. Partner application 1 reads the data from file 2.
8. Partner application 1 writes the data to a warehouse database.
9. Partner application 3 displays the data in the warehouse or reports about the data in the warehouse when users select tables to query.

Partner application management

You can use Data Warehouse Center steps to manage the process of warehousing. A *step* is a single operation on data in a warehouse process. In most cases, a step includes a warehouse source, the transformation (or movement of data), and a warehouse target.. A step can be run according to a schedule, or it can cascade from another step. You use steps to define and schedule each step in the extraction, transformation, and writing of the data.

A basic step performs the following tasks:

- It extracts data from at least one table or file.
- It uses Data Warehouse Center SQL processing to transform the data, or calls a program that transforms the data.
- It writes the transformed data to a table.

In the partner application example, you define three steps, one for each source-to-target transformation:

- The Unload step performs tasks 1 through 3.
- The Clean step performs tasks 4 through 6.
- The Load step performs tasks 7 and 8.

Because Partner application 3 transforms data at user request in task 9, you do not define a step for task 9.

In the definition of the step, you can schedule a date and time to run the step. At that time, the Data Warehouse Center begins the process that the step defines by issuing SQL statements or starting the program. You can also specify that a second step is to start after the first step finishes processing.

You can schedule the first step to run at a particular date and time. You can schedule the second step to start after the first step runs. You schedule the third step to start after the second step runs. In this manner, you can automate the process of running multiple partner applications.

Managing partner metadata

To define the process of managing metadata, you import partner metadata into the Data Warehouse Center. In this book, *partner metadata* is metadata that partner applications can use and store outside of the Data Warehouse Center.

In the partner application example, you import the following metadata into the Data Warehouse Center:

- From Partner application 1, metadata about the databases, File 1, and the application

- From Partner application 2, metadata about File 2 and the application

You can then publish the metadata about the files to the partner applications so that both partner applications use the same information:

- You export metadata about File 2 to Partner application 1.
- You export metadata about File 1 to Partner application 2.

You can also export metadata from the Data Warehouse Center to the Information Catalog Manager to provide information about the data in the warehouse to users of the warehouse. You can import metadata for the sources and targets, as well as the transformations of the data from its source format to its target format. The users of your warehouse can obtain information about the lineage of the data in the warehouse from the metadata that you import. In the partner application example, you export metadata about the table in the warehouse, Table 3, to the Information Catalog Manager.

You can import metadata into the Information Catalog Manager directly from the Data Warehouse Center.

Integration scenarios

The following table lists some common types of warehousing applications and describes how you can integrate them with the Data Warehouse Center.

Table 1. Integration scenarios

Type of application	Integration process
Data warehousing design	To use data from data warehousing design applications in the Data Warehouse Center: 1. Import metadata into the Data Warehouse Center. 2. Use metadata synchronization to propagate metadata into the Information Catalog Manager.
Operational data descriptions	Import metadata into the Data Warehouse Center and business metadata into the Information Catalog Manager. If the metadata is for source data that is included for lineage only and not to define source tables or files, import the metadata into the Information Catalog Manager directly.

Table 1. Integration scenarios (continued)

Type of application	Integration process
Data cleansing	To clean operational data: 1. Determine which application will manage the movement of the source data and target data: the Data Warehouse Center or the partner application. Different applications can manage the source data and the target data. 2. Import source and target definitions, or export source and target definitions, or both. Do so to avoid typing the definitions again. 3. Define the partner application as a Data Warehouse Center program, or write a Data Warehouse Center program that starts the partner application. 4. Develop a user interface that sets the partner application parameters. 5. Import the metadata into the Data Warehouse Center so that the Data Warehouse enter can run the data cleansing application. You can schedule programs by sequence as well as date and time. 6. Import business metadata into the Information Catalog Manager for use by users.
Alternate data storage (such as DB2 OLAP Server™)	To load operational data into alternate data storage: 1. From the Data Warehouse Center, export the data definitions that are needed to build the partner storage. 2. Define the load programs as a Data Warehouse Center program, or write a Data Warehouse Center program that starts the load programs. 3. Develop a user interface that sets the partner application parameters. 4. Import definitions of the load programs into the Data Warehouse Center. Use the load programs to synchronize the values in the operational data store and in the partner data store. 5. Import business metadata for the partner data store into the Information Catalog Manager.

Table 1. Integration scenarios (continued)

Type of application	Integration process
Reporting (such as Brio or Business Objects)	To integrate reporting applications with the Data Warehouse Center: 1. Export business metadata from the Information Catalog Manager into the report application. 2. Import descriptions of the reports into the Information Catalog Manager. 3. Enable starting of the report application from an information catalog.

The models and templates that are described in this article require Data Warehouse Center Version 7.1 which is available in the DB2 Universal Database package, Information Catalog Manager Administrator Version 7.1 which is available in the Warehouse Manager package, and their prerequisite products.

For information about the prerequisite products for the Data Warehouse Center and the Information Catalog Manager, see the *Quick Beginnings* book for your platform and the *DB2 Warehouse Manager Installation Guide*.

Chapter 2. Importing and exporting metadata

This chapter provides detialed information about how to import metadata directly into, and export metadata directly from, the Data Warehouse Center.

Importing metadata into the Data Warehouse Center

You import metadata into theData Warehouse Center so that the Data Warehouse Center can extract and transform data for the warehouse or run partner applications that extract and transform data.

Importing metadata into the Data Warehouse Center involves the following tasks:

1. Bulid a *tag language file* (a file that contains the metadata for the objects to import).
2. Import the tag language file.
3. Prepare the steps to run on your data warehouse.

Building the tag language file

Selecting objects for which to import metadata

You can import metadata for the following types of objects into the Data Warehouse Center:

Agent sites

A *warehouse agent* performs the actual transfer of data between the source database or file (warehouse source), and the target database (warehouse target). It also performs any transformation of that data. The warehouse agent receives commands from the warehouse server. Then, the agent issues SQL commands, starts a partner application, or starts a Data Warehouse Center program that starts a partner application. A warehouse agent can also import table definitions.

An *agent site* is the machine on which an agent runs. The agent site must have access to the machine that contains the source database and the target database.

Warehouse sources and warehouse targets

A *source database* or *source file* is the database or file from which the Data Warehouse Center or a partner application extracts data for further processing. The generic term *source* means a database or a group of one or more files. A source is associated with one or more tables, files or segments. A table, file or segment is associated with one or more columns or fields. A warehouse source is a subset of

tables and views from a single database, or a set of files, that have been defined to the Data Warehouse Center.

A *warehouse target* or *target file* is the database or file to which the Data Warehouse Center or a partner application writes the data after processing it. The generic term *target* means a database or a group of one or more files. A target is associated with one or more tables or files. A table or file is associated with one or more columns or fields. A warehouse target is a subset of tables, or a set of files, that are managed by the Data Warehouse Center.

A warehouse target is the database that contains the warehouse that users will use to run queries and reports.

Data Warehouse Center programs

A *Data Warehouse Center program* is a user-written or partner application that performs some kind of data transformation. You define the program to the Data Warehouse Center so that you can schedule it to run and monitor its operations as part of a step. A Data Warehouse Center program is generally associated with one or more parameters. You can group related Data Warehouse Center programs together by associating them with a Data Warehouse Center program group.

Subject areas

You use a *subject area* to logically group the processes (and the steps, warehouse sources, and warehouse targets within the processes) that are related to a particular topic or function. For example, if you have a series of processes that move and transform sales data, you create a Sales subject area, and create the processes within the subject area. Similarly, you group Marketing processes under a Marketing subject area.

Processes

A process is a a series of steps, which commonly operates on source data, that changes data from its original form into a form conducive to decision support. A Data Warehouse Center process commonly consists of one or more warehouse sources, one or more steps, and one or more warehouse targets.

Steps A step is a single operation on data in a Data Warehouse Center process. A process commonly consists of one or more warehouse sources, one or more steps, and one or more warehouse targets. In most cases, a step includes a warehouse source, a description of the transformation or movement of data, and a warehouse target. You use steps to define and schedule each step in the extraction, transformation, and writing of the data. The metadata for a step includes the source and target tables on which the Data Warehouse

Center or the partner application is to operate. It also includes the SQL statements to issue or the program to start to perform the transformation.

Cascade relationships between steps
A *cascade relationship* is a schedule for a step that is based on the processing status of another step. You can schedule a step to run after another step finishes running.

Relationships between Data Warehouse Center objects
The metadata for Data Warehouse Center objects describes relationships to other objects. For example, the metadata for a step describes relationships to the warehouse source and warehouse target tables that the step uses.

Object definition for the Data Warehouse Center

Data Warehouse Center object definition:

This concept is part of the topic "Importing objects into the Data Warehouse Center." To define objects that you want to import into the Data Warehouse Center you must first build a tag language file from one or more Data Warehouse Center metadata templates.

Each template corresponds to an object, such as a table, or a subset of an object, such as a column. You combine templates to define all the details about an object. For example, if you want to define a source database, you combine database, table, and column templates.

You must write a program that obtains values from the partner metadata store and use these values to replace tokens in the template. This type of program is called an *interchange program*.

Each template contains tokens for which your interchange program must specify values. For example, the token `*TableDescription` represents the description of a table. Your interchange program would search for `*TableDescription` and change it to the string that contains the description of the table specified in the relational catalog. For a DB2® Universal Database table, the description is in the REMARKS field of the syscat.tables table of the system catalog. Because your interchange program replaces the tokens with a value, you do not need to know the syntax of the underlying tag language that identifies metadata in the file.

Installing the Data Warehouse Center metadata templates

This task is part of the main task for *Defining objects for the Data Warehouse Center metadata templates*. You can choose to install the Data Warehouse Center metadata templates when you install the Application Development Client.

Procedure:

To install the templates:

1. Click **Custom** on the installation Setup Type window.
2. Click **Data Warehouse ISV Toolkit**.
3. Specify the directory where you want to install the templates.

   The default directory for the ISV Toolkit is x:\sqllib\templates.the Data Warehouse Center sets the *VWS_TEMPLATES* environment variable to the location of the ISV Toolkit. Your program can query the value of *VWS_TEMPLATES* to locate the templates.

Once you have specified the directory where you want to install the metadata templates, the Data Warehouse Center installs the files in subdirectories of the directory that is set by *VWS_TEMPLATES*. The table below lists the types of files that are installed by the Data Warehouse Center and the subdirectories in which the files are installed.

Table 2. File types and subdirectories for templates

Type of file	Subdirectory
Templates	ISV
Samples	Samples
Header files	Include

Interchange programs

Interchange program writing

When you write an interchange program, you must:

- Include the header file.
- Copy and change the appropriate templates.
- Append the changed copies of the templates to the tag language file.

You can also log processing messages in the same directory that the Data Warehouse Center uses to log processing messages.

The ISV_defines.h header file:: Use of the ISV_Defines.h header file allows your program logic to stay the same even if the template's tokens change. You simply need to recompile your program.

Copying and changing templates:: Your program must use the following procedure to work with the templates:

1. Use the *VWS_TEMPLATES* environment variable to obtain the directory in which the templates are stored. Append \ISV\ to the value to obtain the complete path for the templates.

2. Read a copy of the templates locally into your program.
3. Search the templates for the tokens in the templates and replace the tokens with the metadata from the partner application.

   Use a search and replace methodology, rather than programming to the format of the tag language file. Use of the tokens enables your program to be independent of changes to the tag language that is used in the template file.

   In the templates, each token is enclosed in parentheses; the closing parenthesis identifies the end of the value. Your program should substitute values for only the token and not remove the parentheses.

   Any string that is to replace a token value must follow the following rules:

   - The string must not contain embedded tab characters.
   - Any parenthesis in the string must be enclosed in single quotation marks.

     For example, if you want to replace the `*DatabaseNotes` token with the value:

     ```
     This is my database (managed by the Finance group).
     ```

     You must change the value to:

     ```
     This is my database '('managed by the Finance group')'.
     ```

   If your interchange program does not have a value for a token, it should replace the token with the constant `ISV_DEFAULTVALUE` (defined in ISV_defines.h). However, you must specify a value other than `ISV_DEFAULTVALUE` for any token that is required.

   Because there is no template for security groups, your program must specify the value `ISV_DEFAULTSECURITYGROUP` for any instances of the *SecurityGroup token.

   The templates use default values for Data Warehouse Center specific metadata. For example, retry count and retry interval for warehouse sources and warehouse targets are set to their Data Warehouse Center default values.

Appending templates to the tag language file:: The tables below show the order in which your program must append templates to the tag language file. They also provide the conditions under which the template is required or optional.

Except for the header, you can define as many copies of each template as you need. You must define only one copy of the header in each tag language file.

Table 3. Relationships between templates and conditions

Order	Template	Required or optional
1	HeaderInfo.tag	Always required
2	AgentSite.tag	Required if you do not use the default agent site
3	VWPGroup.tag	Required if you are defining Data Warehouse Center programs
4	VWPProgramTemplate.tag	Required if you are defining Data Warehouse Center programs
5	VWPProgramTemplateParameter.tag	Required if you are defining Data Warehouse Center programs
6	SourceDataBase.tag WarehouseDataBase.tag	Required if you are defining warehouse sources or warehouse targets
7	Table.tag	Required if you are defining warehouse sources or warehouse targets
8	Column.tag	Required if you are defining warehouse sources or warehouse targets

After you append the Column.tag template to the tag language file, the series of templates and the order in which the templates are appended to the tag language file depend on whether you want to define a step or a star schema.

If you are defining a step, append the following templates to the tag language file in the order shown in Table 4.

Table 4. Relationships between templates and conditions when defining a step

Order	Template	Required or optional
9	SubjectArea.tag	Required if you are defining steps.
10	Process.tag	Required if you are defining steps.
11	Step.tag	Required if you are generating SQL transformations between source and target data or defining programs that the Data Warehouse Center is to execute.

Table 4. Relationships between templates and conditions when defining a step (continued)

12	StepInputTable.tag	Required if you are defining a step of type: ISV_StepType_Editioned_Append ISV_StepType_Full_Replace ISV_StepType_Uneditioned_Append Optional if you are defining a step of type: ISV_StepType_VWP_Population
13	StepOutputTable.tag	Required if you are defining a step of type: ISV_StepType_Editioned_Append ISV_StepType_Full_Replace ISV_StepType_Uneditioned_Append StepOutputTable cannot be used for steps of type: ISV_StepType_VWP_Population
14	StepVWPOutputTable.tag	Optionalif you are defining a step of type: ISV_StepType_VWP_Population
15	StepCascade.tag	Required in order to link steps in a cascaded relationship
16	StepVWPProgramInstance.tag	Required if the step uses a Data Warehouse Center program
17	VWPProgramInstanceParameter.tag	Required if the step uses a Data Warehouse Center program which both expects parameters to be passed and has parameters.

If you are defining a star schema, append the following templates to the tag language file in the order shown in Table 5.

Table 5. Relationships between templates and conditions for defining a star schema

Order	Template	Required or optional

Table 5. Relationships between templates and conditions for defining a star schema (continued)

9	StarSchema.tag	Required if you are defining a star schema.
10	StarSchemaInputTable.tag	Required if you are defining a star schema.
11	AgenttoProgram.tag	Required if the Agent Site specified in the tag language file refers to an existing Data Warehouse Center program in the Data Warehouse Center control database.
12	AgenttoDatabase.tag	Required if the Agent Site specified in the tag language file refers to an existing source or target database in the Data Warehouse Center control database.

Logging processing messages:: Your interchange program can write log processing messages or trace files to the directory that the *VWS_LOGGING* environment variable specifies. The Data Warehouse Center uses this directory for its log files and its trace files.

Defining the header for the Data Warehouse Center tag language file

Prerequisites:

Before you define the objects that a tag language file can contain, you must first define the header.

Restrictions:

The following restrictions apply:

Copying templates: Your program must copy and change the HeaderInfo.tag template file.

Substituting values: Your program must supply the default security group, ISV_DEFAULTSECURITYGROUP.

Procedure:

To define the header for the tag language file, copy the applicable template.

Figure 1 is a pseudocode example of the logic that your program can use to build the header portion of the tag language file.

```
Initialize native metadata environment.
   For a C++ ISV application, include isv_defines.h.
   For a Java ISV application, use ISV_Defines.java.
Read a copy of the HeaderInfo.tag template (from the templates directory).
Include the template without modifications.
Write the output to a target file.
```

Figure 1. Pseudocode for adding the header to the tag language file

The ISV_Sample program provides an example of the header portion of the tag language file. You can find the source code for the program in the Samples subdirectory of the directory that is set by the *VWS_TEMPLATES* environment variable.

Defining sources and targets for the Data Warehouse Center

You define sources if you want the Data Warehouse Center or a partner application to read data from those sources. Similarly, you define targets if you want the Data Warehouse Center or a partner application to write data to those targets.

Restrictions:

The following restrictions apply:

- The source or target must already exist within the warehouse control database.
- You must use only the steps that use Data Warehouse Center programs.

Procedure:

To define sources and targets:

1. Copy the applicable templates.
2. Substitute actual values for tokens.

Source and target definition for the Data Warehouse Center

You define sources if you want the Data Warehouse Center or partner application to read data from those sources. Similarly, you define targets if you want the Data Warehouse Center or partner application to write data to those targets.

For copying templates, you can define the following types of source objects:

- Relational databases
- IMS databases

- File systems
- Files

You can also define relational databases as target objects.

The following table lists the templates that your program must copy and change to define each type of source and target object.

Table 6. Templates for relational source and target definitions

Source or target definition	Number of copies of template	Template to copy	Prerequisite template
Database	One copy for each database you want to use	SourceDataBase.tag WarehouseDataBase.tag	HeaderInfo.tag AgentSite.tag if you are not using the default agent
Table	One copy for each table that you want to define in the database	Table.tag	SourceDatabase.tag WarehouseDataBase.tag
Column	One copy for each column that you want to define in each table	Column.tag	Table.tag

You relate the templates for the tables to the template for the database by specifying common values in the templates. Similarly, you relate templates for the columns to the template for the table by specifying common values in the templates.

Value substitution for the Data Warehouse Center

Your program must obtain values that describe databases or files from the partner metadata store. Your program must substitute the values that it obtains for the appropriate tokens in the template.

Databases

Your program must supply the following metadata about the source databases or the target databases:

- The source databases to define or the target databases to define
- The machines on which the databases reside
- The tables in each database to define
- The columns in each table to define

Files

Your program must supply the following metadata about the source files:

- The file system that contains the files
- The source files to define or target files to define
- The machines on which the files reside
- The fields in each file to define

Program logic

The following is a pseudocode example of the logic that your program can use to create or update data resources for source or target definitions.

```
For each source or target to be defined:
   Read a copy of the SourceDatabase.tag or WarehouseDatabase.tag template
   Search for and replace tokens with the metadata from your native metadata source
      (or defaults)
   Append the output to a target file

   For each table, file, or segment that is to be defined:
      Read a copy of the Table.tag template
      Search for and replace tokens with the metadata from your native metadata source
         (or defaults)
      Append the output to a target file

      For each column or field that the table contains:
         Read a copy of the Column.tag template
         Search for and replace tokens with the metadata from your native metadata source
            (or defaults)
         Append the output to a target file
      End (for each column)
   End (for each table)
End (for each source or target data source)
```

Figure 2. Pseudocode for creating or updating data resources for source and target definitions. Use this logic for each source or target definition that you want to create or update.

The ISV_Sample program provides an example of creating or updating data sources for source or target definitions. You can find the source code for the program in the Samples subdirectory of the directory that is set by the *VWS_TEMPLATES* environment variable.

Related reference:

- "DATABASE object metadata for the Data Warehouse Center" on page 85
- "TABLES object metadata for the Data Warehouse Center" on page 90
- "COLUMN object metadata for the Data Warehouse Center" on page 95

Data Warehouse Center program definitions

If you want the Data Warehouse Center to schedule and run a partner application, you must first define the program as a Data Warehouse Center program.

If your tag language file is to point to Data Warehouse Center programs, you must define the following objects, in order:

1. One or more program groups to contain the Data Warehouse Center programs.
2. One or more Data Warehouse Center program templates, which provide the base definition of the program to the Data Warehouse Center.
3. One or more Data Warehouse Center program template parameters, which provide the default parameters that the Data Warehouse Center passes to the program.

   You can change the parameters that are used in a particular step by defining an instance of the program parameters for the step.

Defining Data Warehouse Center programs

If you want the Data Warehouse Center to schedule and run a partner application, you must first define the application as a Data Warehouse Center program. Then you can schedule and run the program by using it on one or more steps.

Prerequisites:

Before your tag language file can contain Data Warehouse Center programs, you must first define the following objects in order:

1. One or more program groups to contain the Data Warehouse Center programs.
2. One or more Data Warehouse Center program templates, which provide the base definintion of the program, to the Data Warehouse Center.
3. One or more Data Warehouse Center program template parameters, which provide the default parameters that the Data Warehouse Center passes to the program.

Procedure:

To define a Data Warehouse Center program:

1. Copy the applicable template.
2. Substitute actual values for tokens.

Copies to the Data Warehouse Center templates

The following table lists the templates that your program must copy and change to define Data Warehouse Center programs.

Table 7. Templates for Data Warehouse Center programs

Definition	Number of copies of template	Template to copy	Prerequisite template
Data Warehouse Center program group	One copy for each program group to define	VWPGroup.tag	HeaderInfo.tag
Data Warehouse Center program template	One copy for each Data Warehouse Center program in the program group	VWPProgramTemplate.tag	VWPGroup.tag
Data Warehouse Center program template parameter	One copy for each parameter passed to the Data Warehouse Center program	VWPProgramTemplateParameter.tag	VWPProgramTemplate.ag

You relate the templates for the Data Warehouse Center program group to the template for the Data Warehouse Center program by specifying common values in the templates. Similarly, you relate templates for the parameters to the template for the Data Warehouse Center program by specifying common values in the templates.

Data Warehouse Center program logic

Your program must obtain values that describe the Data Warehouse Center programs from the warehouse control database:

- The Data Warehouse Center groups to define.
- The Data Warehouse Center programs to define.
- The parameters in each Data Warehouse Center program to define.

Your program must substitute the values that it obtains for the appropriate tokens in the templates.

The following pseudocode example shows the logic that your program can use to define applications that will be managed and run by the Data Warehouse Center.

```
Read a copy of the SubjectArea.tag template
Search for and replace tokens with the metadata from your native metadata store
(or defaults)
Append the output to a target file
Read a copy of the process

For each step to be defined:
Read a copy of the Step.tag template
Search for and replace tokens with the metadata from your native metadata store
(or defaults)
Append the output to a target file
If the step is to execute your application:
Read a copy of the StepVWPProgramInstance.tag template
Search for and replace tokens with the metadata from your native metadata store
(or defaults)
Append the output to a target file
For each parameter that your application needs:
Read a copy of the VWPProgramInstanceParameter.tag template
Search for and replace tokens with the metadata from your native metadata store
(or defaults)
Append the output to a target file
End (for each parameter)

If the step is to be related to its VWP output target data:
Read a copy of the StepVWPOutputTable.tag template
Search for and replace tokens with the metadata from your native metadata store
(or defaults)
Append the output to a target file
End (step relation to its output)
End (if step to execute your application)

If the step is to be related to its input source data:
Read a copy of the StepInputTable.tag template
Search for and replace tokens with the metadata from your native metadata store
(or defaults)
Append the output to a target file
End (step relation to its source)
If the step is to be related to its output target data:
Read a copy of the StepOutputTable.tag template
Search for and replace tokens with the metadata from your native metadata store
(or defaults)
Append the output to a target file
End (step relation to its target)
End (for each step)
```

Exporting data from the Data Warehouse Center

You export metadata from the Data Warehouse Center if yoiu want your partner application to operate on data sources or targets that are defined in the Data Warehouse Center.

Exporting metadata from the Data Warehouse Center involved the following procedures:

1. Select the objects of which to export metadata.
2. Export the metadata to a tag language file.

Object selection for metadata export

You export metadata from the Data Warehouse Center to a tag language file or Common Warehouse Metamodel XML file if you want your partner application to operate on data sources or targets that are defined in the Data Warehouse Center.

Most Data Warehouse Center objects are specific to the Data Warehouse Center. However, you can use metadata about databases, tables, and columns to define source and target databases for partner applications. You can use this capability to share source and target information between partner applications that transform data for the same warehouse.

For example, one partner tool might unload data from a database into a target file. Another partner tool might use the file as a source file and read data from that file, as well as transform the data and write the data to another data file.

A third partner tool might read the data from the file and load it into a target database. If you export the metadata for the databases and files from the Data Warehouse Center, you can make sure that all the partner tools are using the same data definitions.

To define source databases, export one or more warehouse sources (all tables and columns are included automatically). To define a target database, export a warehouse target (all tables and columns are included automatically).

When you export the objects, the Data Warehouse Center writes the objects in a file. You can export the objects in tag language format or the Common Warehouse Metamodel format.

The following table shows the mapping between the logical Data Warehouse Center objects and the tag language object that represents the logical object.

Table 8. Logical objects for source and target databases

Data Warehouse Center logical object	Object in tag language file	Description
Warehouse Source	DATABASE	Source database or file
Warehouse Target	DATABASE	Target database or file
Table	TABLES	Table, file, or segment in source or target database
Column	COLUMN	Column or field in table or field in file

Related reference:

- "Metadata mappings between the Data Warehouse Center and CWM XML objects and properties" in the *Data Warehouse Center Administration Guide*

Exporting metadata into a tag language file

You can use the Data Warehouse Center user interface or a command window to export metadata from the Data Warehouse Center. This topic describes how to use the command window.

Prerequisites:

Before you can export metadata into a tag language file, you must first create an .INP file with the list of warehouse sources and warehouse targets that you want to export. For example:

```
<IR>
LOG_STAT_IR
LOG_STAT_REP
```

LOG_STAT_IR is a warehouse source, and LOG_STAT_REP is a warehouse target. The Data Warehouse Centerr automatically exports the tables and columns that are associated with LOG_STAT_IR and LOG_STAT_REP.

Restrictions:

The import formats and the export formats are release-dependent. You cannot use exported files from a previous release to migrate from one release of the Data Warehouse Center to another.

Procedure:

To export the tag language file, enter the following at a command prompt:

```
iwh2exp2 INPfilename controlDBname userid password [PREFIX = schema]
```

The following defines the command terms:

INPfilename
: The full path and file name of the .INP file.

  Create this file in a read/write directory because the Data Warehouse Center will write the tag language file in this directory. The Data Warehouse Center names the tag language file *INPfilename*.TAG.

controlDBname
: The name of the control database.

userID
: The user ID required to access the control database.

password
The password that is required to access the control database.

[PREFIX = schema]
The table qualifier for the metadata tables.

If a prefix is not specified, the default value is *IWH*.

Related tasks:

- "Exporting tag language files" in the *Information Catalog Center Administration Guide*
- "Importing tag language files" in the *Information Catalog Center Administration Guide*

Part 2. Metadata reference

Chapter 3. Metadata templates

This chapter provides detailed information about each template that is provided with the Data Warehouse Center and the Information Catalog Center. The section for each template lists the tokens for the template. It provides the allowed values and lengths of values for each token.

If your interchange program does not have a value for a token, it should set the token to ISV_DEFAULTVALUE. However, you must specify a value other than ISV_DEFAULTVALUE for any token that is required.

Because there is no template for security groups, your program must specify the value ISV_DEFAULTSECURITYGROUP for any instances of the **SecurityGroup* token.

If the template does not set a Data Warehouse Center parameter, the Data Warehouse Center definition will have the default value of the parameter. For example, the Data Warehouse sets the Retry Count and Retry Interval parameters for source databases to their default values.

Table 9 lists the metadata templates that are supplied with the Data Warehouse Center and the section that covers each template.

Metadata templates supplied with the Data Warehouse Center

The following table lists the metadata templates that are supplied with the Data Warehouse Center and the topic that covers each template.

Table 9. metadata templates supplied with the Data Warehouse Center

Template	Description
AgentSite.tag	Defines an agent site from which the agent accesses the data source or target warehouse, or on which a Data Warehouse Center program runs.
AgenttoDatabase.tag	Associates an agent site to an existing source or target database.
AgenttoProgram.tag	Associates an agent site to an existing Data Warehouse Center program.
Column.tag	Defines a column or field in a table, segment, or file.

Table 9. metadata templates supplied with the Data Warehouse Center (continued)

Template	Description
Commit.tag	Improves performance when you are using large tag language files.
ForeignKey.tag	Defines foreign key constraints on tables.
ForeignKeyAdditional.tag	Defines a composite foreign key.
HeaderInfo.tag	Contains control information for the Data Warehouse Center import utility.
PrimaryKey.tag	Defines primary key constraints on tables.
PrimaryKeyAdditional.tag	Defines a composite primary key.
Process.tag	Defines a process.
StarSchema.tag	Defines a star schema.
StarSchemaInputTable.tag	Defines the relationship between tables and a star schema.
Step.tag	Defines a step.
StepCascade.tag	Defines a cascade relationship between steps.
StepInputTable.tag	Defines the relationship between a step and its source tables.
StepOutputTable.tag	Defines the relationship between a step and its target.
StepVWPOutputTable.tag	Defines the relationship between a step and a warehouse target.
StepVWPProgramInstance.tag	Defines an instance of a specific template used by a step.
SourceDataBase.tag	Defines a warehouse source.
SubjectArea.tag	Defines a subject area to contain the processes and steps being created.
Table.tag	Defines a table or file that the Data Warehouse Center is to access.
VWPGroup.tag	Defines a group that is to contain anyData Warehouse Center program being defined.
VWPProgramInstanceParameter.tag	Adds or modifies a parameter that the Data Warehouse Center passes to an instance of a Data Warehouse Center program used by a specific step.
VWPProgramTemplate.tag	Defines a Data Warehouse Center program.

Table 9. metadata templates supplied with the Data Warehouse Center (continued)

Template	Description
VWPProgramTemplateParameter.tag	Defines a parameter that the Data Warehouse Center is to pass to a Data Warehouse Center program.
WarehouseDataBase.tag	Defines a warehouse target.

AgentSite.tag template for the Data Warehouse Center

You can use the AgentSite.tag template to define an agent site:

- From which the agent accesses the data sources or target warehouses.
- On which a Data Warehouse Center program runs.

You can use one of the following agent site types:

- An agent site that is already defined in the warehouse control database.

  To use an existing agent site, replace all occurrences of the **AgentSite* token with the agent site name.
- The default agent site.

  To use the default agent site, replace all occurrences of the **AgentSite* token with the `ISV_DEFAULTAGENTSITE`.
- A new agent site that you define using the AgentSite.tag template.

  To define a new agent site, specify values for the tokens in the AgentSite.tag template. Replace all occurrences of the **AgentSite* token with the name of the new agent site.

The following tables provide information about and examples for each token in the template.

Table 10. AgentSite.tag tokens

Token	Description	Allowed values
	Entity parameters	

Table 10. AgentSite.tag tokens (continued)

Token	Description	Allowed values
**AgentSite*	The name of a new agent site, or the name of the default agent site, if the agent is not new. If you specify a new name, it must be unique within the warehouse control database. This token is required, but you can specify the default agent site, `ISV_DEFAULTAGENTSITE`	A text string, up to 80 bytes in length. If you do not want to create a new agent site, use `ISV_DEFAULTAGENTSITE` for the default agent site.
**AgentSiteContact*	The name of the person or organization responsible for this agent.	A text string.
**AgentSiteDescription*	The short description of the agent site. This token is optional.	A text string, up to 254 bytes in length.
**AgentSiteNotes*	The long description of the agent site. This token is optional.	A text string, up to 32700 bytes in length.
**AgentSiteOSType*	The type of operating system that runs on the agent site. This token is required.	One of the following values: **ISV_windowsNT** Windows NT® **ISV_AIX** AIX® **ISV_as400** AS/400® **ISV_Solaris** SUN **ISV_MVS** MVS **ISV_Linux** Linux
**AgentSiteTCP/IPHostname*	The TCP/IP host name of the agent site. This token is required.	A text string, up to 200 bytes in length.

Table 10. AgentSite.tag tokens (continued)

Token	Description	Allowed values
**AgentSiteUserid*	The user ID under which the agent runs. This token is required.	A text string, up to 36 bytes in length.

Table 11. Example values for AgentSite.tag tokens

Token	Example value
**AgentSite*	My agent site
**AgentSiteContact*	DEPT W24A
**AgentSiteDescription*	This is the description of my agent site
**AgentSiteNotes*	These are the notes for my agent site.
**AgentSiteOSType*	ISV_Solaris
**AgentSiteTCP/IPHostname*	CHI11W71.stl.ibm.com
**AgentSiteUserid*	VWADMIN

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

Token Column.tag template for the Data Warehouse Center

The Column.tag template defines a column in a table, or a field in a segment or file. You can use this template to define columns or fields for both sources and targets.

The Column.tag template defines the relationship between the column or field and the table, segment, or file that contains the column or field. You must include this template if you defined sources or targets by using the Table.tag template.

The following tables provide information about each token in the template.

Table 12. Column.tag tokens

Token	Description	Allowed values
	Entity parameters	

Table 12. Column.tag tokens (continued)

Token	Description	Allowed values
**ColumnName*	The name of the column or field. The name must be unique within a table or field. This token is required.	A text string, up to 80 bytes in length.
**ColumnDescription*	The short description of the column or field. This token is optional.	A text string, up to 254 bytes in length.
**ColumnNotes*	The long description of the column or field. This token is optional.	A text string, up to 32700 bytes in length.
**ColumnOffsetFromZero*	The offset in bytes from the start of the file to where the data for this field starts.	A numeric value or 0.
**ColumnOrdinalNumber*	The ordinal position of the column. Usually the same as the **ColumnPositionNumber*.	A numeric value or 0.
**ColumnUserActions*	The actions that a user can perform on this column or field. This token is optional.	A text string, up to 254 bytes in length.
**ColumnLength*	The length of the column or field being created. This token is required.	A numeric value.
**ColumnPrecision*	The precision of the column or field for columns or fields with a decimal data type. This token is required.	A numeric value or 0.
**ColumnKeyPosition*	If this column is part of a key, the column's position within the key. This token is required.	A numeric value. If there is no precision value, specify 0.

Table 12. Column.tag tokens (continued)

Token	Description	Allowed values
**ColumnPositionNumber*	A number, starting with 1, that indicates the order of the column within the row. This token is required.	A numeric value.
**ColumnAllowsNulls*	A flag that specifies whether the column or field allows null data. This token is required.	One of the following values: **ISV_NULLSYES** The column allows null data. **ISV_NULLSNO** The column does not allow null data.
**ColumnDataIsText*	A flag that specifies whether the column or field contains only text data for character types. This token is required.	One of the following values: **ISV_ISTEXTYES** The column contains only text data. **ISV_ISTEXTNO** The column does not contain only text data.
**ColumnEditionType*	Identifies whether the column holds Data Warehouse Center edition information.	One of the following values: **ISV_ColumnIsEditionColumn** The column is an edition column. **ISV_ColumnIsNormal** The column is a normal column.

Table 12. Column.tag tokens (continued)

Token	Description	Allowed values
**ColumnNativeDataType*	The data type of the column or field as defined to the database manager or file system. This token is required.	One of the following values: ISV_NATIVE_CHAR ISV_NATIVE_VARCHAR ISV_NATIVE_LONGVARCHAR ISV_NATIVE_VARCHAR2 ISV_NATIVE_GRAPHIC ISV_NATIVE_VARGRAPHIC ISV_NATIVE_LONGVARGRAPHIC ISV_NATIVE_CLOB ISV_NATIVE_INT ISV_NATIVE_TINYINT ISV_NATIVE_BLOB ISV_NATIVE_SMALLINT ISV_NATIVE_INTEGER ISV_NATIVE_FLOAT ISV_NATIVE_SMALLFLOAT ISV_NATIVE_DOUBLE ISV_NATIVE_REAL ISV_NATIVE_DECIMAL ISV_NATIVE_SMALLMONEY ISV_NATIVE_MONEY ISV_NATIVE_NUMBER

Table 12. Column.tag tokens (continued)

Token	Description	Allowed values
Token	**Description**	**Allowed values**
Relationship parameters		
**DatabaseName*	The business name of the warehouse source or warehouse target. This token is required.	A text string, up to 40 bytes in length.
**TablePhysicalName*	The physical name of the table or file that contains the column as defined to the database manager or file system. This token is required.	A text string, up to 80 bytes in length.
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table that contains the column. This token is required.	A text string, up to 15 bytes in length.

Table 13. Example values for Column.tag tokens

Token	Example value
**ColumnName*	Geography_code
**ColumnDescription*	This column contains the geography code
**ColumnNotes*	The valid values for this column can be found in the Geography reference manual
**ColumnOffsetFromZero*	0
**ColumnOrdinalNumber*	0
**ColumnUserActions*	User cannot directly view a single column
**ColumnLength*	10
**ColumnPrecision*	0
**ColumnKeyPosition*	0
**ColumnAllowsNulls*	ISV_NULLSNO
**ColumnDataIsText*	ISV_ISTEXTYES
**ColumnNativeDataType*	ISV_NATIVE_CHAR
**DatabasePhysicalName*	FINANCE
**TableOwner*	DB2ADMIN
**TablePhysicalName*	GEOGRAPHY

Related reference:

• "Metadata templates supplied with the Data Warehouse Center" on page 29

Token HeaderInfo.tag template for the Data Warehouse Center

This template is always required and must be at the beginning of the tag language file. This template contains control information for the Data Warehouse Center import utility. There are no tokens to be substituted and the template is to be used without modifications.

Related reference:

• "Metadata templates supplied with the Data Warehouse Center" on page 29

Token Process.tag template for the Data Warehouse Center

Use the Process.tag template to define a process to group steps. Each step must be in only one process. This process is related to subject areas, and each partner application must have at least one subject area that any processes resides in. The template defines the relationship between the subject area and the partner application's security group as well as between the process and the subject area.

This template is required if the partner application is defining steps to the Data Warehouse Center.

If you create a new process object, the value that you provide for the **ProcessName* token must be unique to all processes defined in the warehouse control database.

The following tables provide information about and examples for each token in the template.

Table 14. Process.tag tokens. This template contains only relationship parameters.

Token	Description	Allowed values
	Entity parameters	
**ProcessName*	The unique name of the process.	A text string, up to 80 bytes in length.
**ProcessDescription*	The description that is associated with the process.	A text string, up to 254 bytes in length.
**ProcessNotes*	The long description that is associated with the process.	A text string, up to 32,700 bytes in length.
**ProcessContact*	The name of a person or group to contact for questions or concerns about this step.	A text string.
**ProcessType*	The processing options if there was no source data.	One of the following values: **ISV_ProcessType_Normal** Process is a normal user process.
	Relationship parameters	
**SubjectArea*	The name of a subject area that is to contain this process and the steps being created or being added to this process.	A text string, up to 80 bytes in length.

Table 14. Process.tag tokens (continued). This template contains only relationship parameters.

Token	Description	Allowed values
**SecurityGroup*	The security group that is to contain all the objects that you are importing. This token is required, and you must specify the default security group.	ISV_DEFAULTSECURITYGROUP for the default security group.

Table 15. Example values for Process.tag tokens

Token	Example value
**ProcessName*	Marketing process
**ProcessDescription*	A collection of steps that is used by the marketing organization
**ProcessNotes*	Steps that create the star schema that is used by the marketing organization
**ProcessContact*	Marketing
**ProcessType*	ISV_ProcessType_2
**SubjectArea*	Group of processes generated for this partner application
**SecurityGroup*	ISV_DEFAULTSECURITYGROUP

Token StarSchema.tag template for the Data Warehouse Center

You can use the StarSchema.tag template to define a star schema as a way to group related tables. You can use this template to relate tables within the same database (for further use by the DB2 OLAP Integration Server), or to logically group related tables from multiple databases.

The following tables provides information and examples about each token in the template.

Table 16. StarSchema.tag tokens

Token	Description	Allowed values
	Entity parameters	
**StarSchemaName*	The unique name of the star schema that is being created or related.	A text string, up to 80 bytes in length.

Table 16. StarSchema.tag tokens (continued)

Token	Description	Allowed values
**StarSchemaDescription*	A description that is associated with the star schema.	A text string, up to 254 bytes in length.
**StarSchemaNotes*	The long description that is associated with the step.	A text string, up to 32,700 bytes in length.
**StarSchemaContact*	The name of a person or group to contact for questions or concerns about this step.	A text string.
**StarSchemaDBName*	The business name of the database that is being created.	A text string.

Table 17. Example values for StarSchema.tag tokens

Token	Example value
**StarSchemaName*	Marketing schema
**StarSchemaDescription*	This star schema represents the marketing division's internal databases
**StarSchemaNotes*	Tables used for the marketing division
**StarSchemaContact*	Marketing group
**StarSchemaDBName*	Marketing

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

StarSchemaInputTable.tag template for the Data Warehouse Center

You use this template to define the relationship between a star schema and its input source. This relationship is required for all star schemas.

The following tables provide information about and examples for each token in the template.

Table 18. StarSchemaInputTable.tag tokens

Token	Description	Allowed values
	Entity parameters	
**StarSchemaName*	The name of the star schema that is being created or related.	A text string.

Table 18. StarSchemaInputTable.tag tokens (continued)

Token	Description	Allowed values
	Relationship parameters	
**DatabaseName*	The business name of the database that is being created.	A text string.
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table that is being described. This value must be a valid qualifier as defined by the rules of ODBC.	A text string.
**TablePhysicalName*	The physical table name as it is known to ODBC (the system DSN name).	A text string.

The following table provides example values for each token to illustrate the kind of metadata that you might provide for each token.

Table 19. Example values for StarSchemaInputTable.tag tokens

Token	Example value
**StarSchemaName*	Finance schema
**DatabaseName*	Finance Warehouse
**TableOwner*	DB2ADMIN
**TablePhysicalName*	DB2ADMIN.GEOGRAPHY

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

Step.tag template for the Data Warehouse Center

You use the Step.tag template to define a step that will be managed by the Data Warehouse Center. This template includes information about the relationships to security group, process, and agent site.

This template is required for all partner applications that are generating relationships between source and target data or defining programs that the Data Warehouse Center is to run.

If you create a new step object, the value that you provide for the **StepName* token must be unique to all steps that are defined in the warehouse control database.

The following tables provides information about and examples for each token in the template.

Table 20. Step.tag tokens

Token	Description	Allowed values
	Entity parameters	
**StepName*	The name of the step that is being created or related. The name must be unique withing the Data Warehouse Center.	A text string, up to 80 bytes in length.
**StepDescription*	The description that is associated with the step.	A text string, up to 254 bytes in length.
**StepNotes*	The long description that is associated with the step.	A text string, up to 32,700 bytes in length.
**StepDataNotPresent*	The processing options if there was no source data.	One of the following values: **ISV_StepDataNotPresent_OK** If data is not present, continue processing. **ISV_StepDataNotPresent_Warning** If data is not present, issue a warning and continue processing. **ISV_StepDataNotPresent_Error** If data is not present, issue an error message and stop processing.
**StepSelectStatement*	The SQL statement to be issued if ISV_StepSelectStatementNo.	A SQL string.
**StepContact*	The name of a person or group to contact for questions or concerns about this step.	A text string.
**StepExternalPopulation*	A flag that indicates that the step is expected to be run outside the Data Warehouse Center environment..	One of the following values: **ISV_StepExternalNo** The table will not be externally populated by other means. **ISV_StepExternalYes** The table will be externally populated by other means.

Table 20. Step.tag tokens (continued)

Token	Description	Allowed values
**StepType*	The type of step that is being created.	One of the following values: **ISV_StepType_Editioned_Append** The data in the table will be appended when the Step is run. **ISV_StepType_Full_Replace** The data in the table will be replaced when the Step is run. **ISV_StepType_Uneditioned_Append** The data in the table will be appended when the Step is run. **ISV_StepType_VWP_Population** The data in the table is populated by a Data Warehouse Center program.
**StepSQLWarning*	The processing options if an SQL warning occurs.	One of the following values: **ISV_StepSQLWarning_OK** If an SQL warning occurs, continue processing. **ISV_StepSQLWarning_Warning** If an SQL warning occurs, issue a warning and continue processing. **ISV_StepSQLWarning_Error** If an SQL warning occurs, issue an error and stop processing.
**StepCommit*	A flag that specifies if the Data Warehouse Center is to intermittently commit after **StepCommitAfterNumberRows* is inserted into the target table of the step.	One of the following values: **ISV_Step_Incremental_Commit_On** The data is to be incrementally committed at the target. **ISV_Step_Incremental_Commit_Off** The data is not to be incrementally committed at the target.
**StepCommitAfterNumberRows*	The number of rows to insert before committing.	A numeric value.
Relationship parameters		

Table 20. Step.tag tokens (continued)

Token	Description	Allowed values
**SecurityGroup*	The security group that is to contain all the objects that you are importing. This token is required, and you must specify the default security group.	ISV_DEFAULTSECURITYGROUP for the default security group.
**ProcessName*	The name of the process. This token is required.	A text string, up to 80 bytes in length.
**AgentSite*	The name of a new agent site, or the name of the default agent site, if the agent is not new. If you specify a new name, it must be unique within the Data Warehouse Center control database. This token is required, but you can specify the default agent site, ISV_DEFAULTAGENTSITE	A text string, up to 80 bytes in length. If you do not want to create a new agent site, use ISV_DEFAULTAGENTSITE for the default agent site.

Table 21. Example values for Step.tag tokens

Token	Example value
**StepName*	Revenue by location
**StepDescription*	This step will pull data to create the revenue for each location in a DB2 table
**StepNotes*	Revenue for Geography 7 comes from 4 source Oracle tables
**StepDataNotPresent*	ISV_StepDataNotPresent_Error
**StepSelectStatement*	SELECT * FROM IWH.REVENUE_BY_LOCATION
**StepContact*	Jason Smythe
**StepExternalPopulation*	ISV_StepExternalNo
**StepType*	ISV_StepType_Full_Replace
**StepSQLWarning*	ISV_StepSQLWarning_Warning
**StepCommit*	ISV_Step_Incremental_Commit_On
**StepCommitAfterNumberRows*	10000
**SecurityGroup*	ISV_DEFAULTSECURITYGROUP

Table 21. Example values for Step.tag tokens (continued)

Token	Example value
**ProcessName*	Marketing process
**AgentSite*	My agent site

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

StepCascade.tag template for the Data Warehouse Center

You use the StepCascade.tag template to define a relationship between two steps to specify that another step is to be started at the completion of the named step.

This template is required only if the partner application links steps in a cascaded relationship.

The following tables provide information about and examples for each StepCascade.tag token in a template.

Table 22. StepCascade.tag tokens

Token	Description	Allowed values
	Entity parameters	
**StepName*	The name of the step that is being related.	A text string.
**PostStepName*	The name of the step that is to be run after the completion of another step.	A text string.

Table 23. Example values for StepCascade.tag tokens

Token	Example value
**StepName*	Revenue by location
**PostStepName*	Revenue for all Geographies

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

StepInputTable.tag template for the Data Warehouse Center

You use this template to define the relationship between a star schema and its input source. This relationship is required for all star schemas.

The following tables provide information about and examples for each token in the template.

Table 24. StepInputTable.tag tokens

Token	Description	Allowed values
	Entity parameters	
**StarSchemaName*	The name of the star schema that is being created or related.	A text string.
	Relationship parameters	
**DatabaseName*	The business name of the database that is being created.	A text string.
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table that is being described. This value must be a valid qualifier as defined by the rules of ODBC.	A text string.
**TablePhysicalName*	The physical table name as it is known to ODBC (the system DSN name).	A text string.

The following table provides example values for each token to illustrate the kind of metadata that you might provide for each token.

Table 25. Example values for StepInputTable.tag tokens

Token	Example value
**StarSchemaName*	Finance schema
**DatabaseName*	Finance Warehouse
**TableOwner*	DB2ADMIN
**TablePhysicalName*	DB2ADMIN.GEOGRAPHY

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

StepOutputTable.tag template for the Data Warehouse Center

You use the StepOutputTable.tag template to define the relationship between a step and its output target.

This relationship is required for steps of type ISV_StepType_Editioned_Append, ISV_StepType_Full_Replace, ISV_StepType_Uneditioned_Append.

The following tables provide information about and examples for each token in the template.

Table 26. StepOutputTable.tag tokens

Token	Description	Allowed values
	Entity parameters	
**StepName*	The name of the step that is being created or related.	A text string.
	Relationship parameters	
**DatabaseName*	The business name of the database that is being related.	A text string.
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table being described. This value must be a valid qualifier as defined by the rules of ODBC.	A text string.
**TablePhysicalName*	The physical table name as it is known to ODBC (the system DSN name).	A text string.
**ProcessName*	The name of the process that is being related.	A text string.

Table 27. Example values for StepOutputTable.tag tokens

Token	Example value
**StepName*	Revenue by product
**DatabaseName*	Finance Warehouse
**TableOwner*	FINADMIN
**TablePhysicalName*	INVENTORY
**ProcessName*	Marketing process

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

StepVWPOutputTable.tag template for the Data Warehouse Center

Use this template to optionally define the relationship between a step of type ISV_StepType_VWP_Population and its output targets.

The following tables provide information about and examples for each token in the template.

Table 28. StepVWPOutputTable.tag tokens

Token	Description	Allowed values
	Entity parameters	
**StepName*	The name of the step that is being related.	A text string.
	Relationship parameters	
**DatabaseName*	The business name of the database that is being created.	A text string.
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table that is being described. This value must be a valid qualifier as defined by the rules of ODBC.	A text string.
**TablePhysicalName*	The physical table name as it is known to ODBC (the system DSN name).	A text string.
**ProcessName*	The name of the process that is being created or related	A text string.

Table 29. Example values for StepVWPOutputTable.tag tokens

Token	Example value
**StepName*	Revenue by product
**DatabaseName*	Finance Warehouse
**TableOwner*	FINADMIN
**TablePhysicalName*	INVENTORY
**ProcessName*	Marketing process

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

StepVWPProgramInstance.tag

Use this template to define an instance of a Data Warehouse Center program that is run by a warehouse agent. This template also defines the relationship to the Data Warehouse Center program definition, called the VWPTemplate, as well as the step that uses the Data Warehouse Center program. This template is required for each step that utilizes the Data Warehouse Center program.

The following tables provide information about and examples for each token in the template.

Table 30. StepVWPProgramInstance.tag tokens

Token	Description	Allowed values
	Entity parameters	
**VWPProgramInstanceKey*	Key that uniquely identifies this program instance. The key must be unique from all other keys in the tag language file. **Tip**: Finish processing the VWPProgramInstance.tag template before increasing the value of the key. This token is required.	A numeric value.
	Relationship parameters	
**StepName*	The name of the step that is being related.	A text string.
**VWPProgramTemplateName*	The business name of the Data Warehouse Center program template that is being created.	A text string.

Table 31. Example values for StepVWPProgramInstance.tag tokens

Token	Example value
**VWPProgramInstanceKey*	070001
**StepName*	Revenue by location
**VWPProgramTemplateName*	My ISV Program

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

AgenttoDatabase.tag template for the Data Warehouse Center

The AgenttoDatabase.tag template associates an agent site to an existing source or target database. This template is required if the agent site that is defined inthe tag language file refers to a source or target database that exists in the Data Warehouse Center control database.

Table 32. AgenttoDatabase.tag tokens

Token	Description	Allowed values
	Relationship parameters	
**DatabaseName*	The database name. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**DatabasePhysicalName*	The physical database name that is defined to the database manager and known to ODBC. This token is required.	A text string, up to 40 bytes in length.
**AgentSite*	The agent site name to use for the source or target. This token is required.	A text string, up to 80 bytes in length. Specify `ISV_DEFAULTAGENTSITE` to use the default agent site.

Table 33. Example values for AgenttoDatabase.tag tokens

Token	Example value
**DatabaseName*	Finance Warehouse
**DatabasePhysicalName*	Finance
*AgentSite	My agent site name

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

AgenttoProgram.tag template for the Data Warehouse Center

Use the AgenttoProgram.tag template to associate an agent site to an existing Data Warehouse Center program. The template is required if the agent site that is defined in the tag language file refers to a Data Warehouse Center program that exists in the Data Warehouse Center control database.

Table 34. AgenttoProgram.tag tokens

Token	Description	Allowed values
	Relationship parameters	
**VWPProgramTemplateName*	The name of the Data Warehouse Center program template. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**AgentSite*	The name of the agent site to use for the source or target. This token is required.	A text string, up to 80 bytes in length. Specify `ISV_DEFAULTAGENTSITE` for the default agent site.

Table 35. Example values for AgenttoProgram.tag tokens

Token	Example value
**VWPProgramTemplateName*	My ISV program name
**AgentSite*	My agent site name

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

Commit.tag template for the Data Warehouse Center

Use this template to improve performance when you are using large tag language files. You can insert a commit template between any of the groups of templates described here. You cannot insert a commit template between templates within the following groups:

- AgenttoDatabase.tag, AgenttoProgram.tag
- AgentSite.tag
- VWPGroup.tag
- VWPProgramTemplate.tag, VWPProgramTemplateParameter.tag
- SourceDatabase.tag

- WarehouseDatabase.tag
- Table.tag, Column.tag
- SubjectArea.tag
- Process.tag
- Step.tag, StepInputTable.tag, StepOutputTable.tag, StepVWPOutputTable.tag, StepVWPProgramInstance.tag,VWPProgramInstanceParameter.tag
- StepCascade.tag
- StarSchema.tag, StarSchemaInputTable.tag
- PrimaryKey.tag, PrimaryKeyAdditional.tag
- ForeignKey.tag, ForeignKeyAdditional.tag

For example, it is valid to insert a commit template between AgentSite.tag and VWPGroup.tag but invalid to insert a commit tag between VWPProgramTemplate.tag and VWPProgramTemplateParameter.tag. If commit templates are used incorrectly, import may report an error.

The use of the commit template is optional.

Table 36. Commit.tag tokens

Token	Description	Allowed values
	Relationship parameters	
**CurrentCheckPointID++*	An index, starting with 0, that increases each time it is substituted in a token. This token is required.	A numeric value.

Table 37. Example values for Commit.tag tokens

Token	Example value
**CurrentCheckPointID++*	1

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

ForeignKey.tag template for the Data Warehouse Center

Use this template to define foreign key constraints on tables. The ForeignKey.tag template defines the relationships to the table and the column on which the constraint is being defined. This template also defines the relationships to the table and column of the primary key that is being referred to. Before you use the ForeignKey.tag template, you must define the primary

key constraint (using the PrimaryKey.tag template) and the tables and columns (using the Table.tag and Column.tag templates) on which you want to define the foreign key constraint.

Table 38. ForeignKey.tag tokens

Token	Description	Allowed values
	Entity parameters	
**ConstraintName*	The name of the constraint. The name must be unique within a table or field. This token is required.	A text string, up to 80 bytes in length.
**ForeignColumnKeyName*	The name of the column on which the foreign key constraint is being defined.	A text string, up to 254 bytes in length.
**ForeignKeyID*	The key that uniquely identifies the foreign key. The key must be unique from all other keys in the tag language file. **Tip:** Finish processing the ForeignKey.tag template before increasing the value of thekey. This token is required.	A numeric value.
**MapID*	An arbitrary number that is unique from all other keys in the interchange file. **Tip:** Finish processing the ForeignKey.tag template before increasing the value of this token. This token is required.	A numeric value.
**PrimaryColumnKeyName*	The column name of the referenced column.	A text string, up to 80 bytes in length.

Table 38. ForeignKey.tag tokens (continued)

Token	Description	Allowed values
Token	**Description**	**Allowed values**
**ReferencedPrimaryKeyID*	The key that uniquely identifies the primary key. The key must be unique from all other keys in the tag language file. **Tip:** Finish processing the ForeignKey.tag template before increasing the value of the key. This token is required.	A numeric value.
	Relationship parameters	
**DatabaseName*	The business name of the warehouse source or warehouse target. This token is required.	A text string, up to 40 bytes in length.
ForeignTablePhysicalName	The database-defined name of the physical table containing the foreign keys that reference the keys in other tables.	A text string, up to 254 bytes in length.
**PrimaryTablePhysicalName*	The database-defined name of the physical table containing the keys that are referenced by the foreign keys.	A text string, up to 80 bytes in length.
**PrimaryTableOwner*	The owner, high-level qualifier, collection, or schema of the table that contains the primary key column that is being referenced. This token is required.	A text string, up to 128 bytes in length.
**ForeignTableOwner*	The owner, high-level qualifier, collection, or schema of the table that contains the foreign key constraint column. This token is required.	A text string, up to 128 bytes in length.

Table 39. Example values for ForeignKey.tag tokens

Token	Example value
**ConstraintName*	Department
**DatabaseName*	Finance Warehouse
**ForeignColumnKeyName*	Geography_code
**ForeignKeyID*	07011
**ForeignTablePhysicalName*	GEOGRAPHY
**MapID*	02568
**PrimaryColumnKeyName*	State_code
**ReferencedPrimaryKeyID*	Name
**PrimaryTablePhysicalName*	City
**PrimaryTableOwner*	DB2ADMIN
**ForeignTableOwner*	IWH

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29
- "ForeignKeyAdditional.tag template for the Data Warehouse Center" on page 56

ForeignKeyAdditional.tag template for the Data Warehouse Center

Use this template to define a composite foreign key. Before you use the ForeignKeyAdditional.tag template, you must define a constraint (using the ForeignKey.tag template) on the first column. You can then add columns by using this template for each column that you want to add.

Table 40. ForeignKeyAdditional.tag tokens

Token	Description	Allowed values
	Entity parameters	
**ForeignColumnKeyName*	The name of the column which the foreign key constraint is being defined. This token is required.	A text string, up to 80 bytes in length.

Table 40. ForeignKeyAdditional.tag tokens (continued)

Token	Description	Allowed values
Token	**Description**	**Allowed values**
**ForeignKeyID*	The key that uniquely identifies the foreign key. The key must be unique from all other keys in the tag language file. **Tip:** Finishe processing the ForeignKeyAdditional.tag template before increasing the value of the key. This token is required.	A numeric value.
**MapID*	An arbitrary number that is unique from all other keys in the interchange file. **Tip:** Finish processing the ForeignKeyAdditional.tag tempalte before increasing the value of this token. This token is required.	A numeric value.
**MapSeqNo*	A number signifying each additional column added as part of a composite key to the foreign key constraint.	A unique, increasing, consecutive number starting at 2.
**PrimaryColumnKeyName*	The column name of the referenced column.	A text string, up to 80 bytes in length.
	Relationship parameters	
**DatabaseName*	The business name of the warehouse source or warehouse target. This token is required.	A text string, up to 40 bytes in length.
**ForeignTablePhysicalName*	The database-defined name of the physical table containing the keys that are referenced by the keys in other tables.	A text string, up to 80 bytes in length.
**PrimaryTablePhysicalName*	The database-defined name of the physical table containing the keys that are referenced by the foreign keys.	A text string, up to 80 bytes in length.

Table 40. ForeignKeyAdditional.tag tokens (continued)

Token	Description	Allowed values
**PrimaryTableOwner*	The owner, high-level qualifier, collection, or schema of the table that contains the primary key column that is being referenced. This token is required.	A text string, up to 128 bytes in length.
**ForeignTableOwner*	The owner, high-level qualifier, collection, or schema of the table that contains the foreign key constraint name. This token is required.	A text string, up to 128 bytes in length.

Table 41. Example values for ForeignKeyAdditional.tag tokens

Token	Example value
**DatabaseName*	Finance Warehouse
**ForeignColumnKeyName*	Geography_code
**ForeignKeyID*	07011
**ForeignTablePhysicalName*	GEOGRAPHY
**MapID*	22578
**MapSeqNo*	2
**PrimaryColumnKeyName*	State_code
**PrimaryTablePhysicalName*	City
**PrimaryTableOwner*	DB2ADMIN
**ForeignTableOwner*	IWH

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29
- "ForeignKey.tag template for the Data Warehouse Center" on page 53

PrimaryKey.tag template for the Data Warehouse Center

Use this template to define primary key constraints on tables. The template also defines the relationships to the table and the column on which the constraint is being defined. Before you use the PrimaryKey.tag template, you must define the tables and columns (using the Table.tag and Column.tag templates) on which you want to define the primary key constraint.

Table 42. PrimaryKey.tag tokens

Token	Description	Allowed values
	Entity parameters	
**ColumnName*	The name of the column or field. The name must be unique within a table or field. This token is required.	A text string, up to 80 bytes in length.
**MapID*	An arbitrary number that is unique from all other keys in the interchange file. **Tip:** Finish processing the PrimaryKey.tag template before increasing the value of this token. This token is required.	A numeric value.
**PrimaryKeyID*	The key that uniquely identifies the primary key. The key must be unique from all other keys in the tag language file. **Tip:** Finish processing the PrimaryKey.tag template before increasing the value of the key. This token is required.	A numeric value.
	Relationship parameters	
**DatabaseName*	The business name of the warehouse source or warehouse target. This token is required.	A text string, up to 40 bytes in length.

Table 42. PrimaryKey.tag tokens (continued)

Token	Description	Allowed values
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table that contains the column. This token is required.	A text string, up to 128 bytes in length.
**TablePhysicalName*	The physical name of the table or file that contains the column as defined to the databae manager or file system. This token is required.	A text string, up to 80 bytes in length.

Table 43. Example values for PrimaryKey.tag tokens

Token	Example value
**ColumnName*	Geography_code
**DatabaseName*	Finance Warehouse
**MapID*	54627
**PrimaryKeyID*	74622
**TableOwner*	DB2ADMIN
**TablePhysicalName*	Geography

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29
- "PrimaryKeyAdditional.tag template for the Data Warehouse Center" on page 60

PrimaryKeyAdditional.tag template for the Data Warehouse Center

Use this template to define a composite primary key. Before you use the PrimaryKeyAdditional.tag template, you must define a constraint on the first column by using the PrimaryKey.tag template. Any additional columns can then be added using this template. The template also relates the additional primary keys to the first primary key which is defined using PrimaryKey.tag.

Table 44. PrimaryKeyAdditional.tag tokens

Token	Description	Allowed values
	Entity parameters	

Table 44. PrimaryKeyAdditional.tag tokens (continued)

Token	Description	Allowed values
Token	**Description**	**Allowed values**
Relationship parameters		
**ColumnName*	The name of the column or field. The name must be unique within a table or field. This token is required.	A text string, up to 80 bytes in length.
**FirstPrimaryKeyID*	The key that uniquely identifies the primary key. The key must be unique from all other keys inthe tag language file. **Tip:** Finish processing the PrimaryKeyAdditional.tag template before increasing the value of the key. This token is required.	A numeric value.
**MapID*	An arbitrary number that is unique from all other keys int he interchange file. **Tip:** Finish processing the PrimaryKeyAdditional.tag template before increasing the value of this token. This token is required.	A numeric value.
**MapSeqNo*	A number signifying each additional column added as part of a composite key to the primary key constraint. This token is required.	A unique, increasing, consecutive number starting at 2.
**DatabaseName*	The business name of the warehouse source or warehouse target. This token is required.	A text string, up to 40 bytes in length.

Table 44. PrimaryKeyAdditional.tag tokens (continued)

Token	Description	Allowed values
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table that contains the column. This token is required.	A text string, up to 15 bytes in length.
**TablePhysicalName*	The physical name of the table or file that contains the column as defined to the database manager or file system. This token is required.	A text string, up to 80 bytes in length.

Table 45. Example values for PrimaryKeyAdditional.tag tokens

Token	Example value
**ColumnName*	Geography_code
**DatabaseName*	Finance Warehouse
**MapID*	99542
**MapSeqNo*	2
**FirstPrimaryKeyID*	07801
**TableOwner*	DB2ADMIN
**TablePhysicalName*	GEOGRAPHY

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29
- "PrimaryKey.tag template for the Data Warehouse Center" on page 59

SourceDataBase.tag template for the Data Warehouse Center

Use the SourceDataBase.tag template to define source databases, file systems, or files to import into the Data Warehouse Center. You can use this template to define a relational non-DB2 source database as well as a DB2 source database.

This template also defines the relationship between the following objects:

- The source databases
- The agent site to use for the source database
- The security group in which to define the source database

The following tables provide information about each token in the template.

Table 46. SourceDataBase.tag tokens

Token	Description	Allowed values
	Entity parameters	
**DatabaseName*	The name of the database. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**DatabaseDescription*	The short description of the database. This token is optional.	A text string, up to 254 bytes in length.
**DatabaseNotes*	The long description of the database. This token is optional.	A text string, up to 32700 bytes in length.
**DatabaseContact*	The person to contact for information about this database. This token is optional.	A text string, up to 64 bytes in length.
**DatabaseServerName*	The name of the server on which the database resides. This token is required for Flat File LAN files. Otherwise, it is optional.	A text string, up to 64 bytes in length.
**DatabaseVersion*	The version of the database.	A text string.
**DatabasePhysicalName*	The physical database name of the database as defined to the database manager, as known to ODBC. This token is required.	A text string, up to 40 bytes in length.

Table 46. SourceDataBase.tag tokens (continued)

Token	Description	Allowed values
**DatabaseType*	The type of database family. This token is required.	One of the following values: **ISV_IR_DB2Family** DB2 Family **ISV_IR_Oracle** Oracle **ISV_IR_Sybase** Sybase **ISV_IR_MSSQLServer** Microsoft® SQL Server **ISV_IR_Informix** Informix **ISV_IR_GenericODBC** Generic ODBC **ISV_IR_FFLan** Flat File LAN **ISV_IR_VSAM** VSAM **ISV_IR_IMS** IMS
**DatabaseTypeExtended*	The type of AS/400 system or file. This token is required.	One of the following values: **ISV_IR_DB2400CISC** DB2 UDB for AS/400 for CISC **ISV_IR_DB2400RISC** DB2 UDB for AS/400 for RISC **ISV_IR_FFLanLocalCmd** Local flat file **ISV_IR_FFLanFTPCopy** Local flat file sent using FTP from a remote system
**DatabaseUserid*	The user ID with which to access the database. This token is optional.	A text string, up to 36 bytes in length.
Relationship parameters		

Table 46. SourceDataBase.tag tokens (continued)

Token	Description	Allowed values
**SecurityGroup*	The security group in which to create the source or target database. This token is required, and you must specify the default security group.	ISV_DEFAULTSECURITYGROUP for the default security group.
**AgentSite*	The agent site to use for the source or target database. This token is required, but you can specify the default agent site.	A text string, up to 80 bytes in length. ISV_DEFAULTAGENTSITE for the default agent site.

Table 47. Example values for SourceDataBase.tag tokens

Token	Example value
**DatabaseName*	Finance Warehouse
**DatabaseDescription*	This database contains financial information.
**DatabaseNotes*	This is the warehouse where all geographies keep financial information.
**DatabaseContact*	Valerie Zieman
**DatabaseServerName*	CHI11W71
**DatabaseVersion*	V6.1.0
**DatabasePhysicalName*	FINANCE
**DatabaseType*	ISV_IR_DB2Family
**DatabaseTypeExtended*	ISV_DEFAULTVALUE
**DatabaseUserid*	DB2ADMIN
**SecurityGroup*	ISV_DEFAULTSECURITYGROUP
**AgentSite*	My agent site

Related reference:

• "Metadata templates supplied with the Data Warehouse Center" on page 29

SubjectArea.tag tokens for the Data Warehouse Center

Use this template to define a subject area to contain the processes and steps that you create. Each tag language file must have at least one subject area to contain any processes and steps that you create. This template is required if you are defining processes and steps.

This template also defines the relationship between the subject area and the security group that the header file specifies.

The following tables provide information about and examples for each token in the template.

Table 48. SubjectArea.tag tokens

Token	Description	Allowed values
	Entity parameters	
**SubjectArea*	The name of a group that is to contain all of the processes and steps that are created or added to a particular subject area. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**SubjectAreaContact*	The name of the person or organization that is responsible for this subject area.	A text string.
**SubjectAreaDescription*	A short description of the group of processes and steps. This token is optional.	A text string, up to 254 bytes in length.
**SubjectAreaNotes*	A long description of the group of processes and steps. This token is optional.	A text string, up to 32700 bytes in length.
	Relationship parameters	

Table 48. SubjectArea.tag tokens (continued)

Token	Description	Allowed values
**SecurityGroup*	The security group in which to create the subject area. This token is required, and you must specify the default security group.	ISV_DEFAULTSECURITYGROUP for the default security group.
**CurrentCheckPointID++*	An index, starting with 0, that increases each time that it is substituted in a token. This token is required.	A numeric value.

Table 49. Example values for SubjectArea.tag tokens

Token	Example value
**SubjectArea*	Group of processes and steps generated for the partner tool
**SubjectAreaContact*	DEPT W24A
**SubjectAreaDescription*	This subject area contains all the processes and steps generated for Data Warehouse Center by the partner tool.
**SubjectAreaNotes*	The processes and steps in this subject area will be used to evaluate the product.
**SecurityGroup*	ISV_DEFAULTSECURITYGROUP
**CurrentCheckPointID++*	9

Table.tag template for the Data Warehouse Center

You can use this template to define both source and target tables as well as source files and segments that Data Warehouse Center is to access. You can use this template to define source and target tables, files, and segments.

The template defines all the metadata that the Data Warehouse Center requires to define a table in an ODBC data source as well as a DB2 target table. The template also defines the relationships between the table and the database that contains the table.

The following tables provide information about and examples for each token in the template.

Table 50. Table.tag tokens

Token	Description	Allowed values
	Entity parameters	
**TableFullName*	The fully qualified name of a relational table or a file. For a table, this name is the concatenation of the value of the **TableOwner* and **TablePhysicalName* tokens, separated by a period. For a file, the **TableOwner* value should be left blank, and the **TableFullName* and **TablePhysicalName* values should be the same. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**TableDescription*	The short description of the table. This token is optional.	A text string, up to 254 bytes in length.
**TableNotes*	The long description of the table. This token is optional.	A text string, up to 32700 bytes in length.
**TableOwner*	The owner, high-level qualifier, collection, or schema of the table. This token is required, except for files and IMS databases, which should not specify an owner.	A text string, up to 15 bytes in length.

Table 50. Table.tag tokens (continued)

Token	Description	Allowed values
**TablePhysicalName*	The physical table name as defined to the database manager or file system. If the name has mixed case or spaces, you must place double quotes around the table name (for example, *"MYTABLE"*). This token is required.	A text string, up to 80 bytes in length.
**TableBinaryIfFile*	A flag that specifies whether the file contains only binary data if the table represents a file. This token is optional.	One of the following values: **ISV_DR_FILE_IS_BINARY** The file is binary. **ISV_DR_FILE_IS_NOT_BINARY** The file is in ASCII or mixed format.
**TableFirstRowNamesIfFile*	A flag that specifies whether the first row of the file contains column names if the table represents a file. This token is optional.	One of the following values: **ISV_DR_ROW_CONTAINS_NAMES** The first row of the file contains column names. **ISV_DR_ROW_DOES_NOT_CONTAIN_NAMES** The first row of the file contains data.
**TableTypeIfFile*	The type of file if the table represents a file. This token is optional.	One of the following values: **ISV_DR_REL_TABLE** The table is a relational table. **ISV_DR_COMMA_DELIMITED** The columns in the file are separated by commas. **ISV_DR_FIXED_FORMAT** The columns in the file are in fixed format. **ISV_DR_TAB_DELIMITED** The columns in the file are separated by tabs. **ISV_DR_CHAR_DELIMITED** The columns in the file are separated by the value of **TableDelimiterIfFile*.

Table 50. Table.tag tokens (continued)

Token	Description	Allowed values
**TableDelimiterIfFile*	The value of the delimiter to separate fields if the file type is ISV_DR_CHAR_DELIMITED. This token is optional.	A text string, 1 byte in length.
**TableIsAView*	A token that specifies whether the table is a view.	One of the following values: **ISV_TableIsAView** The table is a view. **ISV_TableIsNotAView** The table is not a view.
**TableIsADimensionTable*	A token that specifies whether the table is a part of a star schema and contains dimensional data.	One of the following values: **ISV_TableIsADimensionalTable** The table is a dimensional table. **ISV_TableIsNotADimensionalTable** The table is not a dimensional table.
**TableIsAnAlias*	A token that specifies whether the table is actually an alias of another table.	One of the following values: **ISV_TableIsAnAlias** This table is an alias for another table. **ISV_TableIsNotAnAlias** This table is not an alias for another table.
**TableCreatedByDWC*	A token that specifies whether the Data Warehouse Center should create and manage this table.	One of the following values: **ISV_TableIsToBeCreatedByDWC** The table is to be created by the Data Warehouse Center. **ISV_TableIsNotToBeCreatedByDWC** The table is not to be created by the Data Warehouse Center.
**TableGrantedToPublic*	A token that specifies whether the Data Warehouse Center should grant public access to this table when the table is created. This is only valid if the Data Warehouse Center creates the table.	One of the following values: **ISV_GrantTableAccessToPublic** The Data Warehouse Center is to grant PUBLIC access to this table. **ISV_DoNotGrantTableAccessToPublic** The Data Warehouse Center is not to grant PUBLIC access to this table.

Table 50. Table.tag tokens (continued)

Token	Description	Allowed values
***TableIsPersistent*	A token that specifies whether the data in the table is to persist between executions of the steps that use this table. If the table is not persistent, the data in the table will be deleted after each use.	One of the following values: **ISV_TableIsPersistent** The table is to be considered persistent. **ISV_TableIsTransient** The table is to be considered transient.
**TableMaximumEditions*	The maximum number of editions the table is to have, if the table supports editions.	A numeric value.
**TableGenerateCreateStatement*	A token that specifies whether the Data Warehouse Center is to generate the create table statement.	One of the following values: **ISV_GenerateCreateTableStmt** The Data Warehouse Center should generate the CREATE TABLE statement. **ISV_DoNotGenerateCreateTableStmt** The Data Warehouse Center should not generate the CREATE TABLE statement.
**TableIsAFactTable*	A token that specifies whether the table is part of a star schema, and the table contains the fact information.	One of the following values: **ISV_TableIsAFactTable** The table is a fact table. **ISV_TableIsNotAFactTable** The table is not a fact table.
**TableCreateStatement*	The DDL to create the table. Use this token only if the ISV_DoNotGenerateCreateTableStmt has been specified.	A text string.
Relationship parameters		
**DatabaseName*	The name of the database that contains the table. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.

Table 50. Table.tag tokens (continued)

Token	Description	Allowed values
**DatabasePhysicalName*	The physical database name of the database that contains the table. This token is required.	A text string, up to 40 bytes in length.

Table 51. Example values for Table.tag tokens

Token	Example value
**TableFullName*	DB2ADMIN.GEOGRAPHY
**TableDescription*	Contains geography information
**TableNotes*	This table contains all the information about geographies serviced by our company
**TableOwner*	DB2ADMIN
**TablePhysicalName*	GEOGRAPHY
**TableBinaryIfFile*	ISV_DEFAULTVALUE
**TableFirstRowNamesIfFile*	ISV_DEFAULTVALUE
**TableTypeIfFile*	ISV_DEFAULTVALUE
**TableDelimiterIfFile*	ISV_DEFAULTVALUE
**TableIsAView*	ISV_TableIsAView
**TableIsADimensionTable*	ISV_TableIsNotADimensionTable
**TableIsAnAlias*	ISV_TableIsAnAlias
**TableCreatedByDWC*	ISV_TableIsToBeCreatedByDWC
**TableGrantedToPublic*	ISV_GrantTableAccessToPublic
**TableIsPersistent*	ISV_TableIsTransient
**TableMaximumEditions*	12
**TableGenerateCreateStatement*	ISV_GenerateCreateTableStmt
**TableIsAFactTable*	ISV_TableIsAFactTable
**TableCreateStatement*	Create table xyz
**DatabaseName*	Finance warehouse
**DatabasePhysicalName*	FINANCE

Related reference:

• "Metadata templates supplied with the Data Warehouse Center" on page 29

VWPGroup.tag template for the Data Warehouse Center

Use this template to define a group that is to contain any Data Warehouse Center programs that you are defining. This template is required if you are defining Data Warehouse Center programs.

The following tables provide information about and examples for each token in the template.

Table 52. VWPGroup.tag tokens

Token	Description	Allowed values
	Entity parameters	
**VWPGroup*	The unique name of a program group that is to contain all of the Data Warehouse Center programs being created. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**VWPGroupDescription*	The short description of the group of Data Warehouse Center programs. This token is optional.	A text string, up to 254 bytes in length.
**VWPGroupNotes*	The long description of the group of Data Warehouse Center programs. This token is optional.	A text string, up to 32700 bytes in length.

Table 53. Example values for VWPGroup.tag tokens

Token	Example value
**VWPGroup*	Group of programs for the partner tool
**VWPGroupDescription*	This group contains all the programs used by Data Warehouse Center for the partner tool
**VWPGroupNotes*	These programs can be used to determine the relationship between sales and location.

Related reference:

• "Metadata templates supplied with the Data Warehouse Center" on page 29

VWPProgramInstanceParameter.tag template for the Data Warehouse Center

Use this template to add or change a parameter that the Data Warehouse Center passes to an instance of a Data Warehouse Center program for a specific step. For example, you set a default value for a host name parameter in the VWPProgramTemplateParameter.tag file. You use this template to change the value that is passed to the Data Warehouse Center program when this particular step runs.

This template is required if the Data Warehouse Center program requires the Data Warehouse Center to pass parameters to it. You can specify that the Data Warehouse Center pass multiple parameters to the program by including this template for each parameter.

This template also defines the relationship between the parameter and its program instance.

The following tables provide information about and examples for each token in the template.

Table 54. VWPProgramInstanceParameter.tag tokens

Token	Description	Allowed values
	Entity parameters	
**VWPProgramInstanceParameterName*	The unique name or description of a parameter that is to be passed to a Data Warehouse Center program. This token is required.	A text string, up to 80 bytes in length.
**VWPProgramInstanceParameterOrder*	A number, starting with 0, that indicates the order of the parameter in the parameter list. This token is required.	A numeric value.
**VWPProgramInstanceParameterData*	The data that is passed to the Data Warehouse Center program as the value of the parameter. This token is required.	A text string or a numeric value up to 240 bytes in length.

Table 54. VWPProgramInstanceParameter.tag tokens (continued)

Token	Description	Allowed values
**VWPProgramInstanceParameterKey*	A key that uniquely identifies this program parameter instance. The key must be unique from all other parameter keys in the interchange file. **Tip:** Finish processing the VWPProgramInstanceParameter.tag template before increasing the value of the key. This token is required.	A text value, up to 10 bytes in length.
**VWPProgramInstanceParameterType*	The type of value that this parameter contains. For example, character, numeric, or password data.	One of the following values: **ISV_ParameterTypeNone** The parameter type is unknown or not applicable. **ISV_ParameterTypeCharacter** The parameter type is character. **ISV_ParameterTypeNumeric** The parameter type is numeric. **ISV_ParameterTypePassword** The parameter type is password.
Relationship parameters		
**VWPProgramInstanceKey*	A key that uniquely identifies this program instance. The key must be unique from all other keys in the interchange file. **Tip:** Finish processing the VWPProgramInstance.tag template before increasing the value of the key. This token is required.	A text value, up to 10 bytes in length

Table 55. Example values for VWPProgramInstanceParameter.tag tokens

Token	Example value
**VWPProgramInstanceParameterName*	DB2 UDB user ID

Table 55. Example values for VWPProgramInstanceParameter.tag tokens (continued)

Token	Example value
**VWPProgramInstanceKey*	070000
**VWPProgramInstanceParameterOrder++*	1
**VWPProgramInstanceParameterData*	my_userid
**VWPProgramInstanceParameterKey*	012994
**VWPProgramInstanceParameterType*	ISV_ParameterTypeNumeric
**VWPProgramInstanceKey*	070001

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29

VWPProgramTemplate.tag template for the Data Warehouse Center

Use this template to define a Data Warehouse Center program. This template is required if the tag language file refers to a Data Warehouse Center program, unless the warehouse program already exists in the Data Warehouse Center control database.

The template also defines the relationship between the warehouse program definition and the Data Warehouse Center program group to which the program belongs.

The following tables provide information about and examples for each token in the template.

Table 56. VWPProgramTemplate.tag tokens

Token	Description	Allowed values
	Entity parameters	
**VWPProgramTemplateName*	The name of the Data Warehouse Center program template. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**VWPProgramTemplateDescription*	The short description of the Data Warehouse Center program and what it does. This token is optional.	A text string, up to 254 bytes in length.

Table 56. VWPProgramTemplate.tag tokens (continued)

Token	Description	Allowed values
**VWPProgramTemplateNotes*	The long description of the Data Warehouse Center program and what it does. This token is optional.	A text string, up to 32700 bytes in length.
**VWPProgramTemplateExecutableName*	The fully qualified program name of the Data Warehouse Center program that is to run when the Data Warehouse Center runs. If the Data Warehouse Center program is installed in the system path, the warehouse program name need not be fully qualified. This token is required.	A text string, up to 240 bytes in length.
**VWPProgramTemplateType*	The type of program. This token is required.	One of the following values: **ISV_PROGRAMTYPECOMMAND** The Data Warehouse Center program is a command file. **ISV_PROGRAMTYPEDLL** The Data Warehouse Center program is loaded from a dynamic link library (DLL) or is a load module. **ISV_PROGRAMTYPEEXECUTABLE** The Data Warehouse Center program is an executable file.
**VWPProgramTemplateFunctionName*	The name of the entry point in the DLL that the Data Warehouse Center is to invoke if the value of **VWPProgramTemplateType* is ISV_PROGRAMTYPEDLL. This token is required if the value of **VWPProgramTemplateType* is ISV_PROGRAMTYPEDLL.	A text string, up to 80 bytes in length.

Table 56. VWPProgramTemplate.tag tokens (continued)

Token	Description	Allowed values
	Relationship parameters	
**VWPGroup*	The name of the group that is to contain the Data Warehouse Center program. This token is required.	A text string, up to 80 bytes in length.
**AgentSite*	The agent site to use for the source or target. This token is required.	A text string, up to 80 bytes in length. Specify `ISV_DEFAULTAGENTSITE` for the default agent site.

Table 57. Example values for VWPProgramTemplate.tag tokens

Token	Example value
**VWPProgramTemplateName*	My ISV program
**VWPProgramTemplateDescription*	This program exports data from an ODBC database.
**VWPProgramTemplateNotes*	This program will export data from an ODBC database, process it, and place it into another database.
**VWPProgramTemplateExecutableName*	c:\ISV\BIN\MYPROG.EXE
**VWPProgramTemplateType*	ISV_PROGRAMTYPEEXECUTABLE
**VWPProgramTemplateFunctionName*	My_Prog_Func_Name
**VWPGroup*	Group of programs for partner tool

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29
- "VWPProgramInstanceParameter.tag template for the Data Warehouse Center" on page 74
- "VWPProgramTemplateParameter.tag template for the Data Warehouse Center" on page 78

VWPProgramTemplateParameter.tag template for the Data Warehouse Center

Use this template to define a parameter that the Data Warehouse Center is to pass to a Data Warehouse Center program.

This template is required if the Data Warehouse Center program requires that the Data Warehouse Center pass parameters to it. You can specify that

multiple parameters are passed to the Data Warehouse Center program by including this template for each parameter.

Use this template with the VWPProgramTemplate.tag file. This template defines the relationship between the parameter and its Data Warehouse Center program definition (VWPProgramTemplate.tag).

The following tables provide information about and examples for each token in the template.

Table 58. VWProgramTemplateParameter.tag tokens

Token	Description	Allowed values
	Entity parameters	
**VWPProgramTemplateParameterName*	The name or description of a parameter that is to be passed to a Data Warehouse Center program. The name must be unique within the Data Warehouse Center program. This token is required.	A text string, up to 80 bytes in length.
**VWPProgramTemplateParameterOrder*	A number, starting with 0, that indicates the order of the parameter in the parameter list. This token is required.	A numeric value.
**VWPProgramTemplateParameterData*	The data that is passed to the Data Warehouse Center program as the value of the parameter. This token is required.	A text string or a numeric value up to 240 bytes in length.
**VWPProgramTemplateParameterKey*	A key that uniquely identifies this program parameter template. The key must be unique from all other keys in the interchange file. **Tip:** Finish processing the VWPProgramTemplateParameter.tag template before increasing the value of the key. This token is required.	A numeric value.

Table 58. VWProgramTemplateParameter.tag tokens (continued)

Token	Description	Allowed values
**VWPProgramInstanceParameterType*	The type of value that this parameter contains. For example, character, numeric, or password data.	One of the following values: **ISV_ParameterTypeNone** The parameter type is unknown or not applicable. **ISV_ParameterTypeCharacter** The parameter type is character. **ISV_ParameterTypeNumeric** The parameter type is numeric. **ISV_ParameterTypePassword** The parameter type is password.
	Relationship parameters	
**VWPProgramTemplateName*	The name of the Data Warehouse Center program that is to use this parameter. This token is required.	A text string, up to 80 bytes in length.

Table 59. Example values for VWPProgramTemplateParameter.tag tokens

Token	Example value
**VWPProgramTemplateParameterName*	DB2 UDB user ID
**VWPProgramTemplateParameterOrder*	1
**VWPProgramInstanceKey*	070000
**VWPProgramTemplateParameterData*	my_userid
**VWPProgramTemplateParameterKey*	012994
**VWPProgramInstanceParameterType*	ISV_ParameterTypePassword
**VWPProgramTemplateName*	My ISV program

Related reference:

- "Metadata templates supplied with the Data Warehouse Center" on page 29
- "VWPProgramInstanceParameter.tag template for the Data Warehouse Center" on page 74
- "VWPProgramTemplate.tag template for the Data Warehouse Center" on page 76

WarehouseDataBase.tag template for the Data Warehouse Center

Use this template to define target warehouse databases to import into the Data Warehouse Center.

This template also defines the relationship between the following objects:

- The target warehouse database
- The agent site to use for the target warehouse database
- The security group in which to define the target warehouse database

The following tables provide information about and examples for each token in the template.

Table 60. WarehouseDataBase.tag tokens

Token	Description	Allowed values
	Entity parameters	
**DatabaseName*	The unique name of the database. The name must be unique within the warehouse control database. This token is required.	A text string, up to 80 bytes in length.
**DatabaseDescription*	The short description of the database. This token is optional.	A text string, up to 254 bytes in length.
**DatabaseNotes*	The long description of the database. This token is optional.	A text string, up to 32700 bytes in length.
**DatabaseContact*	The person to contact for information about this database. This token is optional.	A text string, up to 64 bytes in length.
**DatabaseServerName*	The name of the server on which the database resides. This token is optional.	A text string, up to 64 bytes in length.
**DatabaseVersion*	The version of the database.	A text string.

Table 60. WarehouseDataBase.tag tokens (continued)

Token	Description	Allowed values
**DatabasePhysicalName*	The physical database name of the database as defined to the database manager. This token is required.	A text string, up to 40 bytes in length.
**DatabaseType*	The type of database family. This token is required.	One of the following values: **ISV_IR_DB2Family** DB2 Family **ISV_IR_GenericODBC** Generic ODBC **ISV_IR_FFLan** Flat File LAN
**DatabaseTypeExtended*	The type of AS/400 system or file. This token is required.	One of the following values: **ISV_IR_DB2400CISC** DB2 UDB for AS/400 for CISC **ISV_IR_DB2400RISC** DB2 UDB for AS/400 for RISC **ISV_IR_FFLanLocalCmd** Local flat file
**DatabaseUserid*	The user ID with which to access the database. This token is optional.	A text string, up to 36 bytes in length.
Relationship parameters		
**SecurityGroup*	The security group in which to create the source or target database. This token is required, but you can specify the default security group.	A text string, up to 80 bytes in length. Specify `ISV_DEFAULTSECURITYGROUP` for the default security group.
**AgentSite*	The agent site to use for the source or target. This token is required.	A text string, up to 80 bytes in length. Specify `ISV_DEFAULTAGENTSITE` for the default agent site.

Table 61. example values for WarehouseDataBase.tag tokens

Token	Example value
**DatabaseName*	Finance Warehouse
**DatabaseDescription*	This database contains financial information.
**DatabaseNotes*	This is the warehouse where all geographies keep financial information.
**DatabaseContact*	Valerie Zieman
**DatabaseServerName*	CHI11W71
**DatabaseVersion*	V6.1.0
**DatabasePhysicalName*	FINANCE
**DatabaseType*	DB2 Family
**DatabaseTypeExtended*	ISV_DEFAULTVALUE
**DatabaseUserid*	DB2ADMIN
**SecurityGroup*	ISV_DEFAULTSECURITYGROUP
**AgentSite*	My agent site

Related reference:

• "Metadata templates supplied with the Data Warehouse Center" on page 29

Chapter 4. Data Warehouse Center metadata

This chapter describes the Data Warehouse Center metadata that describes source databases and target databases. Other applications can export the metadata to share information about the databases.

Table 62 describes the mapping between each object in the tag language file and the corresponding logical object in the Data Warehouse Center.

Table 62. Logical objects for source and target databases

Object in tag language file	Data Warehouse Center logical object
DATABASE	A warehouse source or warehouse target
TABLE	A table, file, or IMS segment
COLUMN	A column or field

The Data Warehouse Center also defines relationships between the database, tables, and columns. The section for each object lists the relationships in which hte object participates that are useful for partner applications.

DATABASE object metadata for the Data Warehouse Center

The DATABASE object contains metadata about a source database or target database, file system, or file.

The following tables provide properties, relationships, examples of the DATABASE object.

Table 63. Properties of the DATABASE object

Tag language property name	Description	Allowed values
NAME	The business name of the source.	A text string, up to 80 bytes in length.
DBNAME	The physical database name as defined to the database manager. This value is null for generic ODBC databases, Sybase databases, IMS databases, generic ODBC databases, and file systems.	A text string, up to 40 bytes in length.
SHRTDESC	The short description of the source.	A text string, up to 200 bytes in length.

Table 63. Properties of the DATABASE object (continued)

Tag language property name	Description	Allowed values
LONGDESC	The long description of the source.	A text string, up to 32700 bytes in length.
DBTYPE	The database or file family.	One of the following values: **1** DB2 Family **20** Oracle **30** Sybase **40** Microsoft SQL Server **50** Informix **60** Generic ODBC **70** Flat File LAN **80** VSAM **90** IMS

Table 63. Properties of the DATABASE object (continued)

Tag language property name	Description	Allowed values
DBETYPE	The type of database or file within a family.	One of the following values: **1** DB2/2 **3** DB2 MVS **4** AS/400 CISC **5** AS/400 RISC **6** DB2/6000 **8** DB2 HP **9** DB2 SUN **11** DB2 NT **12** DB2 VM **13** DB2 SINIX **14** DB2 SCO **15** DB2 VSE **16** DB2 ESE **18** DB2 family **19** DataJoiner **20** Oracle **30** Sybase **40** Microsoft SQL Server **50** Informix **60** User-defined ODBC
DBETYPE (continued)	The type of database or file within a family.	One of the following values: **70** Flat File LAN Local Command **71** Flat File LAN FTP Copy **80** VSAM **90** IMS

Table 63. Properties of the DATABASE object (continued)

Tag language property name	Description	Allowed values
ISWH	A flag that indicates whether this source is a warehouse target or warehouse source.	One of the following values: **Y** This source is a warehouse target. **N** This source is a warehouse source.
USERID	The user ID that the Data Warehouse Center uses to connect to the source.	A text string, up to 36 bytes in length.
CONTACT	The name of the person who is responsible for the source.	A text string, up to 64 bytes in length.
USEODBC	A flag that specifies whether to use the user-supplied connect string or to generate the string. Use N for files.	One of the following values: **Y** Use the user-defined connect string. **N** Generate the connect string.
ODBCSTR	The user-defined ODBC connect string to use if USEODBC is set to Y. Otherwise, this property is null.	A text string, up to 254 bytes in length.
PREACCMD	If the source is a local Flat File LAN source, a command to run to access the remote file.	A text string, up to 64 bytes in length.
POSTACMD	If the source is a local Flat File LAN source, a command to run after accessing the remote file.	A text string, up to 64 bytes in length.
RETRYCNT	The number of times to try to extract data from this source in case of an error.	A numeric value.
RETRYINT	The time that is to elapse between attempts to extract data.	A numeric value.
VERSION	The version of DB2 in use.	A text string, up to 128 bytes in length.
DBMSSERV	The database instance/subsystem/server name for ODBC connect.	A text string, up to 128 bytes in length.
DFLTDEL	The System 390 database default character string delimiter.	A text string, up to 1 byte in length.

The following figure shows an example of a DATABASE object instance that defines a target warehouse database.

```
:COMMENT.  Begin DATABASE Instance
:COMMENT.
:ACTION.OBJINST(MERGE)
```

```
:OBJECT.TYPE(DATABASE)
:INSTANCE.
    NAME(iwhtar)
    DBNAME(IWHTAR)
    DBTYPE(1)
    DBETYPE(11)
    ISWH(Y)
    USERID(marlow)
    USEODBC(N)
    CODEPAGE(437)
    RETRYCNT(3)
    RETRYINT(30)
```

The following figure shows an example of a DATABASE object instance that defines a source file.

```
 :ACTION.OBJINST(MERGE)
 :OBJECT.TYPE(DATABASE)
 :INSTANCE.
     NAME(TBC Operations)
     SHRTDESC(The Beverage Company operational data sources)
     DBTYPE(70)
     DBETYPE(70)
     ISWH(N)
     LOCATION(Thirsty City)
     USERID(XXXXXXXX)
     USEODBC(N)
     CODEPAGE(437)
     RETRYCNT(0)
     RETRYINT(0)
```

The following figure shows an example of a relationship between a DATABASE object instance and a TABLES object instance.

```
:COMMENT.  Relation: DATABASE to TABLES
:COMMENT.
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(CONTAIN) SOURCETYPE(DATABASE) TARGETYPE(TABLES)
:INSTANCE.
    SOURCEKEY(NAME(TBC Operations) DBNAME() )
    TARGETKEY(DBNAME(TBC Operations) OWNER() TABLES(d:\iwhdemo\outcusti.txt) )
```

The following table shows the relationship in which the DATABASE object participates and that is useful for partner applications. The Source column and the Target column indicate how many times the source object or the target object of the relationship can participate in the relationship.

Table 64. Relationships in which the DATABASE object participates

Source	Source tag language object type	Relation type	Target	Target tag language object type	Description
1	DATABASE	CONTAIN	M	TABLES	Tables or files that are contained in the database or file system.

Related reference:

- "TABLES object metadata for the Data Warehouse Center" on page 90
- "COLUMN object metadata for the Data Warehouse Center" on page 95

TABLES object metadata for the Data Warehouse Center

The TABLES object contains metadata about a warehouse source table, segment, or file, or a target table. It is associated with a DATABASE object.

The following tables provide properties, relationships, and examples of the TABLES object.

Table 65. Properties of the TABLES object

Tag language property name	Description	Allowed values
NAME	The name of the table, file, or IMS segment. The table name includes the high-level qualifier, schema or collection, such as IWH.TABLE1. The combination of the database name and the table name is unique. This property is the fully qualified path and file name for a file.	A text string, up to 80 bytes in length.
SHRTDESC	The short description of the file or segment.	A text string, up to 200 bytes in length.
LONGDESC	The long description of the table.	A text string, up to 32700 bytes in length.
DBNAME	The business name of the source that contains this table or file.	A text string, up to 80 bytes in length.

Table 65. Properties of the TABLES object (continued)

Tag language property name	Description	Allowed values
OWNER	The owner, high-level qualifier, or collection of the table. This property is null for files and IMS segments.	A text string, up to 15 bytes in length.
TABLES	The physical table, file, or segment name as defined to the database manager or file system. For files and IMS segments, this value is the same as the value of NAME.	A text string, up to 80 bytes in length.
TBLISBIN	A flag that specifies the file transfer mode for Flat File LAN files.	One of the following values: **Y** The file transfer mode is binary. **N** The file transfer mode is ASCII.
TBLNAMESP	The name of the DB2 table space.	A text string, up to 90 bytes in length.
TBLFTYPE	For files, the type of the file.	One of the following values: **1** Fixed **2** Comma **3** Tab **4** Character
TBLL1NAM	A flag that specifies whether the first row of the file contains column names.	One of the following values: **Y** The first row of the file contains column names. **N** The first row of the file contains data.
CHARDELM	For files, the character separator if the file type is character.	A text string that is 1 byte in length.

Table 65. Properties of the TABLES object (continued)

Tag language property name	Description	Allowed values
CREATYPE	The method used to define the table in the Data Warehouse Center.	One of the following values: **1** The table was defined manually. **2** The table definition was imported from the database manager. **3** The table definition was imported from the Information Catalog Center. **4** The table was created by the Data Warehouse Center for a step when the step was promoted to test mode.
TABALIAS	A flag that specifies whether the table has an alias.	One of the following values: **Y** The table has an alias. **N** The table does not have an alias.
IWHCRTAR	A flag that specifies whether the target table is created by the Data Warehouse Center.	One of the following values: **Y** The target table is created by the Data Warehouse Center. **N** The target table is not created by the Data Warehouse Center.
IWHGRANT	A flag that specifies whether GRANT TO PUBLIC is enabled for the table.	One of the following values: **Y** GRANT TO PUBLIC is enabled for the table. **N** GRANT TO PUBLIC is been enabled for the table.
IWHDRATN	The warehouse target duration, either transient or persistent.	One of the following values: **Y** The table is persistent. **N** The table is transient.
IWHMAXED	The maximum number of editions of the table.	A numeric value.

Table 65. Properties of the TABLES object (continued)

Tag language property name	Description	Allowed values
IWHCREGN	A flag that specifies whether the create statement is automatically generated.	One of the following values: **Y** The Create statement is automatically generated. **N** The Create statement is not automatically generated.
IWHCRERU	The create statement for the table.	A text string, up to 32,700 bytes in length.
IDSFACT	A flag that specifies whether the table is used as a fact table.	One of the following values: **Y** The table is used as a fact table. **N** The table is not used as a fact table.
CDSSCHEMA	The table schema for replication.	A text string, up to 128 bytes in length.
CDTABNAM	The table name for replication.	A text string, up to 128 bytes in length.
BEFORIMG	The replication before-image prefix.	A text string, up to 4 bytes in length.
IDSREPL	A flag that specifies whether the table is used for replication.	One of the following values: **Y** The table is used for replication. **N** The table is not used for replication.
NAMINDEX	The DB2 table name index.	A text string, up to 90 bytes in length.
PARTTBSP	A flag that specifies whether the table is in a partitioned table space.	One of the following values: **Y** The table is in a partitioned table space. **N** The table is not in a partitioned table space.
DBNAM390	The System 390 database name.	A text string, up to 8 bytes in length.

The following figure shows an example of a TABLES object instance for a relational table.

```
:COMMENT.  Begin TABLES Instance
:COMMENT.
:ACTION.OBJINST(MERGE)
```

```
:OBJECT.TYPE(TABLES)
:INSTANCE.
    NAME(IWH.ATOMICED)
    DBNAME(iwhtar)
    OWNER(IWH)
    TABLES(ATOMICED)
    TBLISBIN(N)
    TBLFTYPE(0)
    TBLL1NAM(N)
    CREATYPE(4)
:COMMENT.
:COMMENT.  End TABLES Instance
```

The following figure shows an example of a TABLES object instance for a file.

```
:COMMENT.  Begin TABLES Instance
:COMMENT.
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(TABLES)
:INSTANCE.
    NAME(d:\iwhdemo\outcusti.txt)
    SHRTDESC(File containing operational data for Institutions Customers)
    DBNAME(TBC Operations)
    OWNER()
    TABLES(d:\iwhdemo\outcusti.txt)
    TBLISBIN(Y)
    TBLFTYPE(3)
    TBLL1NAM(N)
    CREATYPE(1)
:COMMENT.
:COMMENT.  End TABLES Instance
```

The following figure shows an example of a relationship between a TABLES object instance and a DATABASE object instance.

```
:COMMENT.  Relation: DATABASE to TABLES
:COMMENT.
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(CONTAIN) SOURCETYPE(DATABASE) TARGETYPE(TABLES)
:INSTANCE.
    SOURCEKEY(NAME(TBC Operations) DBNAME() )
    TARGETKEY(DBNAME(TBC Operations) OWNER() TABLES(d:\iwhdemo\outcusti.txt) )
```

The following figure shows an example of a relationship between a TABLES object instance and a COLUMN object instance.

```
:COMMENT.  Relation: TABLES to COLUMN
:COMMENT.
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(CONTAIN) SOURCETYPE(TABLES) TARGETYPE(COLUMN)
:INSTANCE.
    SOURCEKEY(DBNAME(TBC Operations) OWNER() TABLES(d:\iwhdemo\outcusti.txt) )
    TARGETKEY(DBNAME(TBC Operations) OWNER() TABLES(d:\iwhdemo\outcusti.txt)
   COLUMNS(Zipcode) )
```

The following table lists the relationships in which the TABLES object participates and that are useful for partner applications. The Source column and the Target column indicate how many times the source object or target object of the relationship can participate in the relationship.

Table 66. Relationships in which the TABLES object participates

Source	Source tag language object type	Relation type	Target	Target tag language object type	Description
1	DATABASE	CONTAIN	M	TABLES	Database or file system with which this table or file is associated.
1	TABLE	CONTAIN	M	COLUMN	Columns associated with this table.

Related reference:

- "DATABASE object metadata for the Data Warehouse Center" on page 85
- "COLUMN object metadata for the Data Warehouse Center" on page 95

COLUMN object metadata for the Data Warehouse Center

The COLUMN object contains metadata about a column or field in a source table, target table, or file. It is associated with a TABLES object.

The following tables provide the properties, relationships, and examples of the COLUMN object.

Table 67. Properties of the COLUMN object

Tag language property name	Description	Allowed values
NAME	The name of the column or field. The combination of the database name, table name, and column name is unique.	A text string, up to 80 bytes in length.
SHRTDESC	The short description of the column or field.	A text string, up to 200 bytes in length.
LONGDESC	The long description of the column or field.	A text string, up to 32700 bytes in length.

Table 67. Properties of the COLUMN object (continued)

Tag language property name	Description	Allowed values
DATATYPE	The ODBC data type to which the database manager data type maps. The Data Warehouse Center derives the data type from the native data type. You cannot add a GRAPHIC data type column to a table in a VSAM database.	One of the following values: CHAR NUMERIC DECIMAL INTEGER SMALLINT FLOAT DOUBLE DATE TIME TIMESTAMP VARCHAR LONG_VARCHAR GRAPHIC VARGRAPHIC LONG_VARGRAPHIC BLOB CLOB DBCLOB TINYINT BIT REAL BIGINT
LENGTH	The length of the column or field.	A numeric value.
SCALE	The precision of the column or field for columns or fields with a decimal data type.	A numeric value.
POSNO	An index, starting with 0, of the column or field in the row of the table or file.	A numeric value.
NULLS	A flag that specifies whether the column or field allows null data.	One of the following values: **Y** The column allows null data. **N** The column does not allow null data.
ISTEXT	A flag that specifies whether the column or field data is binary or text data.	One of the following values: **Y** The column data is binary data. **N** The column data is text data.
DBNAME	The business name of the source or target that contains this table or file.	A text string, up to 80 bytes in length.

Table 67. Properties of the COLUMN object (continued)

Tag language property name	Description	Allowed values
OWNER	The owner, high-level qualifier, or collection of the table. This property is null for files and IMS segments.	A text string, up to 15 bytes in length.
TABLES	The physical table, file, or segment name as defined to the database manager or file system. For files and IMS segments, this value is the same as the value of NAME.	A text string, up to 80 bytes in length.
NATIVEDT	Native data type of the column or field.	The data type for the column as defined to the database manager. The data type is a text string, up to 40 bytes in length. In most cases, the value of this property will match the value of DATATYPE. For the mapping of the database manager data types to ODBC data types, see the Data Warehouse Center online help.
ORDINAL	Column or field ordinality.	A numeric value.
OFFSET	The offset of the field in a fixed-length file.	A numeric value.
COLTYPE	The column type for DPropR.	One of the following values: **A** After image column **B** Before image column

The following figure shows an example of a COLUMN object instance.

```
:ACTION.OBJINST(MERGE)
:OBJECT.TYPE(COLUMN)
:INSTANCE.
    NAME(CORR_COEF)
    SHRTDESC(Correlation Coefficient)
    DATATYPE(DOUBLE)
    LENGTH(0)
    SCALE(0)
    POSNO(4)
    NULLS(Y)
```

```
    ISTEXT(N)
    DBNAME(TRANSFORMER_TARGET)
    OWNER(IWH)
    TABLES(TR_CORRELATION_06)
    COLUMNS(CORR_COEFF)
    NATIVEDT(DOUBLE)
    TRANSNAM(Correlation Coefficient(r))
```

The following figure shows an example of a relationship between a COLUMN object instance and a TABLES object instance.

```
:COMMENT.  Relation: TABLES to COLUMN
:COMMENT.
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(CONTAIN) SOURCETYPE(TABLES) TARGETYPE(COLUMN)
:INSTANCE.
    SOURCEKEY(DBNAME(TBC Operations) OWNER() TABLES(d:\iwhdemo\outcusti.txt) )
    TARGETKEY(DBNAME(TBC Operations) OWNER() TABLES(d:\iwhdemo\outcusti.txt)
     COLUMNS(Zipcode) )
```

The following table shows the relationship in which the COLUMN object participates. This relationship is useful for partner applications. The Source column and the Target column indicate how many times the source object or the target object of the relationship can participate in the relationship.

Table 68. Relationship in which the COLUMN object participates

Source	Source tag language object type	Relation type	Target	Target tag language object type	Description
1	TABLES	CONTAIN	M	COLUMN	The table with which this column is associated.

Related reference:

- "DATABASE object metadata for the Data Warehouse Center" on page 85
- "TABLES object metadata for the Data Warehouse Center" on page 90

Chapter 5. Information Catalog Manager object types

This chapter provides detailed information about Information Catalog Manager object types.

Default properties for all Data Warehouse Center objects

The Information Catalog Center provides a set of default properties for the generic object. These default properties serve as the base for any user-defined tables. Some properties are generated by the Information Catalog Center; some of these properties are required, and some are optional.

FLGID

An ID, generated by the Information Catalog Center, that uniquely identifies an instance.

The FLGID ID is 16 digits, with the first 6 digits used for the object ID (OBJTYPID) and the next 10 digits used for the instance ID (INSTIDNT). FLGID has the following format:

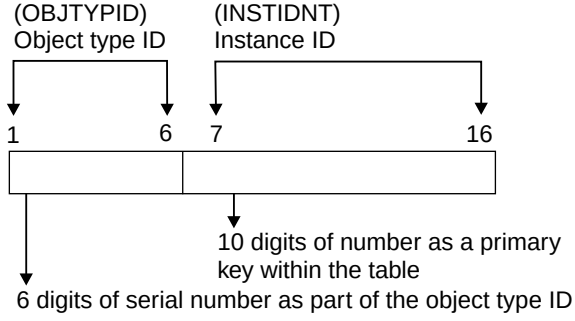


Figure 3. FLGID Format

Name Name of the step. The name can be used on glossary, news queries, and other objects. This is a required property, and it is not nullable. It is displayed in the Information Catalog Center windows.

UPDATIME

A system time stamp that indicates the date and time of the creation or last update to the instance.

UPDATEBY

The user ID of the information catalog administrator or user with special privileges who last updated the instance. For Attachment objects, this field can be the user ID of an information catalog user.

The information catalog administrator can use the predefined template to create an object. The information catalog administrator can append attributes to the template to customize it for the organization. The predefined template has several optional fields. The following table shows the default properties.

Table 69. Default properties of the predefined template

Column name	Data type	Description	Nullable	NLS
OBJTYPID	CHAR(6)	A six-digit object ID, generated by the Information Catalog Center, that represents a specific object.	No	SBCS
INSTIDNT	CHAR(10)	The unique instance ID generated by the Information Catalog Center. It is the second part of the FLGID, the 10-digit serial number that uniquely identifies this instance within its own object.	No	SBCS
NAME	VARCHAR(80)	This name is entered by the information catalog user to identify each user-defined object instance in the product.	No	Both SBCS and DBCS
UPDATIME	CHAR (26)	The date and time of metadata creation or last update. This value is generated by the Information Catalog Center.	No	SBCS
UPDATEBY	CHAR(8)	The user ID of the information catalog administrator or user with special update privileges who last updated the instance. For attachment objects this field might be the user ID of the information catalog user. This value is generated by the Information Catalog Center.	No	Both SBCS and DBCS

Note:

NLS: National Language Support

SBCS: Single Byte Character Set

DBCS: Double Byte Character Set

Guidelines for extendible objects types for the Data Warehouse Center

1. An object is extendible if it can be changed. An object category is extendible if other objects can be added to it.
2. All objects must include a universal unique identifier, UUI, as part of their object definition. The UUI is used to compare with a similar identifier in the target information catalog during the import process.
3. If the property has a data type such as LONG VARCHAR, the Information Catalog Center will automatically put the property and its metadata into a

separate overflow table and split the property into smaller segments so a user can search for it. The search will proceed slowly because of the size of the property.

Valid data types for Information Catalog Center descriptive data

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

Table 70. Valid data types for Information Catalog Center descriptive data

Data type	Description
INTEGER (I)	A integer is a four byte integer with a precision of 10 digits. The range of integers is -2 147 483 648 to +2 147 483 647.
SMALLINT (S)	A small integer is a two byte integer with a precision of 5 digits. The range of small integers is -32 768 to 32 767.
BIGINT (G)	A big integer is an eight byte integer with a precision of 19 digits. The range of big integers is -9 223 372 036 854 775 808 to +9 223 372 036 854 775 807.
DECIMAL (E)	A decimal value is a packed decimal number with an implicit decimal point. The position of the decimal point is determined by the precision and the scale of the number. The scale, which is the number of digits in the fractional part of the number, cannot be negative or greater than the precision. The maximum precision is 31 digits
DOUBLE (U)	A double-precision floating-point number is a 64 bit approximation of a real number. The number can be zero or can range from -1.79769E+308 to -2.225E-307, or from 2.225E-307 to 1.79769E+308.
REAL (R)	A single-precision floating-point number is a 32 bit approximation of a real number. The number can be zero or can range from -3.402E+38 to -1.175E-37, or from 1.175E-37 to 3.402E+38.
BLOB (B)	Binary large object. A sequence of bytes with a size ranging from 0 bytes to 2 gigabytes, less 1 byte. You cannot specify a property with a data type of BLOB as a unique identifier property.
CLOB (O)	Character large object. A sequence of characters (single-byte, multibyte, or both) with a size ranging from 0 bytes to 2 gigabytes, less 1 byte. You cannot specify a property with a data type of CLOB as a unique identifier property.

Table 70. Valid data types for Information Catalog Center descriptive data (continued)

Data type	Description
CHAR (C)	Fixed-length character string between 1 and 254 bytes long. Pad the value on the right with trailing blanks if the value is shorter than the defined data length for the property.
TIMESTAMP (T)	26-character timestamp in the following format: `yyyy-mm-dd-hh.mm.ss.nnnnnn`
TIME (M)	15-character time in the following format: `hh.mm.ss.nnnnnn`
DATE (D)	10-character date in the following format: `yyyy-mm-dd`
LONG VARCHAR (L)	Long varying-length character string between 1 and 32 700 bytes long. You cannot specify a property with a data type of LONG VARCHAR as a unique identifier property.
VARCHAR (V)	Varying-length character string between 1 and 32 672 bytes long.

The Information Catalog Center automatically removes the trailing blanks from variable values and adjusts their length accordingly before validating and accepting the request.

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

Related concepts:

- "Object definition for the Data Warehouse Center" on page 11

Related reference:

- "Tag language" on page 165

Information Catalog Center predefined object types

The Information Catalog Center includes predefined object types that you can exchange with metadata from other Data Warehouse Center components. The sample information catalog contains the predefined object types and sample objects of each type. The following list gives a brief description of each object type.

Application data
The Information Catalog Center uses the Application data object type internally for some data exchanges. Objects of this object type might appear in your information catalog. However, you will not use this object type to create objects.

Attribute
The attribute object type represents an attribute of an entity.

Audio clips
The Audio clips object type represents files that contain audio information. These objects might represent electronic (AUD files) or physical (for example, CDs, tapes) audio information.

Business subject areas
The Business subject areas object type represents logical groupings of objects.

Case models
The Case Models object type represents the logical or physical representation of data such as a table.

Charts The Charts object type represents either hardcopy or electronic charts.

Column mapping
The Column mapping object type represents column mappings in the Data Warehouse Center.

Columns or fields
The Columns or fields object type represents columns within a relational table, fields within a file, or fields within an Internet Management Specification (IMS) segment.

Comments
The Comments object type holds comments about other objects in the information catalog. The Comments object type is created when an information catalog is created.

Databases
The Databases object type represents relational databases.

Dimensions within a multidimensional database
The Dimensions within a multidimensional database object type represents dimensions within a multi-dimensional database. A dimension consists of members.

Documents
The Documents object type represents books, manuals, and technical papers. These publications might be printed or electronic, found locally or within a library.

DWC process
This object type represents a process in the Data Warehouse Center. A process commonly operates on source data and changes data from its original form into a form conducive to decision support. In the Data Warehouse Center, a process commonly consists of one or more sources, one or more steps, and one or more targets.

Elements
The Elements object type represents element objects that do not map directly to the Columns or fields object type.

Entity The Entity object type represents an entity within case model.

Files The Files object type represents a file within a file system.

Glossary entries
The Dictionary category contains the Glossary entries object type. The Glossary entries object type represents definitions for terms that are used in the information catalog.

Images or graphics
The Images or graphics object type represents graphic images, such as bitmaps.

IMS database definitions (DBD)
The IMS database definitions (DBD) object type represents IMS database definitions.

IMS program control blocks (PCB)
The IMS program control blocks (PCB) object type represents IMS program control blocks.

IMS program specification blocks (PSB)
The IMS program specification blocks (PSB) object type represents IMS program specification blocks.

IMS segments
The IMS segments object type represents IMS segments.

Information Catalog Center news
The Information Catalog Center news object type conveys information to end users about changes to the information catalog.

Internet documents
The Internet documents object type represents Web sites and other documents on the Internet that might be of interest.

Lotus Approach queries
The Lotus Approach queries object type represents available Lotus Approach queries for use with your organization's data.

Members within a multidimensional database
The Members within a multidimensional database object type represents a member within a multidimensional database. A member is part of a dimension, and a dimension is part of a multidimensional database.

Multidimensional databases

The Multidimensional databases object type represents multidimensional databases.

OLAP integration server model

The OLAP integration server model object type represents an OLAP Integration Server model. It can be linked to one or more OLAP Multi-dimensional database objects.

Online news services

The Online news services object type represents news services and information services that you can access online.

Online publications

The Online publications object type represents publications and other documents that you can access through online services.

People to contact

The People to contact object type identifies a person or group that is responsible for single or multiple objects within the information catalog.

Presentations

The Presentations object type represents various hardcopy or electronic presentations. These presentations might include product, customer, quality, and status presentations.

Programs

The Program category can contain only the Programs object type. The Programs object type is created when an information catalog is created. It is used to define an application capable of processing a particular object type.

In the sample information catalog the Programs object type is named ″Programs that can be invoked from information catalog objects″ .

Records

The Records object type represents record objects that do not map directly to the Files or Relational tables or views object types. Records consist of Elements.

Relational tables and views

The Relational tables and views object type represents tables or views of relational databases.

Spreadsheets

The Spreadsheets object type represents desktop spreadsheets (for example, Lotus 1-2-3 or Microsoft Excel spreadsheets).

Star Schemas

This object type represents a relational star schema structure. A star schema contains a fact table and one or more dimension tables.

Subschemas

The Subschemas object type represents logical groupings of records within a database.

Text-based reports

The Text-based reports object type represents either hardcopy or electronic reports.

Transformations

The Transformations object type represents expressions or logic that is used to populate columns of data within the target database. Transformations objects indicate either the expression used to convert source-operational data to target columns, or the one-to-one mapping of source fields to target columns.

Video clips

The Video clips object type represents files that contain video information. These objects might represent electronic (AVI files) or physical (for example, videotapes or laser disks) video information.

Related concepts:

- "Object types" in the *Information Catalog Center Administration Guide*

Related reference:

- "Predefined relationship type models" on page 109
- "Predefined program objects" in the *Information Catalog Center Administration Guide*

Relation types for the Data Warehouse Center

1. The Information Catalog Center supports the following types of relationships that are created and deleted through the same FLGRelation API. Different APIs, such as FLGNavigate, FLGWhereUsed, and FLGListContacts are used to access each type of the relationship. These APIs call their corresponding IPIs to complete the user's request.
   a. Contains (C)

      For example: a hierarchical business structure or a relational table to the relational columns.

      This relation is retrieved by APIs such as FLGNavigate and FLGWhereUsed.

   b. Contact (T)

For example: the name of a person providing services for specified objects.

The FLGListContacts API is used to access this relation.

c. Attaches relationship (A)

For example: comments for a specified object.

The FLGListAssociates and FLGFoundIn API are used to retrieve this relation.

d. Link relationship (L)

A grouping or elemental category object instance can link to any other grouping or elemental category object instance.

The FLGListAssociates API is used to retrieve this relation.

2. Objects are not required to have relationships. You can find all objects by using the Information Catalog Center windows, the FLGSearch API, or by viewing the FLG.NAMEINST table.

If there is a relation between two object instances, this instance-to-instance relation is added to the relation instance table.

The table has the following format:

FLGID of source (16 digits)	FLGID of target (16 digits)	RelType C/T/L/A

Predefined Data Warehouse Center objects

The Information Catalog Center 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 all of the predefined Information Catalog Center object types, including how the object properties map to MDIS object types.

The Information Catalog Center 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 for each of the seven Information Catalog Center categories. This section describes how to create the sample information catalog.

The following table lists all the object types in the sample information catalog. Object types can represent data or a relationship between two object types.

Object types that represent data

Most predefined object types represent types of data such as the Charts or Documents object types.

Object types that represent relationships

The Transformations object is a special object that represents a relationship between two other object types. Specifically, it represents the transformation of data from the data's source format to its target format. You can use Transformations object types to provide information about the lineage of the data within a target relational database.

Table 71. Predefined data object types summary

Object type name	Description
Application data	Internal use only
Audio clips	Represents files that contain audio information
Business subject areas	Represents logical grouping of objects
Charts	Represents either printed or electronic charts
Columns or fields	Represents columns within a relational table, fields within a file, or fields within an IMS segment
Comments	Contains comments about other objects in the information catalog
Databases	Represents relational databases
Information Catalog Center news	Conveys information about changes to the information catalog
Dimensions within a multi-dimensional database	Represents dimensions within a multidimensional database
Documents	Represents books, manuals, and technical papers
Elements	Represents MDIS Element objects that do not map directly to the "Columns or fields" object
Files	Represents a file within a file system
Glossary entries	Represents definitions for terms used in the information catalog
Images or graphics	Represents graphic images, such as bitmaps
IMS database definitions (DBD)	Represents IMS database definitions
IMS program control blocks (PCB)	Represents IMS program control blocks
IMS program specification blocks (PSB)	Represents IMS program specification blocks
IMS segments	Represents IMS segments
Internet documents	Represents Web sites and other documents on the Internet that might be of interest

Table 71. Predefined data object types summary (continued)

Object type name	Description
Lotus® Approach® queries	Represents available Lotus Approach queries for use with your organization's data
Members within a multidimensional database	Represents a member within a multidimensional database
Multidimensional databases	Represents multidimensional databases
Online news services	Represents news and information services that can be accessed online
Online publications	Represents publications and other documents that can be accessed from online services
People to contact	Identifies a person or group that is responsible for single or multiple objects within the information catalog
Presentations	Represents printed or electronic presentations
Programs that can be invoked from Information Catalog Center objects	Defines an application capable of processing a particular object
Records	Represents MDIS Record objects that do not map directly to the "Files" or "Relational tables or views" object
Relational tables and views	Represents tables or views of relational databases
Subschemas	Represents logical groupings of records within a database
Transformations	Represents expressions or logic used to populate columns of data within the target relational database
Spreadsheets	Represents desktop spreadsheets (for example, Lotus 1-2-3® or Microsoft Excel spreadsheets)
Text-based reports	Represents either printed or electronic reports
Video clips	Represents files that contain video information

Predefined relationship type models

Information Catalog Center predefined object types follow the data models shown in the following figures. The figures show how the predefined relationship types work with the predefined object types.

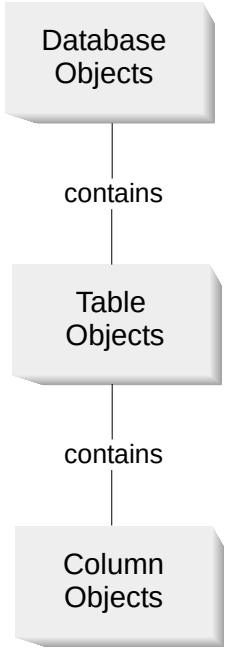


Figure 4. Relational model with the Contains relationship type

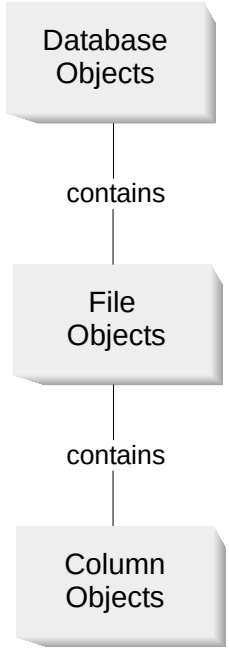


Figure 5. File model with the Contains relationship type

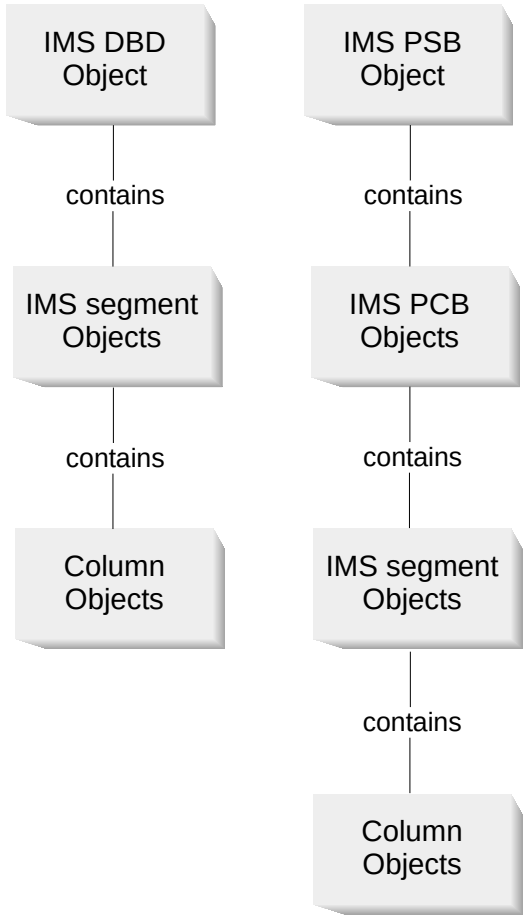


Figure 6. IMS models with the Contains relationship type

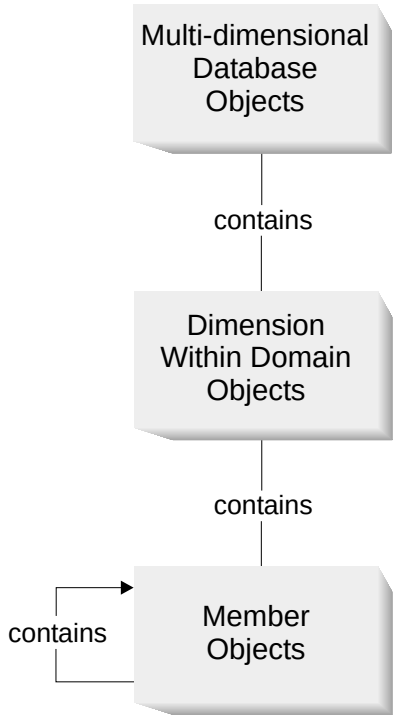


Figure 7. Multi-dimensional model with the Contains relationship type

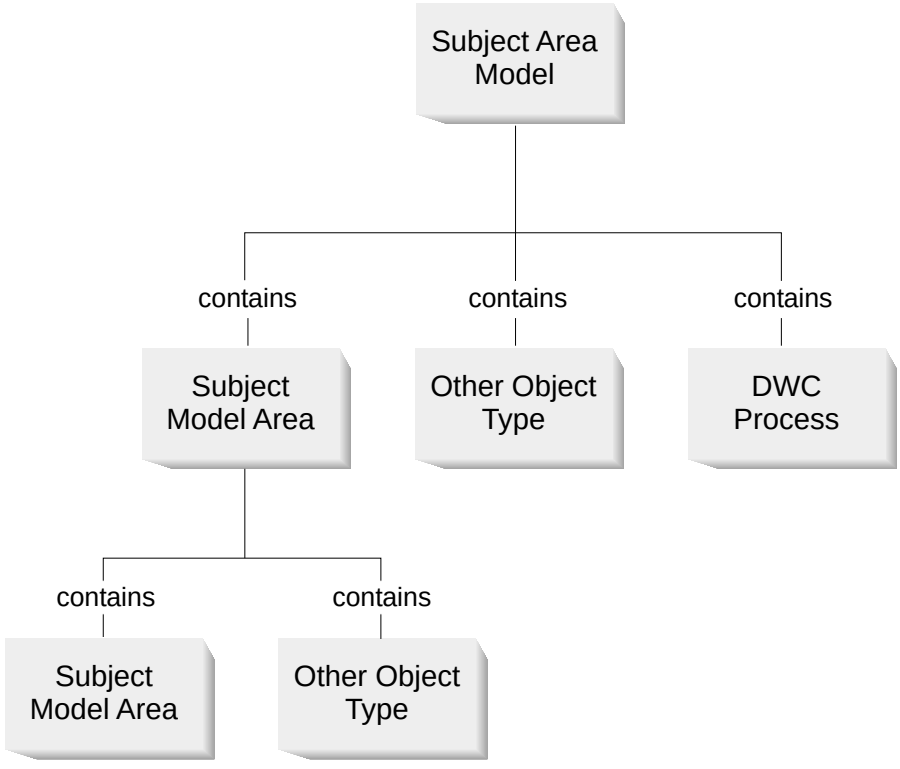


Figure 8. Subject area model with the Contains relationship type

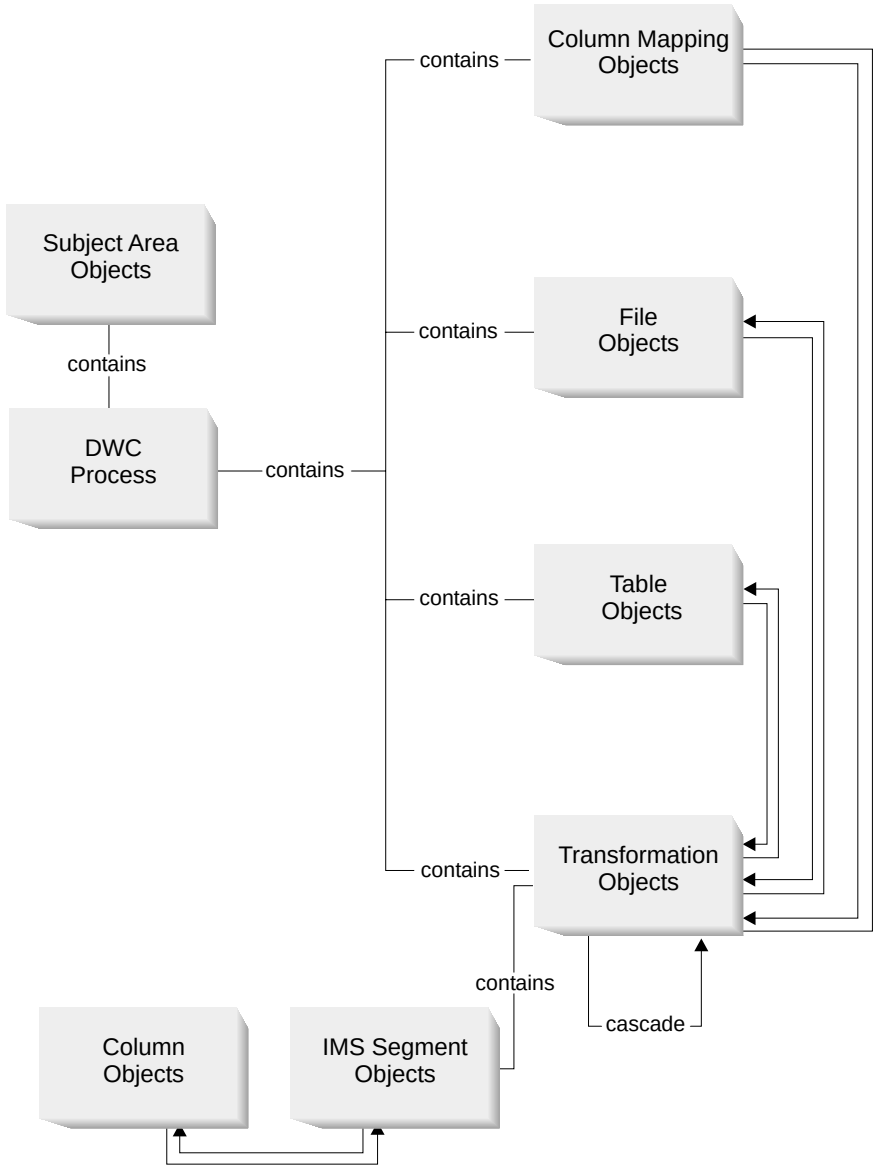


Figure 9. Process and transformation model with the Cascade and Contains relationship types

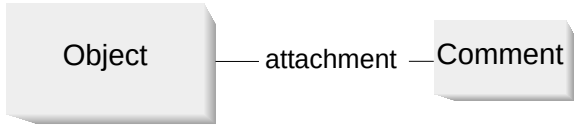


Figure 10. Attachment model with the Attachment relationship type

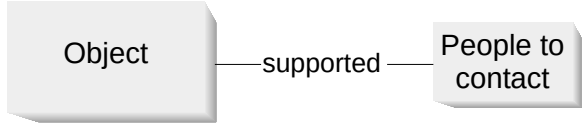


Figure 11. Contact model with the Contact predefined relationship type

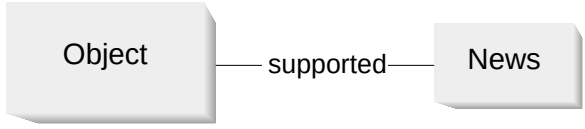


Figure 12. Supported model with the Supported predefined relationship type

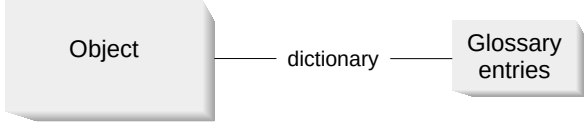


Figure 13. Dictionary model with the Dictionary predefined relationship type

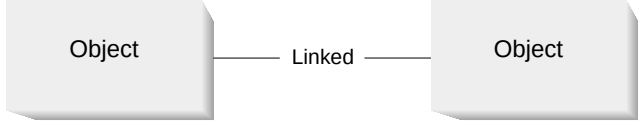


Figure 14. Linked model with the Linked predefined relationship type

Related concepts:

- "Relationship types" in the *Information Catalog Center Administration Guide*

Related tasks:

- "Adding a relationship between objects" in the *Information Catalog Center Administration Guide*
- "Removing a relationship between objects" in the *Information Catalog Center Administration Guide*

Related reference:

- "Mapping Version 7 object type categories to Version 8 relationship types, categories, and roles" in the *Information Catalog Center Administration Guide*
- "Information Catalog Center predefined object types" on page 102

Predefined object descriptions: Application data

Application data is used by the Information Catalog Center for some MDIS metadata exchanges. Objects of this object might appear in your information catalog, but you do not use this object to create objects.

The Information Catalog Center DB2 storage table name for this object is XXX.APPLDATA.

The following table provides information about the properties of the Application data object.

Table 72. Properties of the Application data object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	O	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Source object identifier	CHAR	16	FLGID	R	1
Application data field 0	LONG VARCHAR	32700	APPLDAT0	O	
Application data field 1	LONG VARCHAR	32700	APPLDAT1	O	
Application data field 2	LONG VARCHAR	32700	APPLDAT2	O	

Table 72. Properties of the Application data object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Application data field 3	LONG VARCHAR	32700	APPLDAT3	O	
Application data field 4	LONG VARCHAR	32700	APPLDAT4	O	
Application data field 5	LONG VARCHAR	32700	APPLDAT5	O	
Application data field 6	LONG VARCHAR	32700	APPLDAT6	O	
Application data field 7	LONG VARCHAR	32700	APPLDAT7	O	
Application data field 8	LONG VARCHAR	32700	APPLDAT8	O	
Application data field 9	LONG VARCHAR	32700	APPLDAT9	O	
Timestamp source definition created	CHAR	26	CRTTIME	O	
Timestamp source definition last changed	CHAR	26	SRCDATCF	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Business subject areas

Business subject areas represent logical groupings of objects.

The Information Catalog Center DB2 storage table name for this object is XXX.INFOGRPS.

The following table provides information about the properties of the Business subject areas object.

Table 73. Properties of the Business subject areas object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	

Table 73. Properties of the Business subject areas object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	1
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Filename	VARCHAR	254	FILENAME	O	
URL to access data	VARCHAR	254	URL	O	
	VARCHAR	80	CONTACT	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Colums or fields

The Columns or fields object represents columns within a relational table, fields within a file, or fields within an IMS segment.

The Information Catalog Center DB2 storage table name for this object is XXX.COLUMNS.

The following table provides information about the properties of the Columns or fields object.

Table 74. Properties of the Columns or fields object types. The MDIS name for this object is Element.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		

Table 74. Properties of the Columns or fields object types (continued). The MDIS name for this object is Element.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name
Name	VARCHAR	80	NAME	R		ElementLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
Catalog remarks	VARCHAR	254	REMARKS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Column or field last refreshed	CHAR	26	FRESHDAT	O		ElementLastRefreshDate
Data type of column or field	CHAR	30	DATATYPE	O		ElementDataType
Length of column or field	CHAR	20	LENGTH	O		ElementLength
Scale of column or field	CHAR	5	SCALE	O		ApplicationData
Precision of column or field	CHAR	5	PRECDIG	O		ElementPrecision
Can column or field be null	CHAR	1	NULLS	O		ElementNulls
Column or field ordinality	CHAR	5	ORDINAL	O		ElementOrdinality
Column or field position	CHAR	5	POSNO	O		ElementPosition

Table 74. Properties of the Columns or fields object types (continued). The MDIS name for this object is Element.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name
Byte offset of column or field from start	CHAR	10	STARTPOS	O		ApplicationData
Is column or field part of a key	CHAR	1	ISKEY	O		ApplicationData
Is column or field a unique key	CHAR	1	UNIQKEY	O		ApplicationData
Position of column or field within key	CHAR	5	KEYPOSNO	O		ElementKeyPosition
Database host server name	VARCHAR	80	SERVER	O		ServerName
Database or subsystem name	VARCHAR	80	DBNAME	R	1	DatabaseName
Table owner	VARCHAR	80	OWNER	R	2	OwnerName
Table name	VARCHAR	80	TABLES	R	3	RecordName
Column or field name	VARCHAR	254	COLUMNS	R	4	ElementName
Filename	VARCHAR	254	FILENAME	R	5	ApplicationData
URL to access data	VARCHAR	254	URL	O		ApplicationData
Containing dimension	VARCHAR	80	DIMENSION	O		DimensionName
Is data a before or after image, or computed	CHAR	50	COLIMAGE	O		ApplicationData
Source column or field name or expression used to populate column	VARCHAR	254	COLEXPR	O		ApplicationData

Table 74. Properties of the Columns or fields object types (continued). The MDIS name for this object is Element.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name
String used to represent null values	VARCHAR	30	IDSNREP	O		ApplicationData
Resolution of dates	CHAR	1	IDSRES	O		ApplicationData
Is data text	CHAR	1	ISTEXT	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Databases

The Databases object represents relational databases.

The Information Catalog Center DB2 storage table name for this object is XXX.DATABAS.

The following table provides information about the properties of the Databases object.

Table 75. Properties of the Databases object. The MDIS name for this object is Database.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		DatabaseLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		

Table 75. Properties of the Databases object (continued). The MDIS name for this object is Database.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database owner	VARCHAR	80	OWNER	O		OwnerName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database server type	VARCHAR	80	SRVRTYPE	O		ServerType
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName
Database type	VARCHAR	80	DBTYPE	R	3	DatabaseType
Database extended type	VARCHAR	40	DBETYPE	O		DatabaseExtendedType
Database status	VARCHAR	80	DBSTAT	O		DatabaseStatus
Database location	VARCHAR	80	LOCATION	O		ApplicationData
URL to access data	VARCHAR	254	URL	O		ApplicationData
System code page	VARCHAR	10	CODEPAGE	O		ApplicationData
Agent type	VARCHAR	80	AGENTYPE	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Table 75. Properties of the Databases object (continued). The MDIS name for this object is Database.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Note: S = generated by the Information Catalog Center, R = required, O = optional						

Predefined object descriptions: Dimensions within a multidimensional database

The Dimensions within a multidimensional database object represents dimensions within a multidimensional database. A dimension is comprised of members.

The Information Catalog Center DB2 storage table name for this object is XXX.DIMENSION.

The following table provides information about the properties of the dimensions within a multidimensional database object.

Table 76. Properties of the Dimensions within a multidimensional database object. The MDIS name for this object is Dimension.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		DimensionLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database last refreshed	CHAR	26	FRESHDAT	O		ApplicationData

Table 76. Properties of the Dimensions within a multidimensional database object (continued). The MDIS name for this object is Dimension.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName
Using application name	VARCHAR	80	APPLNAME	R	3	ApplicationData
Dimension owner	VARCHAR	80	OWNER	O		OwnerName
Dimension name	VARCHAR	80	DIMENSON	R	4	DimensionName
Dimension class or type	VARCHAR	80	TYPE	O		DimensionType
Total member count	CHAR	10	TOTALCNT	O		DimensionCount
Level count	CHAR	10	LEVELCNT	O		DimensionLevelCount
Application-specific information	VARCHAR	512	APPLDATA	O		ApplicationData
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: DWC Process

The DWC Process object represents a process in the Data Warehouse Center.

The Information Catalog Center DB2 storage table name for this object is XXX.DWCPROC.

The following table provides information about the properties of the Business subject areas object.

Table 77. Properties of the DWC Process object

Property name	Data type	Size	Property short name	Value flag	UUI order
Name	VARCHAR	80	NAME	R	1
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
For further information . . .	VARCHAR	80	RESPNSBL	O	
URL to access data	VARCHAR	254	URL	O	
Timestamp source definition last changed	CHAR	26	SRCDATCF	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptoins: Elements

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 this object type is in the file FLGNYELE.TYP in the \VWSWIN\DGWIN\TYPES directory.

The Information Catalog Center DB2 storage table name for this object type is XXX.ELEMENT.

Table 78. Properties of the Elements object type. The MDIS name for this object type is Element.

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	

Table 78. Properties of the Elements object type (continued). The MDIS name for this object type is Element.

Property name	Data type	Size	Property short name	Value flag	UUI order
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
For further information...	VARCHAR	80	RESPNSBL	O	
Element last refreshed	CHAR	26	FRESHDAT	O	
Database host server name	VARCHAR	80	SERVER	R	1
Database or subsystem name	VARCHAR	80	DBNAME	R	2
Element owner	VARCHAR	80	OWNER	R	3
Dimension or record name	VARCHAR	80	DIMRECNM	R	4
Element name	VARCHAR	80	ELEMNAME	R	5
URL to access data	VARCHAR	254	URL	O	
Data type of element	CHAR	30	DATATYPE	O	
Length of element	CHAR	20	LENGTH	O	

Table 78. Properties of the Elements object type (continued). The MDIS name for this object type is Element.

Property name	Data type	Size	Property short name	Value flag	UUI order
Scale of element	CHAR	5	SCALE	O	
Precision of element	CHAR	5	PRECDIG	O	
Can element be null	CHAR	1	NULLS	O	
Position of element within primary key	CHAR	5	KEYPOSNO	O	
Element position	CHAR	5	POSNO	O	
Element ordinality	CHAR	5	ORDINAL	O	
Timestamp source definition created	CHAR	26	CRTTIME	O	
Timestamp source definition last changed	CHAR	26	SRCDATCF	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Files

The Files object represents a file within a file system.

The Information Catalog Center DB2 storage table name for this object is XXX.FILE.

The following table provides information about the properties of the Files object.

Table 79. Properties of the Files object. The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		RecordLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short Description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long Description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName
File owner	VARCHAR	80	OWNER	R	3	OwnerName
File path or directory	VARCHAR	254	FILEPATH	R	4	ApplicationData
File filename	VARCHAR	254	FILENAME	R	5	RecordName
File data last refreshed	CHAR	26	FRESHDAT	O		RecordLastRefreshDate
Transformation program last run	CHAR	26	LASTRUN	O		ApplicationData
Transformation program run frequency	VARCHAR	80	RUNFREQ	O		RecordUpdateFrequency
Transformation program type	VARCHAR	32	SOURCE	O		ApplicationData

Table 79. Properties of the Files object (continued). The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Partial or full file copy/update	CHAR	1	COPYCOMP	O		ApplicationData
Copied/updated data is in a consistent state	CHAR	1	CONSIST	O		ApplicationData
Transformation program last changed	CHAR	26	PGMGEND	O		ApplicationData
Transformation program last compiled	CHAR	26	PGMCOMP	O		ApplicationData
File class or type	VARCHAR	80	TYPE	O		RecordType
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: IMS database definitions (DBD)

The IMS database definition (DBD) object represents IMS database definitions.

The Information Catalog Center DB2 storage table name for this object is XXX.IMSDBD.

he following table provides information about the properties of the IMD database definitions (DBD) object.

Table 80. Properties of the IMS database definitions (DBD) object. The MDIS name for this object is Database.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		DatabaseLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
Database last refreshed	CHAR	26	FRESHDAT	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database owner	VARCHAR	80	OWNER	O		OwnerName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database server type	VARCHAR	80	SRVRTYPE	O		ServerType
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName
Database type	VARCHAR	80	DBTYPE	R	3	DatabaseType
Database extended type	VARCHAR	40	DBETYPE	O		ApplicationData
Database status	VARCHAR	80	DBSTAT	O		DatabaseStatus

Table 80. Properties of the IMS database definitions (DBD) object (continued). The MDIS name for this object is Database.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
IMS access method	VARCHAR	80	IMSACC	O		ApplicationData
Operating system access method	VARCHAR	80	OSACC	O		ApplicationData
Shared index names	VARCHAR	320	SHRINDEX	O		ApplicationData
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: IMS program control blocks (PCB)

The IMS program control block object represents IMS program control blocks.

The Information Catalog Center DB2 storage table name for this object is XXX.IMSPCB.

The following table provides information about the properties of the IMS program control blocks (PCB) object.

Table 81. Properties of the IMS program control blocks (PCB) object. The MDIS name for this object is Subschema.

Property name	Data type	Size	Product short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		

Table 81. Properties of the IMS program control blocks (PCB) object (continued). The MDIS name for this object is Subschema.

Property name	Data type	Size	Product short name	Value flag	UUI order	Maps to MDIS name:
Name	VARCHAR	80	NAME	R		SubschemaLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName
PCB name	VARCHAR	80	PCBNAME	R	3	SubschemaName
PCB owner	VARCHAR	80	OWNER	O		OwnerName
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: IMS program specification blocks (PSB)

The IMS predefined program block object represents IMS program specification blocks.

The Information Catalog Center DB2 storage table name for this object is XXX.PSB.

Table 82. Properties of the IMS program specification blocks (PSB) object. The MDIS name for this object is Subschema.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		DatabaseLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database server type	VARCHAR	80	SRVRTYPE	O		ServerType
Database type	VARCHAR	80	DBTYPE	R	3	DatabaseType
Database extended type	VARCHAR	40	DBETYPE	O		ApplicationData
Database status	VARCHAR	80	DBSTAT	O		DatabaseStatus
PSB name	VARCHAR	80	PSBNAME	R	2	DatabaseName
PSB owner	VARCHAR	80	OWNER	O		OwnerName
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated

Table 82. Properties of the IMS program specification blocks (PSB) object (continued). The MDIS name for this object is Subschema.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: IMS segments

The IMS segments object represents IMS segments.

The Information Catalog Center DB2 storage table name for this object is XXX.IMSSEG.

The following table provides information about the properties of the IMS segments object.

Table 83. Properties of the IMS segments object. The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		RecordLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short Description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long Description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
Segment last refreshed	CHAR	26	FRESHDAT	O		RecordLastRefreshDate
For further information...	VARCHAR	80	RESPNSBL	O		ContactName

Table 83. Properties of the IMS segments object (continued). The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Database host server name	VARCHAR	80	SERVER	O		ServerName
Database or subsystem name	VARCHAR	80	DBNAME	R	1	DatabaseName
Segment name	VARCHAR	80	SEGNAME	R	2	RecordName
Segment owner	VARCHAR	80	OWNER	O		OwnerName
Segment type	VARCHAR	80	TYPE	O		RecordType
Segment maximum length	CHAR	5	MAXLEN	O		ApplicationData
Segment minimum length	CHAR	5	MINLEN	O		ApplicationData
Real logical child segment source	CHAR	20	PSEGSRC	O		ApplicationData
Logical parent concatenated key source	CHAR	20	LPCKSRC	O		ApplicationData
Transformation program last run	CHAR	26	LASTRUN	O		ApplicationData
Transformation program run frequency	VARCHAR	80	RUNFREQ	O		RecordUpdateFrequency
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Multidimensional databases

The Multidimensional databases object represents multidimensional databases.

The Information Catalog Center DB2 storage table name for this object is XXX.OLAPMODL.

The following table provides information about the properties of the Multidimensional databases object.

Table 84. Properties of the Multidimensional databases object. The MDIS name for this object is Database.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		DatabaseLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database last refreshed	CHAR	26	FRESHDAT	O		ApplicationData
Database owner	VARCHAR	80	OWNER	O		OwnerName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database server type	VARCHAR	80	SRVRTYPE	O		ServerType
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName

Table 84. Properties of the Multidimensional databases object (continued). The MDIS name for this object is Database.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Database type	VARCHAR	80	DBTYPE	O		DatabaseType
Database extended type	VARCHAR	20	DBETYPE	O		ApplicationData
Database status	VARCHAR	80	DBSTAT	O		DatabaseStatus
Using application name	VARCHAR	80	APPLNAME	R	3	ApplicationData
Application-specific information	VARCHAR	512	APPLDATA	O		ApplicationData
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Records

The Records object represents MDIS Record objects that do not map directly to the “Files” or “Relational tables or views” object types. Records are comprised of elements.

The Information Catalog Center DB2 storage table name for this object is XXX.RECORD.

The following table provides information about the properties of the Records object.

Table 85. Properties of the Records object. The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		RecordLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName
Record owner	VARCHAR	80	OWNER	R	3	OwnerName
Record name	VARCHAR	80	RECNAME	R	4	RecordName
Record data last refreshed	CHAR	26	FRESHDAT	O		RecordLastRefreshDate
Transformation program last run	CHAR	26	LASTRUN	O		ApplicationData
Transformation program run frequency	VARCHAR	80	RUNFREQ	O		RecordUpdateFrequency
Record type	VARCHAR	80	TYPE	O		RecordType
URL to access data	VARCHAR	254	URL	O		ApplicationData

Table 85. Properties of the Records object (continued). The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Relational tables and views

The Relational tables and views object represents tables or views of relational databases.

The Information Catalog Center DB2 storage table name for this object is XXX.TABLES.

The following table provides information about the properties of the Relational tables and views object.

Table 86. Properties of the Relational tables and views object. The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		RecordLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short Description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long Description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData

Table 86. Properties of the Relational tables and views object (continued). The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Catalog remarks	VARCHAR	254	REMARKS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database host server name	VARCHAR	80	SERVER	O		ServerName
Local database alias	CHAR	8	DBALIAS	O		ApplicationData
Database or subsystem name	VARCHAR	80	DBNAME	R	1	DatabaseName
Table owner	VARCHAR	80	OWNER	R	2	OwnerName
Table name	VARCHAR	80	TABLES	R	3	RecordName
Base table owner name	CHAR	30	SRCOWNER	O		ApplicationData
Base table name	CHAR	128	SRCTBNAM	O		ApplicationData
Table data last refreshed	CHAR	26	FRESHDAT	O		RecordLastRefreshDate
Transformation program run mode	CHAR	30	RUNMODE	O		ApplicationData
Transformation program last run	CHAR	26	LASTRUN	O		ApplicationData
Transformation program run frequency	VARCHAR	80	RUNFREQ	O		RecordUpdateFrequency
Transformation program type	VARCHAR	32	SOURCE	O		ApplicationData
Partial or full table copy/update	CHAR	1	COPYCOMP	O		ApplicationData
Copied/updated data is in a consistent state	CHAR	1	CONSIST	O		ApplicationData

Table 86. Properties of the Relational tables and views object (continued). The MDIS name for this object is Record.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Catalog refresh / update frequency	VARCHAR	80	REFRESH	O		ApplicationData
Transformation program last changed	CHAR	26	PGMGEND	O		ApplicationData
Transformation program last compiled	CHAR	26	PGMCOMP	O		ApplicationData
Table type	VARCHAR	80	TYPE	O		RecordType
Definition represents a view	CHAR	1	TABLVIEW	O		ApplicationData
Internal name of table	CHAR	18	IDSINAME	O		ApplicationData
Table is used as a dimension table	CHAR	1	IDSDIM	O		ApplicationData
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Star Schemas

The Star Schemas object represents relational data.

The Information Catalog Center DB2 storage table name for this object is XXX.STARSCHM.

The following table provides information about the properties of the Business subject areas object.

Table 87. Properties of the Star Schemas object

Property name	Data type	Size	Property short name	Value flag	UUI order
Name	VARCHAR	80	NAME	R	1
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
For further information . . .	VARCHAR	80	RESPNSBL	O	
URL to access data	VARCHAR	254	URL	O	
Timestamp source definition last changed	CHAR	26	SRCDATCF	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Subschemas

The Subschemas object represents logical groupings of records within a database.

The Information Catalog Center DB2 storage table name for this object is XXX.SUBSCHEM.

The following table provides information about the properties of the Subschemas object.

Table 88. Properties of the Subschemas object. The MDIS name for this object is Subschema.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		SubschemaLongName

Table 88. Properties of the Subschemas object (continued). The MDIS name for this object is Subschema.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
For further information...	VARCHAR	80	RESPNSBL	O		ContactName
Database host server name	VARCHAR	80	SERVER	R	1	ServerName
Database or subsystem name	VARCHAR	80	DBNAME	R	2	DatabaseName
Subschema owner	VARCHAR	80	OWNER	O		OwnerName
Subschema name	VARCHAR	80	SSNAME	R	3	SubschemaName
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Transformations

The Transformations object represents expressions or logic used to populate columns of data within the target relational 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 Information Catalog Manager DB2 storage table name for this object type is XXX.FILTER.

The following table provides information about the properties of the Transformations object.

Table 89. Properties of the Transformations object. The MDIS name for this object is Relationship.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Object type identifier	CHAR	6	OBJTYPID	S		
Instance identifier	CHAR	10	INSTIDNT	S		
Name	VARCHAR	80	NAME	R		RelationshipLongName
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S		
Last Changed By	CHAR	8	UPDATEBY	S		
Short description	VARCHAR	250	SHRTDESC	O		BriefDescription
Long description	LONG VARCHAR	32700	LONGDESC	O		LongDescription
Actions	VARCHAR	254	ACTIONS	O		ApplicationData
Transformation program name	VARCHAR	80	FPNAME	R	1	ApplicationData
Transformation identifier	VARCHAR	254	FIDENT	R	2	RelationshipName
Transformation class or type	VARCHAR	80	TYPE	R	3	RelationshipType
Source column/field name, expression or parameters	LONG VARCHAR	32700	FEXPRESS	O		RelationshipExpression

Table 89. Properties of the Transformations object (continued). The MDIS name for this object is Relationship.

Property name	Data type	Size	Property short name	Value flag	UUI order	Maps to MDIS name:
Database host server name	VARCHAR	80	SERVER	O		ServerName
Transformation owner	VARCHAR	80	OWNER	O		OwnerName
Source sequence	CHAR	5	SRCSEQ	O		SourceSequenceOrder
Transformation ordinality	CHAR	5	ORDINAL	O		RelationshipOrdinality
Transformation bi-directionality	CHAR	1	DIRECT	O		RelationshipBidirectional
URL to access data	VARCHAR	254	URL	O		ApplicationData
Timestamp source definition created	CHAR	26	CRTTIME	O		DateCreated, TimeCreated
Timestamp source definition last changed	CHAR	26	SRCDATCF	O		DateUpdated, TimeUpdated
For further information...	VARCHAR	80	RESPNSBL			ContactName

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Elemental category

The Elemental category of object types for the Information Catalog Center contains the following object types:

- Audio clips
- Charts
- Documents
- Images or graphics
- Internet documents
- Lotus approach queries
- Presentations
- Spreadsheets

- Text-based reports
- Video clips

Predefined object descriptions: Audio clips

The Audio clips object represents files that contain audio information. These objects might represent electronic (AUD files) or printed (for example, CDs, tapes) audio information.

The Information Catalog Center DB2 storage table name for this object is XXX.AUDIO.

The following table provides information about the properties of the Audio clips object.

Table 90. Properties of the Audio clips object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Audio clip filename	VARCHAR	254	FILENAME	R	1
Audio clip class or type	VARCHAR	80	TYPE	R	2
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Charts

The Charts object represents either printed or electronic charts.

The Information Catalog Center DB2 storage table name for this object is XXX.CHARTS.

he following table provides information about the properties of the Charts object.

Table 91. Properties of the Charts object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Chart title	VARCHAR	254	TITLE	O	
Chart publication date	CHAR	26	RPRTDATE	O	
Chart presentation format	VARCHAR	80	RPRTFRMT	O	
Chart presentation requirements	VARCHAR	254	DPPRESNT	O	
Chart owner	VARCHAR	80	OWNER	O	
Chart filename	VARCHAR	254	FILENAME	R	1

Table 91. Properties of the Charts object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Chart class or type	VARCHAR	80	TYPE	R	2
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Documents

The Documents object represents books and technical papers. These publications might be printed or electronic, found locally or within a library.

The Information Catalog Center DB2 storage table name for this object is XXX.DOCS.

The following table provides information about the properties of the Documents object.

Table 92. Properties of the Documents object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Document author	VARCHAR	80	AUTHOR	R	1

Table 92. Properties of the Documents object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Document location	VARCHAR	254	LOCATION	R	2
Document filename	VARCHAR	254	FILENAME	R	3
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Images or graphics

The Images or graphics object represents graphic images, such as bitmaps.

The Information Catalog Center DB2 storage table name for this object is XXX.IMAGES.

The following table provides information about the properties of the Images or graphics object.

Table 93. Properties of the Images or graphics object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	

Table 93. Properties of the Images or graphics object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Image filename	VARCHAR	254	FILENAME	R	1
Image class or type	VARCHAR	80	TYPE	R	2
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Internet documents

The Internet documents object represents Web sites and other documents on the Internet that might be of interest.

The Information Catalog Center DB2 storage table name for this object is XXX.INTERNET.

The following table provides information about the properties of the Internet documents object.

Table 94. Properties of the Internet documents object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	

Table 94. Properties of the Internet documents object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
URL to access data	VARCHAR	254	URL	R	1
Local filename	VARCHAR	254	FILENAME	R	2
Internet document class or type	VARCHAR	80	TYPE	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Lotus Approach queries

Represents Lotus Approach queries for available use with your organization's data.

The Information Catalog Center DB2 storage table name for this object is XXX.APPROACH.

The following table provides information about the properties of the Lotus Approach queries object.

Table 95. Properties of the Lotus Approach queries object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	

Table 95. Properties of the Lotus Approach queries object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Approach object filename	VARCHAR	254	FILENAME	R	1
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Presentations

The Presentations object represents printed or electronic presentations. These presentations might include product, customer, quality, and status presentations.

The Information Catalog Center DB2 storage table name for this object is XXX.PRESENT.

The following table provides information about the properties of the Presentations object.

Table 96. Properties of the Presentations object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	

Table 96. Properties of the Presentations object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Presentation filename	VARCHAR	254	FILENAME	R	1
Presentation class or type	VARCHAR	80	TYPE	O	
Presentation script	VARCHAR	254	SCRIPTFN	O	
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Spreadsheets

The Spreadsheets object represents desktop spreadsheets (for example, Lotus 1-2-3 or Microsoft Excel spreadsheets).

The Information Catalog Center DB2 storage table name for this object is XXX.SSHEETS.

The following table provides information about the properties of the Spreadsheets object.

Table 97. Properties of the Spreadsheets object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	

Table 97. Properties of the Spreadsheets object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Actions	VARCHAR	254	ACTIONS	O	
Spreadsheet class or type	VARCHAR	80	TYPE	O	
Spreadsheet filename	VARCHAR	254	FILENAME	R	1
Spreadsheet bitmap <captured> filename	VARCHAR	254	BITMAP	O	
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Text-based reports

The Text-based reports object represents either printed or electronic reports.

The Information Catalog Center DB2 storage table name for this object is XXX.REPORTS.

he following table provides information about the properties of the Text-based reports object.

Table 98. Properties of the Text-based reports object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	

Table 98. Properties of the Text-based reports object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Report title	VARCHAR	254	TITLE	R	
Report publication date	CHAR	26	RPRTDATE	O	
Report presentation format	VARCHAR	80	RPRTFRMT	O	
Report presentation requirements	VARCHAR	254	DPPRESNT	O	
Report owner	VARCHAR	80	OWNER	O	
Report filename	VARCHAR	254	FILENAME	R	1
Report class or type	VARCHAR	80	TYPE	R	2
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Video clips

The Video clips object represents files that contain video information. These objects might represent electronic (AVI files) or printed (for example, video tapes or laser disks) video information.

The Information Catalog Center DB2 storage table name for this object is XXX.VIDEO.

The following table provides information about the properties of the Video clips object.

Table 99. Properties of the Video clips object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Video clip filename	VARCHAR	254	FILENAME	R	1
Video clip class or type	VARCHAR	80	TYPE	R	2
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Contact category and People to contact

The Contact category contains the People to contact object.

The People to contact object identifies a person or group that is responsible for objects within the information catalog.

The Information Catalog Center DB2 storage table name for this object is XXX.CONTACT.

The following table provides information about the properties of the People to contact object.

Table 100. Properties of the People to contact object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	1
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Contact's responsibility	VARCHAR	254	RESPONSE	R	2
Contact's phone number	CHAR	15	PHONE	R	
Contact's e-mail address	VARCHAR	254	EMAIL	R	
Contact's picture filename	VARCHAR	254	FILENAME	O	
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Dictionary category and Glossary entries

The Dictionary category contains the Glossary entries object.

The Glossary entries object represents definitions for terms used in the information catalog.

The Information Catalog Center DB2 storage table name for this object is XXX.GLOSSARY.

The following table provides information about the properties of the Glossary entries object.

Table 101. Properties of the Glossary entries object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	1
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Keywords	VARCHAR	254	KEYWORD	O	
Context of glossary definition	CHAR	32	CONTEXT	O	
Filename containing glossary definition	VARCHAR	254	FILENAME	O	
Glossary class or type	VARCHAR	80	TYPE	O	
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Support category

The Support category contains the following objects:

- Information Catalog Center news
- Online news services
- Online publications

The **Information Catalog Center news** object contains information about changes to the information catalog.

The Information Catalog Center DB2 storage table name for this object is XXX.DGNEWS.

The following table provides information about the properties of the Information Catalog Center news object.

Table 102. Properties of the Information Catalog Center news object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	1
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
News item date	CHAR	26	NEWSDATE	R	
News clip	VARCHAR	254	ABSTRACT	R	
Full news item	LONG VARCHAR	32700	NEWSITEM	O	
URL to access data	VARCHAR	254	URL	O	

Table 102. Properties of the Information Catalog Center news object (continued)

Property name	Data type	Size	Property short name	Value flag	UUI order
Note: S = generated by the Information Catalog Center, R = required, O = optional					

The **Online news services** object represents news and information services that can be accessed online.

The Information Catalog Center DB2 storage table name for this object is XXX.OLNEWS.

The following table provides information about the properties of the Online news services object.

Table 103. Properties of the Online news services object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	1
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Service name	VARCHAR	254	SERVNAME	R	
URL to access data	VARCHAR	254	URL	O	
Note: S = generated by the Information Catalog Center, R = required, O = optional					

The **Online publications** object represents publications and other documents that can be accessed with online services.

The Information Catalog Center DB2 storage table name for this object is XXX.OLPUBS.

The following table provides information about the properties of the Online publications object.

Table 104. Properties of the Online publications object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	1
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Short description	VARCHAR	250	SHRTDESC	O	
Long description	LONG VARCHAR	32700	LONGDESC	O	
Actions	VARCHAR	254	ACTIONS	O	
Service name	VARCHAR	254	SERVNAME	R	
URL to access data	VARCHAR	254	URL	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Program category

The Program category can only contain the *Programs* object.

The *Programs* object is created when an information catalog is created. In the sample information catalog, DGV5SAMP, the *Programs* object is named *Programs that can be invoked from Information Catalog Center objects*.

Used to define an application that is capable of processing a particular object.

The Information Catalog Center DB2 storage table name for this object is XXX.GLOSSARY.

The following table provides information about the properties of the *Programs that can be invoked from Information Catalog Center objects* object.

Table 105. Properties of the "Programs that can be invoked from Information Catalog Center objects" object

Property name [1]	Data type	Size	Property short name	Value flag [2]	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Class	CHAR	25	UUICLASS	R	1
Qualifier 1	VARCHAR	48	UUIQUAL1	R	2
Qualifier 2	VARCHAR	48	UUIQUAL2	R	3
Qualifier 3	VARCHAR	48	UUIQUAL3	R	4
Identifier	VARCHAR	70	UUIDENT	R	5
Object type this program handles	CHAR	8	HANDLES	O	
Start by invoking	VARCHAR	250	STARTCMD	R	
Parameter list is	VARCHAR	1800	PARMLIST	O	
Short description	VARCHAR	250	SHRTDESC	O	

Note:

1. Descriptions and examples of the required properties are in *Information Catalog Center Administration Guide.*
2. S = generated by the Information Catalog Center, R = required, O = optional

Predefined object descriptions: Attachment category

The Attachment category can contain only the Comments object.

The Comments object is created when an information catalog is created.

The comments object is used to comment on other objects in the information catalog.

The following table provides information about the properties of the Comments object.

Table 106. Properties of the Comments object

Property name	Data type	Size	Property short name	Value flag	UUI order
Object type identifier	CHAR	6	OBJTYPID	S	
Instance identifier	CHAR	10	INSTIDNT	S	
Name	VARCHAR	80	NAME	R	1
Last Changed Date and Time	TIMESTAMP	26	UPDATIME	S	
Last Changed By	CHAR	8	UPDATEBY	S	
Creator	CHAR	8	CREATOR	R	2
Creation time stamp	TIMESTAMP	26	CREATSTP	R	3
Status	CHAR	80	STATUS	O	
Actions	VARCHAR	250	ACTIONS	O	
Extra Information	VARCHAR	80	EXTRA	O	
Long Description	LONG VARCHAR	32700	LONGDESC	O	

Note: S = generated by the Information Catalog Center, R = required, O = optional

Chapter 6. Tag language

The Information Catalog Center tag language allows you to format your descriptive data so that you can imort it into your information catalog. The tag language tells the Information Catalog Center what to do with the descriptive data that it imports. The Information Catalog Center 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 107shows the tags in the tag language and the actions that these tags perform.

Tag language

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

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

Table 107. Information Catalog Center tags

Task	Tag names
Identify action to be taken on input data	ACTION.OBJINST ACTION.OBJTYPE ACTION.RELATION ACTION.RELTYPE

Table 107. Information Catalog Center tags (continued)

Task	Tag names
Describe data to the information catalog	OBJECT PROPERTY INSTANCE RELTYPE RELATIONTYPE CONREL
Identify when changes are committed and where check point occurs	COMMIT
Identify user comments	COMMENT
Format data	NL TAB

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

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

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

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

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

Related reference:

- "NL" on page 187
- "OBJECT" on page 187
- "TAB" on page 199
- "COMMIT" on page 179
- "COMMENT" on page 178
- "INSTANCE" on page 181
- "ACTION.OBJINST" on page 169
- "ACTION.OBJTYPE" on page 173
- "PROPERTY" on page 193
- "ACTION.RELATION" on page 176
- "RELTYPE" in the *Information Catalog Center Administration Guide*
- "ACTION.RELTYPE" in the *Information Catalog Center Administration Guide*
- "RELATIONTYPE" on page 197

How the Information Catalog Center reads tag language files

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

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

Table 108. Keyword values enabled for DBCS

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

All user-defined property values can use DBCS characters.

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

Table 109. Keyword values enabled for DBCS blank characters

Tag name	Keywords
ACTION	OBJTYPE OBJINST RELATION RELTYPE
OBJECT	All keywords
PROPERTY	All keywords
RELTYPE	All keywords

Table 109. Keyword values enabled for DBCS blank characters (continued)

Tag name	Keywords
RELATIONTYPE	All keywords
COMMIT	CHKPID
INSTANCE	*UI_name* *or* *name*

Related reference:

- "Tag language" on page 165

Rules for writing tag language files

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

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

  ```
  :ACTION.OBJINST.
  ```

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

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

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

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

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

Related reference:

- "Tag language" on page 165

ACTION.OBJINST

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

Context

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

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

Syntax

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

Options

The following options are valid for ACTION.OBJINST:

ADD
DELETE
DELETE_TREE_ALL
DELETE_TREE_REL
MERGE
UPDATE

ACTION.OBJINST(ADD)

Adds an object.

Context:

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

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

Figure 15. Using the ACTION.OBJINST tag when adding objects

Rules:

- The object must not already exist.
- Both the OBJECT tag and the INSTANCE tag must follow the ACTION.OBJINST(ADD) tag.
  - The OBJECT tag identifies the object type for the new object.

  - The INSTANCE tag specifies the property values for the new object.
- One or more INSTANCE tags can follow a single OBJECT tag, if the objects are for the same object type.
- One or more sets of an OBJECT tag with INSTANCE tags can follow an ACTION.OBJINST(ADD) tag to describe objects of different object types to add.

ACTION.OBJINST(DELETE)

Deletes an object.

Context:

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

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

Figure 16. Using the ACTION.OBJINST tag when deleting objects

Rules:

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

ACTION.OBJINST(DELETE_TREE_ALL)

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

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

Context:

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

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

Figure 17. Using the ACTION.OBJINST tag when deleting Grouping category objects and contained objects

Rules:

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

ACTION.OBJINST(DELETE_TREE_REL)

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

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

Context:

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

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

Figure 18. Using the ACTION.OBJINST tag when deleting Grouping category objects and relationships

Rules:

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

ACTION.OBJINST(MERGE)

Searches for the input object's Unique Identifier in the information catalog to see whether the input object exists.

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

Context:

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

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

Figure 19. Using the ACTION.OBJINST tag when merging objects

Rules:

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

ACTION.OBJINST(UPDATE)

Updates the value of an object.

Context:

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

Figure 20. Using the ACTION.OBJINST tag when updating objects

Rules:

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

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

Related reference:

- "OBJECT" on page 187
- "INSTANCE" on page 181
- "Tag language" on page 165

ACTION.OBJTYPE

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

Context

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

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

Syntax

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

Options

The following options are valid with ACTION.OBJTYPE:

ADD
APPEND
DELETE

DELETE_EXT
MERGE
UPDATE

ACTION.OBJTYPE(ADD)
Creates the object type.

Context:

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

Figure 21. Using the ACTION.OBJTYPE tag when adding object types

Rules:

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

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

Context:

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

Figure 22. Using the ACTION.OBJTYPE tag when adding properties to object types

Rules:

- The object type must exist.
- The property being appended must not exist.
- Do not assign the property a UUISEQ value other than 0 (the default). Appended properties must be optional with NULLS(Y) and cannot be part of the UI.
- An OBJECT tag and one or more PROPERTY tags must immediately follow the ACTION.OBJTYPE(APPEND) tag.
  - The OBJECT tag identifies the object type being appended.

– Each PROPERTY tag defines a property being appended.

ACTION.OBJTYPE(DELETE)

Deletes the object type.

Context:

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

Figure 23. Using the ACTION.OBJTYPE tag when deleting object types

Rules:

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

ACTION.OBJTYPE(DELETE_EXT)

Deletes the object type and objects of that object type.

Context:

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

Figure 24. Using the ACTION.OBJTYPE tag when deleting object types and all objects of that type

Rules:

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

ACTION.OBJTYPE(MERGE)

Checks the information catalog for the input object type name to see if the object type exists.

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

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

Context:

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

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

Figure 25. Using the ACTION.OBJTYPE tag when merging object types

Rules:

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

ACTION.OBJTYPE(UPDATE)

Changes an object-type external name and ICON file information.

Context:

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

Figure 26. Using the ACTION.OBJTYPE tag when updating object types

Rules:

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

Related reference:

- "OBJECT" on page 187
- "PROPERTY" on page 193
- "Tag language" on page 165

ACTION.RELATION

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

Context

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

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

Syntax

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

Options

The following options are valid with ACTION.RELATION:

ADD
DELETE

ACTION.RELATION(ADD)

Defines an ATTACHMENT, CONTACT, DICTIONARY, SUPPORTED, CONTAINS, INPUT, OUTPUT, CASCADE, LINKED, or user defined relationship.

Context:

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

Figure 27. Using the ACTION.RELATION tag when adding relationships

Rules:

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

ACTION.RELATION(DELETE)

Deletes a relationship.

Context:

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

Figure 28. Using the ACTION.RELATION tag when deleting relationships

Rules:

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

Related reference:

- "INSTANCE" on page 181
- "RELATIONTYPE" on page 197
- "Tag language" on page 165

COMMENT

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

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

Syntax

:COMMENT.*your comments*

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

Figure 29. Example of a COMMENT tag

Rules

- You cannot place a COMMENT tag between another tag and its keywords or between keywords.

- The comment text must not contain any Information Catalog Center tags (for example `:ACTION.`), because each tag ends either at the end of the file or at the beginning of the next valid tag.

Related reference:

- "Tag language" on page 165

COMMIT

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

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

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

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

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

Context

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

Syntax

:COMMIT.CHKPID(*checkpt_id***)**

```
:COMMIT.CHKPID(Added_relationships)
```

Figure 30. Example of a COMMIT tag

Keywords

CHKPID
Required keyword.

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

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

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

The maximum length of *checkpt_id* is 26 characters.

checkpt_id is not case-sensitive.

Rules

Specify a COMMIT tag when the data is consistent.

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

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

Related reference:

- "Tag language" on page 165

Tag language syntax diagrams: DISKCNTL

In your tag language file, the tags and keywords must be coded 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).

The DISKCNTL tag 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 31. 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

Context

This tag is required following:

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

Syntax

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

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

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

Context:

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

Figure 32. Using the INSTANCE tag when adding objects

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

Figure 33. Using the INSTANCE tag when merging objects

Keywords:

short_name
Identifies each property by its short name. If an INSTANCE tag has multiple short names associated with it, use only one INSTANCE tag followed the short names as shown in Figure 33.

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

Rules:

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

ACTION.OBJINST(DELETE) or ACTION.OBJINST(DELETE_TREE_ALL) or ACTION.OBJINST(DELETE_TREE_REL)

Deleting an object

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

Context:

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

Figure 34. Using the INSTANCE tag when deleting objects

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

Figure 35. Using the INSTANCE tag when deleting Grouping category objects and contained objects

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

Figure 36. Using the INSTANCE tag when deleting Grouping category objects and relationships

Keywords:

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

SOURCEKEY must be the first keyword of the INSTANCE tag.

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

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

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

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

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

Context:

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

Figure 37. Using the INSTANCE tag when updating objects

Keywords:

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

SOURCEKEY must be the first keyword of the INSTANCE tag.

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

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

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

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

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

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

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

ACTION.RELATION(ADD) or ACTION.RELATION(DELETE)

Adding or deleting relationships

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

Context:

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

Figure 38. Using the INSTANCE tag when adding relationships

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

Figure 39. Using the INSTANCE tag when deleting relationships

Keywords:

SOURCEKEY

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

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

SOURCEKEY must be the first keyword of the INSTANCE tag.

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

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

TARGETKEY must be the second keyword of the INSTANCE tag.

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

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

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

You must separate each *UI_short_name*(*value*) and *short_name*(*value*) pair with a blank, as shown in Figure 40.

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

Figure 40. Example of an INSTANCE tag with several short names

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

Related reference:

- "ACTION.OBJINST" on page 169
- "ACTION.RELATION" on page 176
- "Tag language" on page 165

NL

Specifies a new line within a property value.

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

Syntax

```
:NL.
```

Rules

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

Related reference:

- "Tag language" on page 165

OBJECT

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

Context

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

Syntax

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

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

ACTION.OBJTYPE(ADD) or ACTION.OBJTYPE(MERGE)

Adding or merging object types

Context:

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

Figure 41. Using the OBJECT tag when adding object types

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

Figure 42. Using the OBJECT tag when merging object types

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

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

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

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

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

CATEGORY
Specifies which category to which this object type belongs .

Required keyword.

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

category

Specifies an Information Catalog Center object category. This value can be one of the following:

GROUPING
ELEMENTAL
Support
CONTACT
DICTIONARY

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

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

EXTNAME

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

name

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

This name must be unique within related information catalogs.

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

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

DESCRIPTION

A description of the object type. Optional keyword.

description

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

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

PHYNAME

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

Optional keyword.

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

table_name

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

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

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

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

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

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

Optional keyword.

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

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

ACTION.OBJTYPE(APPEND)

Context:

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

Figure 43. Using the OBJECT tag when adding properties to object types

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Identifies a specific object type by its short name.

ACTION.OBJTYPE(DELETE) or ACTION.OBJTYPE(DELETE_EXT)

Deleting an existing object type.

Context:

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

Figure 44. Using the OBJECT tag when deleting object types

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

Figure 45. Using the OBJECT tag when deleting object types and all objects of that type

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Identifies a specific object type by its short name.

ACTION.OBJTYPE(UPDATE)

Updating object type information.

Context:

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

Figure 46. Using the OBJECT tag when updating object types

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

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

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

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

You can update this value.

This name must be unique within related information catalogs.

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

DESCRIPTION
A description of the object type. Optional keyword.

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

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

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

Optional keyword.

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

You can update this value.

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

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

Context:

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

Figure 47. Using the OBJECT tag when adding objects

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

Figure 48. Using the OBJECT tag when merging objects

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

Figure 49. Using the OBJECT tag when updating objects

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

Figure 50. Using the OBJECT tag when deleting objects

Keywords:

TYPE
Specifies the short name of an object type.

Required keyword.

short_name
Identifies a specific object type by its short name.

Related reference:

- "ACTION.OBJINST" on page 169
- "ACTION.OBJTYPE" on page 173
- "Tag language" on page 165

PROPERTY

Defines a property that belongs to an object type.

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

Syntax

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

Context

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

Figure 51. Using the PROPERTY tag when adding object types

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

Figure 52. Using the PROPERTY tag when merging object types

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

Figure 53. Using the PROPERTY tag when adding properties to object types

Keywords

EXTNAME
Specifies a descriptive name for the property.

Required keyword.

name
Specifies an extended descriptive name.

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

DT
Specifies the data type for the property.

Required keyword.

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

I (INTEGER)
4 bytes

S (SMALLINT)
2 bytes

G (BIGINT)
8 bytes

E (DECIMAL)
16 bytes

U (DOUBLE)
8 bytes

R (REAL)
4 bytes

B (BLOB)
0 bytes to 2 gigabytes of bytes

O (CLOB)
0 bytes to 2 gigabytes of characters

C (CHAR)
Up to 254 characters

V (VARCHAR)
Up to 4 000 characters

L (LONG VARCHAR)
Up to 32 700 characters

T (TIMESTAMP)
26 characters, in this format:

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

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

hh.mm.ss.nnnnnn

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

yyyy-mm-dd

DL
Specifies the data length or maximum data length for the property.

Required property.

data_length
The data length or maximum data length for the property. Valid values for *data_length* depend on the *data_type* that is defined for this property:

SHRTNAME
Specifies the property short name.

Required keyword.

short_name
The short name for the property. The *short_name* value can be up to 18 characters long. This value can contain only SBCS characters.

This value is case sensitive.

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

This value cannot be any of the SQL reserved words for the database that is used for the information catalog.

NULLS
Specifies whether a value for the property is required for every object. This value can be specified in uppercase or lowercase.

Required keyword.

Y indicates that this value can be null. When appending a new property with the ACTION.OBJTYPE(APPEND) tag, you must specify NULLS(Y), because appended properties must be optional.

N indicates that a value for this property is required.

UUISEQ
Identifies the properties that are used in the Unique Identifier.

Optional keyword; the default value is 0. The UUISEQ keyword is optional for properties that are not part of the UI. The unique identifier is a set of properties that are defined by the administrator as the key that uniquely identifies each object.

UI_number
Specifies the position of the property in the unique identifier sequence. Valid values are 0 — 16. The value 0 means that the property is not part of the UI. A nonzero value for *UI_number* indicates that the property is part of the UI.

All object types defined in the tag language file must have at least one property that is part of the UI. The unique identifier can consist of up to 16 properties.

At least one property must be defined as part of the UI.

When assigning *UI_number* values to more than one property, the numbers of the unique identifier properties must range from 1 to the number of properties in the UI. For example, if three properties are

defined as part of the UI, the *UI_number* values must be 1, 2, and 3. You cannot skip numbers in the sequence. The *UI_number* values do not need to be in the same order that the properties are specified.

Rules

- You can define the reserved property NAME as part of the unique identifier when you add a new object type or merge object types. Figure 54 shows the general syntax for identifying NAME as an unique identifier property.

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

Figure 54. Example of specifying the NAME property as part of the UI

  Empty parentheses in this figure denote values that you must provide in a tag language file.
- The maximum length of the unique identifier fields is 250 bytes.

Related reference:

- "ACTION.OBJTYPE" on page 173
- "Tag language" on page 165

RELATIONTYPE

Identifies the relationship type to add, delete, or update for a relationship category.

This tag is required immediately following one of these tags:
:ACTION.RELTYPE(ADD)
:ACTION.RELTYPE(DELETE)

Syntax

```
:RELATIONTYPE.TYPE(short_name) EXTNAME(name)
                   CATEGORY(relationship_category)
                   DECSCRIPTION(description)
```

Context

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

Figure 55. Using the RELATIONTYPE tag when adding relationship types.

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

Figure 56. Using the RELATIONTYPE tag when deleting relationship types.

Keywords

TYPE
Specifies the short name of a relationship type.

Required keyword.

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

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

The maximum length for the value is 18 characters.

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

CATEGORY
Specifies the relationship category for the relationship type. Required keyword.

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

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

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

This name must be unique within related information catalogs.

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

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

DESCRIPTION
Specifies a description for the relationship type. Optional keyword.

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

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

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

Related reference:

- "ACTION.RELTYPE" in the *Information Catalog Center Administration Guide*
- "Tag language" on page 165

TAB

Specifies a tab within a property value.

The Information Catalog Center reads only TAB tags that are specified within non-UI property values and ignores all others.

Syntax

```
:TAB.
```

Rules

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

Related reference:

- "Tag language" on page 165

Chapter 7. Tag language file content for the Information Catalog Center

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

Define your additions, changes, and deletions

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

Defining what you want to do

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

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

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

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

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

Defining the information

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

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

Putting it all together

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
:ACTION.RELATION(ADD)
:RELTYPE.TYPE(ATTACHMENT | CONTACT | DICTIONARY | SUPPORTED |
              CONTAINS | INPUT | OUTPUT | CASCADE | LINKED)
```

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

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

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

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

Committing changes to the database

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

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

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

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

Putting comments in the tag language file

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

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

Related reference:

- "ACTION.OBJINST" on page 169

- “ACTION.OBJTYPE” on page 173
- “ACTION.RELATION” on page 176
- “ACTION.RELTYPE” in the *Information Catalog Center Administration Guide*
- “Tag language” on page 165

Part 3. Supplied program and macro reference

Chapter 8. Supplied Data Warehouse Center programs

The Data Warehouse Center supplies the following programs to support integration with the Data Warehouse Center:

- VWPEXUNX
- ISV_Sample

The VWPEXUNX program supplied with the Data Warehouse Center

The VWPEXUNX program remotely issues a command or runs a program. VWPEXUNX runs on Windows NT, Windows 2000, and UNIX®.

If you are running the VWPEXUNX program on Windows NT or Windows 2000, the REXECD program must also be running on the workstation.

Parameters

The following table shows the parameter list for the VWPEXUNX program. The list includes the predefined token for a parameter if one exists.

Table 110. Parameters for VWPEXUNX

Order	Description
1	The remote host name.
2	The remote user ID.
3	The remote program to execute.
4	The remote error file.
5	The remote warning file. If there is no warning file, specify - (the not-applicable symbol).
6	The remote log (summary) file. If there is no log file, specify - (the not-applicable symbol).
7	The remote operating system type. Specify either UNIX, WINNT, or WIN2000.
8	The password type. Specify either PasswordNotRequired, EnterPassword, or GetPassword.

Table 110. Parameters for VWPEXUNX (continued)

Order	Description
9	The password value if the password type is EnterPassword. - (not-applicable symbol) if the password type is PasswordNot Required. The password program if password type is GetPassword. The password program must reside on the agent site that is selected for the step. The program must write a file that contains the password to use in the first line of the file. It must return 0 if it runs correctly.
10	The password program parameters if the password type is GetPassword

The following example shows how to start the VWPEXUNX program from a command prompt. The command must be typed all on one line. The line break shown in this example is not significant.

```
vwpexunx tomari labriejj db2cmd \usr\labriejj\db2cmd.err - -
 UNIX EnterPassword mypass
```

tomari The name of the remote host

labriejj The user ID used to access the remote host

db2cmd The remote program to run

\usr\labriejj\db2cmd.err
The path and name of the remote error file

- No remote warning file exists

- No remote log (summary) file exists

UNIX The remote operating system

EnterPassword
The password type

mypass The password

Return codes

The VWPEXUNX program uses the remote error file to determine the success or failure of the remote command or program:

- If the error file is empty or nonexistent, the VWPEXUNX program returns an error code that indicates success.
- If the error file is not empty, the VWPEXUNX program:
  - Saves the contents of the error file in a temporary file.
  - Returns an error code that indicates failure.

The VWPEXUNX program does not check the contents of the remote error file.

The following table lists the return codes for the VWPEXUNX program.

Table 111. Return codes for the VWPEXUNX program

Return code	Description
0	The program ran successfully.
4	The program ran with a warning. The program could not erase the password file after the password program ran.
8	Parameter error. Too few or too many parameters were supplied to the program, or an invalid value was supplied for a parameter.
16	Internal error. The program detected an internal error, such as the inability to open, create, or write to a temporary file.
48	Environment variable error. The *VWS_LOGGING* environment variable was not set.
52	Get password program error. The program detected a password program error, such as a missing program, an invalid name, or the wrong number of parameters
56	Remote execution error. The program detected a remote execution error, such as the following errors: • An incorrect user ID or password was supplied. • A remote file was not found. • A remote host is not responding. • The supplied user ID is not authorized to create or read the remote file.

Log files

The VWPEXUNX program writes a trace file to the directory that the *VWS_LOGGING* environment variable specifies.

The ISV_Sample programs supplied with the Data Warehouse Center

The ISV_Sample program reads metadata from ODBC data sources and generates Data Warehouse Center objects from the metadata. The ISV_Sample program runs on Windows.

The following table shows the parameter list for the ISV_Sample program.

No predefined tokens exist for the parameters.

Table 112. Parameters for ISV_Sample

Order	Description
1	ODBC DSN from which to extract metadata
2	ODBC user ID
3	ODBC password

The following example shows how to start the ISV_Sample program in C++:

```
ISV_Sample SAMPLE labriejj mypass
```

The following example shows how to start the ISV_Sample program in Java:

```
java db2_vw.ISV_sample SAMPLE labriejj mypass
```

SAMPLE The ODBC DSN from which to read metadata

labriejj The user ID used to access the ODBC DSN

mypass The password used to access the ODBC DSN

The ISV_Sample program uses the ISV_VWP program. Steps call the ISV_VWP program to write the input parameters to an output file.

Do not modify the Java sample source code and replace the existing classes in the Data Warehouse Center package. Either subclass or rename the Java sample classes to use in ISV applications.

Chapter 9. Net.Data® macros

The Information Catalog Center for the Web uses Net.Data® macros to display data on the Web and search for data in a database. If you are familiar with Net.Data and its macros, you can customize these macros to meet the requirements of your organization.

For example, the Information Catalog Center for the Web requires a user ID and password by default. You can customize the macros to call your own security program instead.

This chapter lists the files that are included with the Information Catalog Center for the Web. For more information about Net.Data and its macros, see the *Net.Data Programming Guide* and *Net.Data Reference Guide.*

Information Catalog Center for the Web files

To work with Information Catalog Center for the Web files, you perform a custom installation of the Administration Client and select Information Catalog Center for the Web. The files are installed in the `x:\sqllib\icuweb` directory.

The file names are lowercase to follow the AIX® naming convention.

The following table lists the Information Catalog Center for the Web files that contain Net.Data macros, which are located in the `x:\sqllib\icuweb\macro` directory.

Table 113. Information Catalog Center Web Net.Data macros

File name	Description
dg_list.mac	Displays the results of a search, tree, or subject call
dg_desc.mac	Displays the results of a description view
dg_frame.mac	Creates the three-frame page
dg_advsearch.mac	Performs an advanced search
dg_comment.mac	Creates or updates a comment
dg_home.mac	Displays the Information Catalog Manager home page
dg_tableviewer.mac	Displays sample data

The following table lists the Information Catalog Center for the Web files that contain Net.Data include files, which are located in the `x:\sqllib\icuweb\macro` directory.

Table 114. Net.Data include files

File name	Description
dg_desc.hti	Include file with common functions for description view
dg_home.hti	Include file with a list of information catalogs to display on the Information Catalog Manager home page
dg_strings.hti	Include file with translatable strings
dg_config.hti	Include file with installation configurable variables
dg_graphics.hti	Include file with graphics look and feel definitions

The following table displays the Information Catalog Center for the Web files that contain HTML, which are located in the `x:\sqllib\icuweb\html` directory.

Table 115. Information Catalog Center for the Web HTML files

File Name	Description
*.htm	Help files

The following table lists the Information Catalog Center for the Web graphic files, which are located in the `x:\sqllib\icuweb\icons` directory.

In addition to the graphics files listed below, you can also create unique icons for any new object type that you create in the Information Catalog Manager. For more information on creating object type icons, see the *Information Catalog Manager Administration Guide*.

Table 116. Information Catalog Center for the Web graphics files

File name	Description
dg_ibmlogo.gif	IBM logo
dg_lgudblogo.gif	Large DB2 logo on Home
dg_smudblogo.gif	Small DB2 logo on header
dg_curve.gif	Small curve joining header and menu
dg_lgappldata.gif	Large Application Data
dg_smappldata.gif	Small Application Data

Table 116. Information Catalog Center for the Web graphics files (continued)

File name	Description
dg_lgapproach.gif	Large Lotus Approach
dg_smapproach.gif	Small Lotus Approach
dg_lgaudio.gif	Large Audio Clips
dg_smaudio.gif	Small Audio Clips
dg_lgcharts.gif	Large Charts
dg_smcharts.gif	Small Charts
dg_lgcolumns.gif	Large Columns
dg_smcolumns.gif	Small Columns
dg_lgcomments.gif	Large Comments
dg_smcomments.gif	Small Comments
dg_lgcontact.gif	Large People to contact
dg_smcontact.gif	Small People to contact
dg_lgdatabas.gif	Large Databases
dg_smdatabas.gif	Small Databases
dg_lgimsdbd.gif	Large IMS database definitions (DBD)
dg_smimsdbd.gif	Small IMS database definitions (DBD)
dg_lgdgnews.gif	Large News
dg_smdgnews.gif	Small News
dg_lgdimenson.gif	Large Dimensions within a multi-dimensional database
dg_smdimenson.gif	Small Dimensions within a multi-dimensional database
dg_lgdocs.gif	Large Documents
dg_smdocs.gif	Small Documents
dg_lgelement.gif	Large Elements
dg_smelement.gif	Small Elements
dg_lgfile.gif	Large Files
dg_smfile.gif	Small Files
dg_lgfilter.gif	Large Transformations
dg_smfilter.gif	Small Transformations
dg_lgglossary.gif	Large Glossary entries
dg_smglossary.gif	Small Glossary entries
dg_lgimages.gif	Large Images or graphics

Table 116. Information Catalog Center for the Web graphics files (continued)

File name	Description
dg_smimages.gif	Small Images or graphics
dg_lginfogrps.gif	Large Business subject areas
dg_sminfogrps.gif	Small Business subject areas
dg_lginternet.gif	Large Internet documents
dg_sminternet.gif	Small Internet documents
dg_lgmember.gif	Large Members within a multidimensional database
dg_smmember.gif	Small ″Members within a multidimensional database
dg_lgolapmodl.gif	Large Multidimensional database
dg_smolapmodl.gif	Small Multidimensional database
dg_lgolnews.gif	Large Online news services
dg_smolnews.gif	Small Online news services
dg_lgolpubs.gif	Large Online news services
dg_smolpubs.gif	Small Online news services
dg_lgiimspcb.gif	Large IMS program control block (PCB)
dg_smimspcb.gif	Small IMS program control block (PCB)
dg_lgpresent.gif	Large Presentations
dg_smpresent.gif	Small Presentations
dg_lgimspsb.gif	Large IMS program specifications (PSB)
dg_smimspsb.gif	Small IMS program specifications (PSB)
dg_lgrecord.gif	Large Records
dg_smrecord.gif	Small Records
dg_lgreports.gif	Large Text-based reports
dg_smreports.gif	Small Text-based reports
dg_lgmsseg.gif	Large IMS segment
dg_smimsseg.gif	Small IMS segment
dg_lgssheets.gif	Large Spreadsheet
dg_smssheets.gif	Small Spreadsheet
dg_lgsubschem.gif	Large Subschemas
dg_smsubschem.gif	Small Subschemas
dg_lgtables.gif	Large Relational tables and views
dg_smtables.gif	Small Relational tables and views

Table 116. Information Catalog Center for the Web graphics files (continued)

File name	Description
dg_lgvideo.gif	Large Video clips
dg_smvideo.gif	Small Video clips
dg_lggrouping.gif	Large Grouping - default category icon
dg_smgrouping.gif	Small Grouping- default category icon
dg_lgelemental.gif	Large Elemental- default category icon
dg_smelemental.gif	Small Elemental- default category icon
dg_lgcontact.gif	Large Contact- default category icon
dg_smcontact.gif	Small Contact- default category icon
dg_lgdictionary.gif	Large Dictionary- default category icon
dg_smdictionary.gif	Small Dictionary- default category icon
dg_lgsupport.gif	Large Support- default category icon
dg_smsupport.gif	Small Support- default category icon
dg_lgattachment.gif	Large Attachment- default category icon
dg_smattachment.gif	Small Attachment- default category icon
dg_collapse.gif	tree - collapse icon
dg_expand.gif	tree - expand icon
dg_lmore.gif	description - long property (more arrow)
dg_clear.gif	clear graphic for spacing

Appendix A. Information Catalog Manager system tables and metadata models

The following tables are defined for Information Catalog Manager system usage.

FLG.ATCHREL table for the Data Warehouse Center

The FLG.ATCHREL table is used to define a relationship between an object instance and a comment.

The RELTYPE, SOURCE, and TARGET columns form the primary key of table.

The RELTYPE column is an index of the table.

The following table provides information about each column found in the FLG.ATCHREL table.

Table 117. FLG.ATCHREL table column properties

Column name	Data type	Description	Nullable	NLS
RELTYPE	CHAR(1)	Relation type: **A** Attachment relation **L** Link relation **M** Comments relation	No	SBCS
SOURCE	CHAR(16)	The FLGID that represents the source object instance.	No	SBCS
TARGET	CHAR(16)	The FLGID that represents the target object instance	No	SBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.CHECKPT table for the Data Warehouse Center

The FLG.CHECKPT table is used by the Import API to restart the import process at a checkpoint.

The table is populated by the Import API. At any time, this table can contain zero to many rows.

The TAGFNAME column is the primary key of table.

The COMMITID, LASTUPDT, and USERID columns are all indexes of the table.

The following table provides information about each column found in the FLG.CHECKPT table.

Table 118. FLG.CHECKPT table column properties

Column name	Data type	Description	Nullable	NLS
TAGFNAME	VARCHAR(240)	The name of the tag language file (without the path information).	No	Both SBCS and DBCS
COMMITID	CHAR(26)	The identifier of the last COMMIT checkpoint. This identifier is supplied by the user in a COMMIT tag placed at appropriate locations in the tag language file. It can be a system timestamp or any series of characters.	No	Both SBCS and DBCS
LASTUPDT	TIMESTAMP	The system timestamp when this entry was either created or updated. The Last Update field will not need padding, because it will always occupy the full 26 bytes.	No	None
USERID	CHAR(8)	The user ID of the information catalog administrator.	No	Both SBCS and DBCS
ENTSAVED	INTEGER	The total number of entries that have been saved in the save area.	No	None
SAVEAREA	LONG VARCHAR	Storage area for a list of object type names. Each object type name is 8 bytes.	No	SBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.COMMENTS table for the Data Warehouse Center

The FLG.COMMENTS table contains all the comments on objects in the Information Catalog Center information catalog.

At any time, this table may contain zero to many rows.

The INSTIDNT column is the primary key of the table.

The NAME, CREATOR, and CREATSTP columns form the unique index of the table.

The NAME, CREATOR, CREATSTP, and UPDATIME columns are indexes of the table.

The following table provides information about each column found in the FLG.COMMENTS table.

Table 119. FLG.COMMENTS table column properties

Column name	Data type	Description	Nullable	NLS
OBJTYPID	CHAR(6)	This six-digit object type ID, generated by the Information Catalog Center, represents a specific object type in the information catalog.	No	SBCS
INSTIDNT	CHAR(10)	The unique instance ID generated by the Information Catalog Center. It is the second part of the FLGID, the 10-digits serial number that will uniquely identify this instance within its own object type.	No	SBCS
NAME	VARCHAR(80)	The name entered by the information catalog user to identify each user-defined object instance.	No	Both SBCS and DBCS
UPDATIME	CHAR(26)	The date and time of the metadata creation or last update. This date is generated by the Information Catalog Center.	Yes	None
UPDATEBY	CHAR(8)	The user ID of the information catalog administrator who last updated the instance.	Yes	Both SBCS and DBCS
CREATOR	CHAR(8)	The creator of the Comments object. The system will set the creator value to the current user ID.	No	Both SBCS and DBCS

Table 119. FLG.COMMENTS table column properties (continued)

Column name	Data type	Description	Nullable	NLS
CREATSTP	CHAR(26)	A timestamp indicating the date and time the Comments object instance was created. This timestamp is supplied by the system when the instance is created.	No	None
STATUS	CHAR(80)	The status of the comment. Users can design their own conventions for this value.	Yes	Both SBCS and DBCS
ACTIONS	VARCHAR(250)	Specifies what action the user should take.	Yes	Both SBCS and DBCS
EXTRA	VARCHAR(80)	Used for extra information.	Yes	Both SBCS and DBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.EXCHANGE table for the Data Warehouse Center

The FLG.EXCHANGE table is used to keep track of the object synchronized between the Information Catalog Center, the Data Warehouse Center, and DB2 OLAP Server™.

This table is populated by the metadata interchange at the time of installation.

The OBJNAME and OBJTYPE columns form the primary key of the table.

The following table provides information about each column found in the FLG.EXCHANGE table.

Table 120. FLG.EXCHANGE table column properties

Column name	Data type	Description	Nullable	NLS
PRODUCT	VARCHAR(40)	The combination of product, version, and release numbers.	No	SBCS
OBJNAME	VARCHAR(200)	The object name, for example, step.	No	Both SBCS and DBCS

Table 120. FLG.EXCHANGE table column properties (continued)

Column name	Data type	Description	Nullable	NLS
IMPDATE	TIMESTAMP	The import timestamp.	No	None
OBJTYPE	CHAR(5)	OBJTYPE can be one of the following values: • IR represents source metadata exchanged • DR represents target metadata • BV represents step metadata • OLAP represents OLAP metadata	No	SBCS
Note: NLS: National Language Support SBCS: Single-byte character set DBCS: Double-byte character set				

FLG.HISTORY table for the Data Warehouse Center

The FLG.HISTORY table is used to keep track of object instances that have been deleted from the Information Catalog Center and the Data Warehouse Center.

The table is populated when the user deletes an object instance and the recording delete history flag is ON. At any time, this table can contain zero to many rows.

The HISSEQ column is the primary key of the table.

The following table provides information about each column found in the FLG.HISTORY table.

Table 121. FLG.HISTORY table column properties

Column name	Data type	Description	Nullable	NLS
HISSEQ	TIMESTAMP	The sequence number of the delete history.	No	None
HISTYPE	INTEGER	The type of the delete history. • A value of 1 in this column indicates a deletion from the information catalog. • A value of 2 in this column indicates a deletion from the Data Warehouse Center.	No	None

Table 121. FLG.HISTORY table column properties (continued)

Column name	Data type	Description	Nullable	NLS
HISTAG	LONG VARCHAR	This column will store the identifier of the object to be deleted.	Yes	Both SBCS and DBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.NAMEINST table for the Data Warehouse Center

The FLG.NAMEINST table contains the name of every object in the information catalog.

The FLGID column is the primary key of the table.

The INSTNAME and TYPENAME columns are indexes of the table.

The following table provides information about each column found in the FLG.NAMEINST table.

Table 122. FLG.NAMEINST table column properties

Column name	Data type	Description	Nullable	NLS
FLGID	CHAR(16)	The 16-character object instance ID.	No	SBCS
TYPENAME	VARCHAR(80)	The external name of the object type.	No	Both SBCS and DBCS
INSTNAME	VARCHAR(80)	The external name of an object instance.	No	Both SBCS and DBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.OBJTYREG table for the Data Warehouse Center

The FLG.OBJTYREG table is used to keep track of all objects and their object types, as well as tables created by the Information Catalog Center.

The OBJTYPID column is the primary key of FLG.OBJTYREG that uniquely identifies an object type in the information catalog and is used as the prefix for all instance IDs.

The columns PTNAME, NAME, and DPNAME are unique index keys of the FLG.OBJTYREG table.

The columns CATEGORY, CREATOR, and UPDATEBY are index keys of the table.

The following table provides information about each column in the FLG.OBJTYREG table.

Table 123. FLG.OBJTYREG table column properties

Column name	Data type	Description	Nullable	NLS
OBJTYPID	CHAR(6)	The six-digit object type ID generated by the Information Catalog Center. The ID represents a specific object type in the information catalog.	No	SBCS
PTNAME	CHAR(30)	The name of the object type. The name is also used as the name of the user's table. The actual size of PTNAME is determined by the value of ENVSIZE on the FLG.PARMS table, which is defined during installation.	No	SBCS
DPNAME	CHAR(8)	The unique object type name within an information catalog.	No	SBCS
NAME	VARCHAR(80)	The external name of this object type.	No	Both SBCS and DBCS
CATEGORY	CHAR(1)	The Information Catalog Center categories: Elemental E, Grouping G, Program P, Contact C, Dictionary D, Support S, and Attachment A.	No	SBCS
CREATOR	CHAR(8)	The user ID of the information catalog administrator who created the object type. It will be blank when the object type is registered. It will also contain a blank after the object type is deleted but before the registration is removed.	Yes	Both SBCS and DBCS

Table 123. FLG.OBJTYREG table column properties (continued)

Column name	Data type	Description	Nullable	NLS
UPDATIME	CHAR(26)	The date and time of the object type that was created or that had its properties extended.	Yes	SBCS
UPDATEBY	CHAR(8)	The user ID of the information catalog administrator who last extended the object type (appended properties).	Yes	Both SBCS and DBCS
LASTINID	INTEGER	The last system-generated instance ID for this object type. This is an internal property, and it will not be visible to the information catalog user. It is accessed and updated by the Create Instance IPI only.	No	None
OBJICON	LONG VARCHAR FOR BIT DATA	The icon bitmap corresponding to the object type.	No	None
Note: NLS: National Language Support SBCS: Single-byte character set DBCS: Double-byte character set				

FLG.OVERDESC table for the Data Warehouse Center

The FLG.OVERDESC table contains all long description properties. Each long description is divided into 3-KB chunks.

The OBJTYPID, INSTIDNT, PHYPRPNM, and SEQNO columns form the primary key of table FLG.OVERDESC.

The following table provides information about each column found in the FLG.OVERDESC table.

Table 124. FLG.OVERDESC table column properties

Column name	Data type	Description	Nullable	NLS
OBJTYPID	CHAR(6)	The six-digit, object type ID generated by the Information Catalog Center, represents a specific object type in the information catalog.	No	SBCS

Table 124. FLG.OVERDESC table column properties (continued)

Column name	Data type	Description	Nullable	NLS
INSTIDNT	CHAR(10)	The unique instance ID generated by the Information Catalog Center. The ID is the second part of the FLGID, the 10-digit portion of the serial number that uniquely identifies this instance within its own object type.	No	SBCS
PHYPRPNM	CHAR(8)	The original property or column name defined by the user.	No	SBCS
SEQNO	SMALLINT	A sequence number to keep track of how many rows reflect the same incoming source.	No	None
ODESC	VARCHAR(3000)	This entry keeps the segments of a long description, which can be up to 32700 bytes, in a smaller and more manageable buffer.	No	Both SBCS and DBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.PARMS table for the Data Warehouse Center

The FLG.PARMS table does not contain metadata. It contains internal, global parameters for the Information Catalog Center. The table is a global storage area for persistent Information Catalog Center parameters such as version, logon message, and code page.

FLG.PARMS stores system parameters. The values in this table are set when you use the Information Catalog Center Create Catalog Utility. You can also use the Information Catalog Center APIs to change the values.

The following table provides information about each column in the FLG.PARMS table.

Table 125. FLG.PARMS table column properties

Column name	Data type	Description	Nullable	NLS
VERSION	CHAR(20)	The version of the information catalog, for example, V1R0M0 or V1R1M0; which is populated at the installation or migration time.	Yes	SBCS

Table 125. FLG.PARMS table column properties (continued)

Column name	Data type	Description	Nullable	NLS
LOGONMSG	VARCHAR(254)	Information Catalog Center logon message, for example, ″Welcome to the Information Catalog Center!″	Yes	Both SBCS and DBCS
CODEPAGE	CHAR(4)	Code page number of the information catalog.	Yes	SBCS
LANGUAGE	CHAR(4)	Language code, for example, ENU (US English). It is loaded from a string file.	Yes	SBCS
DTOKEN	CHAR(1)	The default token of the Information Catalog Center environment used to represent an unspecified data field. This not-applicable symbol is used by the import and export functions. This value is set during installation.	Yes	SBCS
ENVSIZE	SMALL INTEGER	Database server environment size. This value is set during installation, and is used to specify the proper name length for Information Catalog Center tables, columns, and indexes. This value can be 10 for DB2 Universal Database for iSeries, 18 for most other IBM relational databases, and up to a maximum of 30 bytes for non-IBM databases.	Yes	None
LASTYPID	INTEGER	The last system-generated ID for an object type. The ID is accessed and updated by the Create Registration IPI only.	Yes	None
LISTMAX	INTEGER	The maximum number of retrievable objects from a listing or search result.	Yes	None
ISTGROUP	CHAR(8)	The index storage group name for the DB2 Universal Database for OS/390® database.	Yes	SBCS
TSTGROUP	CHAR(8)	The table storage group name for the DB2 Universal Database for OS/390 database.	Yes	SBCS
MDBNAME	CHAR(8)	The DB2 Universal Database for OS/390 database name.	Yes	SBCS
TBSPAC32	CHAR(8)	The 32 KB table space name for the DB2 Universal Database for OS/390 database.	Yes	SBCS
TBSPAC04	CHAR(8)	The 4 KB table space name for DB2 Universal Database for OS/390 database.	Yes	SBCS

Table 125. FLG.PARMS table column properties (continued)

Column name	Data type	Description	Nullable	NLS
PARMFLAG	INTEGER	A flag indicator. **FLG_PARMS_RECORD_DELETE_HISTORY** Records the delete history. **FLG_PARMS_MVS_FOLD_UP** Saves the object values in uppercase in the DB2 Universal Database for OS/390 information catalog. You can search these values in uppercase or lowercase in the Information Catalog Center.	Yes	None
CMTSTAT	VARCHAR(800)	This column stores a list of comments status. Each status is 80 bytes.	Yes	Both SBCS and DBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.PROGRAMS table for the Data Warehouse Center

The FLG.PROGRAMS table is used to keep track of all program objects in the information catalog.

The INSTIDNT column is the primary key of the table FLG.PROGRAMS.

The UUICLASS, UUIQUAL1, UUIQUAL2, UUIQUAL3, and UUIDENT columns form the unique index of table FLG.PROGRAMS.

The NAME, UPDATEBY, UPDATIME, UUICLASS, UUIQUAL1, UUIQUAL2, UUIQUAL3, UUIDENT, and HANDLES columns are indexes of the table.

The following table provides information about each column found in the FLG.PROGRAMS table.

Table 126. FLG.PROGRAMS table column properties

Column name	Data type	Description	Origin	NLS
OBJTYPID	CHAR(6)	The six-digit object type ID, generated by the Information Catalog Center, represents a specific object type.	No	SBCS

Table 126. FLG.PROGRAMS table column properties (continued)

Column name	Data type	Description	Origin	NLS
INSTIDNT	CHAR(10)	The unique instance ID generated by the Information Catalog Center. It is the second part of the FLGID, the 10-digit serial number that uniquely identifies this instance within its own object type.	No	SBCS
NAME	VARCHAR(80)	This name is entered by the information catalog user to identify each user-defined object instance.	No	Both SBCS and DBCS
UPDATIME	CHAR(26)	The date and time of metadata creation or last update. This is generated by the Information Catalog Center.	Yes	SBCS
UPDATEBY	CHAR(8)	The user ID of the information catalog administrator who last updated the instance.	Yes	Both SBCS and DBCS
UUICLASS	CHAR(25)	The part1 name of the universal unique identifier (UUI).	No	Both SBCS and DBCS
UUIQUAL1	VARCHAR(48)	The part2 name of the (UUI).	No	Both SBCS and DBCS
UUIQUAL2	VARCHAR(48)	The part3 name of the (UUI).	No	Both SBCS and DBCS
UUIQUAL3	VARCHAR(48)	The part4 name of the (UUI).	No	Both SBCS and DBCS
UUIDENT	VARCHAR(70)	The part5 name of the (UUI).	No	Both SBCS and DBCS
HANDLES	CHAR(8)	The object type that this program handles.	Yes	SBCS
STARTCMD	VARCHAR(250)	The program name to be invoked. The program can have an extension of .exe, .cmd, .com, or .bat.	No	Both SBCS and DBCS
PARMLIST	VARCHAR(1800)	If a parameter list is required to handle object instances, the value of the parameter is specified by the HANDLES property.	Yes	Both SBCS and DBCS
SHRTDESC	VARCHAR(250)	The short description of the program.	Yes	Both SBCS and DBCS

Table 126. FLG.PROGRAMS table column properties (continued)

Column name	Data type	Description	Origin	NLS
Note:				
NLS: National Language Support				
SBCS: Single-byte character set				
DBCS: Double-byte character set				

FLG.PROPERTY table for the Data Warehouse Center

The FLG.PROPERTY table is used to define a property for an object type. There is one row for each property of each object type defined in this table.

The OBJTYPID column is the index of the table.

The following table provides information about each column found in the FLG.PROPERTY table.

Table 127. FLG.PROPERTY table column properties

Column name	Data type	Description	Nullable	NLS
OBJTYPID	CHAR(6)	System-generated ID that is a unique 6 digits for each object type.	No	SBCS
PHYPRPNM	CHAR(8)	The physical name of the property in the object type. This name will be used to generate the column name in the user's object table.	No	SBCS
PROPNAME	VARCHAR(80)	The external name of this object type property.	No	Both SBCS and DBCS
DATATYPE	CHAR(30)	Property data type, CHAR, VARCHAR, LONG VARCHAR and TIMESTAMP.	No	SBCS
LENGTH	INTEGER	Property length.	No	None
OPTIONS	CHAR(1)	A value flag used to indicate if this field allows null values. **R** Value required (not nullable) **O** Optional value (nullable) **S** System generated value	No	SBCS
UUISEQNO	CHAR(1)	The UUI sequence number of the property in the object type.	Yes	SBCS
PROPSEQ	INTEGER	The sequence number of the property	No	None

Table 127. FLG.PROPERTY table column properties (continued)

Column name	Data type	Description	Nullable	NLS
Note:				
NLS: National Language Support				
SBCS: Single-byte character set				
DBCS: Double-byte character set				

FLG.RELINST table for the Data Warehouse Center

The FLG.RELINST table defines relationships between two objects. The table contains one row for each source-to-target object instance relationship.

The RELTYPE, SOURCE, and TARGET columns form the primary key of the table.

The RELTYPE, SRCCAT, SOURCE, SRCTNAME, SRCINAME, TRGCAT, TARGET, TRGTNAME, and TRGINAME columns are indexes of the table.

The following table provides information about each column found in the FLG.RELINST table.

Table 128. FLG.RELINST table column properties

Column name	Data type	Description	Nullable	NLS
RELTYPE	CHAR(1)	Relation type: **C** Contains **T** Contact	No	SBCS
SRCCAT	CHAR(1)	Category of the source object.	No	SBCS
SOURCE	CHAR(16)	The FLGID that represents the source object instance.	No	SBCS
SRCTNAME	VARCHAR(80)	The external name of the source object type.	No	Both SBCS and DBCS
SRCINAME	VARCHAR(80)	The external name of the source object instance.	No	Both SBCS and DBCS
TRGCAT	CHAR(1)	The category of the target object.	No	SBCS
TARGET	CHAR(16)	The FLGID that represents the target object instance.	No	SBCS

Table 128. FLG.RELINST table column properties (continued)

Column name	Data type	Description	Nullable	NLS
TRGTNAME	VARCHAR(80)	The external name of the target object type.	No	Both SBCS and DBCS
TRGINAME	VARCHAR(80)	The external name of the target object instance.	No	Both SBCS and DBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.USERS table for the Data Warehouse Center

The FLG.USERS table contains a list of all the information catalog administrators and users with special administrative privileges. Unlike most of the other Information Catalog Center store tables, the FLG.USERS table does not contain metadata. It contains definitions of different types of information catalog users and their status.

The USERTYPE and DGUSER columns form the primary key of the table.

The DGUSER column is an index of the table.

The following table provides information about each column found in the FLG.USERS table.

Table 129. FLG.USERS table column properties

Column name	Data type	Description	Nullable	NLS
DGUSER	CHAR(8)	The user ID of the information catalog administrator. The ID is entered at installation.	No	Both SBCS and DBCS
USERTYPE	CHAR(1)	Type of DGUSER. The type can be an information catalog administrator, a user with special update privileges, or a user. This value is set during installation.	No	SBCS

Table 129. FLG.USERS table column properties (continued)

Column name	Data type	Description	Nullable	NLS
ACTIVEKA	CHAR(1)	A flag to indicate the information catalog administrator who is currently logged on to the Information Catalog Center. Only one information catalog administrator can be logged on at a time.	Yes	SBCS

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

FLG.WINICON table for the Data Warehouse Center

The FLG.WINICON table contains the associated Windows icon for each object type.

The OBJTYPID column is the primary key of the table.

The following table provides information about each column found in the FLG.WINICON table.

Table 130. FLG.WINICON table column properties

Column name	Data type	Description	Nullable	NLS
OBJTYPID	CHAR(6)	The six-character object type ID.	No	SBCS
OBJICON	LONG VARCHAR FOR BIT DATA (30000)	The bitmap for the Windows icon.	Yes	None

Note:

NLS: National Language Support

SBCS: Single-byte character set

DBCS: Double-byte character set

Model for Information Catalog Manager system tables

The following illustrations show the relationships between the different Information Catalog Center system tables as well as the object-type tables. For example, a relationship can be a join between two columns. The following Information Catalog Center system tables are not related to the other system tables:

- FLG.PARMS
- FLG.HISTORY
- FLG.USERS
- FLG.EXCHANGE
- FLG.CHECKPT

See the notes following this figure for each numbered relationship.

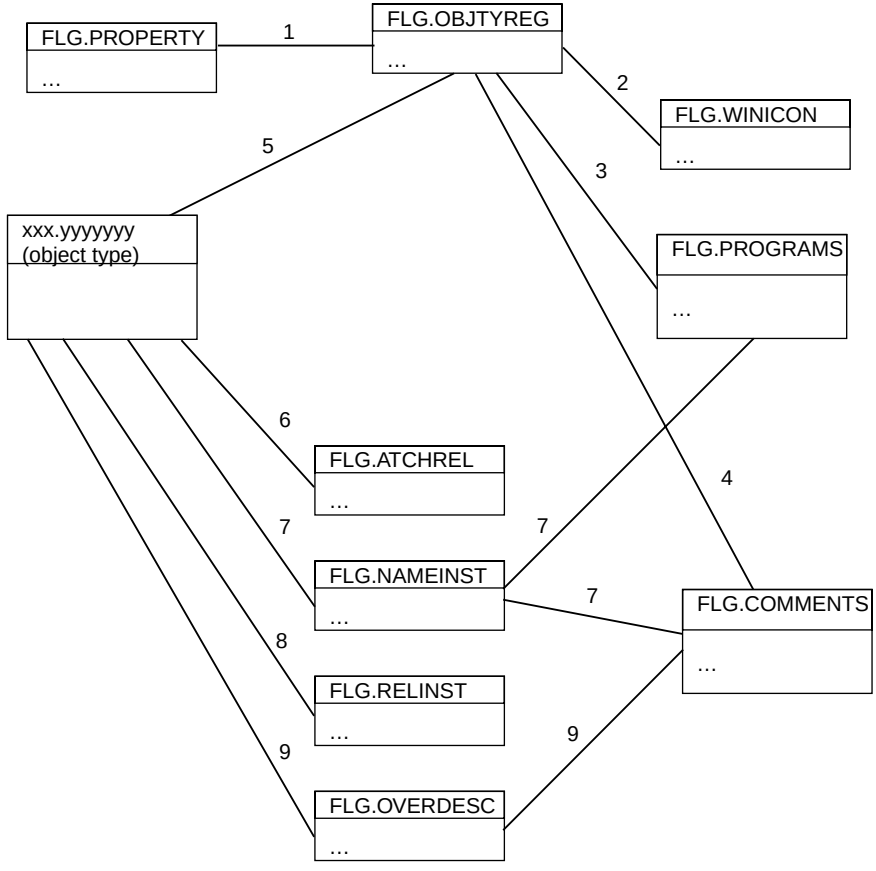


Figure 57. Information Catalog Center system tables

Notes to previous figure

1. The relationship between the two tables exists when the values in the OBJTYPID columns of the tables are equal. The relationship is a join between the two tables based on the OBJTYPID column.
2. The relationship between the two tables exists when the values in the OBJTYPID columns of the tables are equal. The relationship is a join between the two tables based on the OBJTYPID column.
3. The relationship between the two tables exists when the values in the DPNAME and HANDLES columns of the tables are equal. The relationship is a join between the two tables based on the DPNAME and HANDLES columns.
4. The relationship between the tables is derived from the PTNAME and CREATOR columns of the FLG.OBJTYREG table, and the physical name of the FLG.COMMENTS table.

   For example, in the following figure, the first entry in the PTNAME column is COMMENTS, and the first entry in the CREATOR column is FLG. Together these values form the fully qualified FLG.COMMENTS table name.

FLG.OBJTYREG

OBJTYPID	PTNAME	DPNAME	NAME	CATEGORY	CREATOR	...
000001	COMMENTS	COMMENTS	Comments	G	FLG	...
000002	PRESENT	PRESENT	Presentations	E	DGADMIN	...
000003	COLUMNS	COLUMN	Columns or fields in a relational DB	G	DGADMIN	...

FLG.COMMENTS

OBJTYPID	INSTIDNT	Name	UPDATIME	UPDATEBY	SHRTDESC	...	...	...
000001	0000016465	Comment for "My Presentation" object	...	...	...	...	...	...
000001	0000003435	This is a comment for the XYZ presentation	...	...	...	...	...	...
000001	0000064459	this is comment3	...	...	...	...	...	...

DGADMIN.PRESENT

OBJTYPID	INSTIDNT	Name	UPDATIME	UPDATEBY	SHRTDESC	...	...
000002	0000001111	My presentation	...	...	This is a presentation object	...	...
0000021	0000002222	XYZ presentation	...	...	This is another presentation object in the information catalog	...	...

Figure 58. Relationship between table FLG.OBJTYREG and the object type table

5. The relationship between the FLG.OBJTYPREG table and an object type table is derived by concatenating the PTNAME and CREATOR columns of the FLG.OBJTYPREG table. The resulting name is the name of the object type table.

   For example in the previous figure, the second entry in the PTNAME column is PRESENT, and the second entry in the CREATOR column is DGADMIN. Together these values form the fully qualified name DGADMIN.PRESENT.

6. If a relationship is of type A (attaches), the relationship that is stored in the FLG.ATCHREL table is derived by concatenating the object type ID and instance ID of a source table with the object type and instance ID of a target table.

   For example, in the following figure, the object type and instance ID for DGADMIN.PRESENT are concatenated in the source column of the

FLG.ATCHREL table. The concatenated object type and instance ID of the associated comment attached to the presentation object in DGADMIN.PRESENT are stored in the target column.

FLG.COMMENTS

OBJTYPID	INSTIDNT	Name	UPDATIME	UPDATEBY	SHRTDESC	...	...	...
000001	0000016465	Comment for "My Presentation" object	...	...	...	...	...	...
000001	0000064459	this is comment3	...	...	...	...	...	...
000001	0000003435	This is a comment for the XYZ presentation	...	...	...	...	...	...

FLG.ATCHREL

RELTYPE	SOURCE	TARGET
A	0000020000001111	0000010000016465
A	0000020000002222	0000010000003435
A	0000030000123456	0000010000004459

DGADMIN.PRESENT

OBJTYPID	INSTIDNT	Name	UPDATIME	UPDATEBY	SHRTDESC	...	...
000002	0000001111	My presentation	...	...	This is a presentation object	...	...
000002	0000002222	XYZ presentation	...	...	This is another presentation object in the information catalog	...	...

Figure 59. Relationship between FLG.ATCHREL table, source, and target

7. The relationship between each pair of tables is derived from the FLGID of the tables. The FLGID represents the concatenation of the OBJTYPID column and the INSTIDNT column of the tables.
8. The relationship stored in FLG.RELINST is for the following relationships: Contains, Link, and Contact. The relationship is derived from the FLGID columns of the source table and the target table.

9. The relationship between each pair of tables is derived from the FLGID of the two tables. There might be multiple rows of data in the FLG.OVERDESC table. If so, the rows are sequenced by the SEQNO column of the FLG.OVERDESC table.

Using SQL to access metadata in the Data Warehouse Center

You can use SQL to extract metadata directly from the database tables that make up the information catalog. This topic provides examples.

Prerequisites:

Before you determine the property names for a specific object type, you must first determine the object type definition, and then determine the property names for a specific object type.

To determine what object type definitions exist in the information catalog, enter the following SQL statement:

```
SELECT OBJTYPID, DPNAME, NAME, CREATOR, PTNAME FROM FLG.OBJTYREG
```

This statement returns the following information:

OBJTYPID
Internal identifier for the object type

DPNAME
Object type name

NAME
External object type name

CREATOR,PTNAME
The table (object instance table) where object instances of that type are stored

To determine the property names for a specific object type after you determine the object type ID, enter the following SQL statement:

```
SELECT  PHYPRPNM, PROPNAME, DATATYPE, LENGTH, OPTIONS, UUISEQNO,
  PROPSEQ FROM FLG.PROPERTY WHERE OBJTYPID = 'object_type_ID'
  ORDER BY PROPSEQ
```

This statement returns the following information (in the order that the properties were created):

PHYPRPNM
Physical column name in the object instance table that maps to an object type property

PROPNAME
Business name of the property

DATATYPE
Data type of the property

LENGTH
Length of the property

OPTIONS
Indicates whether a value is required for this property in the object instance

UUISEQNO
UUI indicator, and sequence number if not 0

PROPSEQ
The order that the properties were added to the properties table

Procedure:

To find an instance of a specific pobject type after you have determined both the physical tables where the object is stored and the properties that you want, enter the following SQL statement:

```
SELECT OBJTYPID, INSTIDNT, NAME,phyprpnm1,phyprpnm2...
  FROM creator.ptname
  WHERE phyprpnm LIKE '%search_criteria%'
```

This statement returns the following information:

OBJTYPID
Internal identifier for the object type

INSTIDNT
Internal identifier for an instance of this object type

phyprpnm1
Value for the property specified in the SELECT statement

phyprpnm2
Value for the property specified in the SELECT statement

In addition, you must enter the following SELECT statement to retrieve any property values that are of the data type long variable character (LONG VARCHAR):

```
SELECT PHYPRPNM, ODESC FROM FLG.OVERDESC
  WHERE OBJTYPID = object_type_ID
  AND INSTIDNT = object_instance_ID
  ORDER BY SEQNO
```

Where object_type_ID and object_instance_ID are the values that you obtained after you generated the SELECT statement. This statement returns the following information:

PHYPRPNM
Physical property name of the property that is a long variable character

ODESC
Value of the long variable character (there might be more than one ODESC for each property value; the order is by sequence)

To retrieve a list of all objects in the information catalog, enter the following SQL statement:

```
SELECT FLGID, INSTNAME, TYPENAME FROM FLG.NAMEINST
```

This statement returns the following information:

FLGID
Concatenated object type and instance IDs for the object

INSTNAME
External name of the object

TYPENAME
Type of object (external name for the object type)

To determine hierarchical or contact relationships between objects, enter the following statement:

```
SELECT SOURCE, TARGET, RELTYPE FROM FLG.RELINST
```

This statement returns the following information:

SOURCE
Concatenated object type and instance ID for the object that is the source in a relationship

TARGET
Concatenated object type and instance ID for the object that is the target of a relationship

RELTYPE
Relationship type (C for container or T for contact)

To determine linked or attachment relationships between objects, enter the following SQL statement:

```
SELECT SOURCE, TARGET, RELTYPE FROM FLG.ATCHREL
```

This statement returns the following information:

SOURCE
Concatenated object type and instance ID for the object that is the source in a relationship

TARGET
Concatenated object type and instance ID for the object that is the target of a relationship

RELTYPE
Relationship type (A for attachment or L for linked)

You can use the SOURCE and TARGET values to look up the object instance information in the object tables. You can also qualify an SQL statement to select specific object values.

Sample: SQL metadata for the Data Warehouse Center

Example: You have an application for which you want to display the metadata about a relational table named Employee, and show all of its columns. The object type for Employee is TABLES, and the object type for the columns is COLUMN. Your application includes the following SQL statements:

1. To retrieve the name of the table where TABLES object instances are stored:

   ```
   SELECT OBJTYPID, DPNAME, NAME, CREATOR, PTNAME FROM FLG.OBJTYREG
   WHERE DPNAME = 'TABLES'
   ```

   The statement returns the following information:

   ```
   '000001', 'TABLES', 'Relational Tables', 'USERXYZ', 'TABLES'
   ```

2. To retrieve the OBJTYPID of the COLUMN object:

   ```
   SELECT OBJTYPID, DPNAME, CREATOR, PTNAME from FLG.OBJTYREG
   WHERE DPNAME = 'COLUMN'
   ```

   The statement returns the following information:

   ```
   '000007', 'COLUMN', 'Columns or fields', 'USERXYZ', 'COLUMN'
   ```

3. To retrieve the information about the specific TABLES object for which you want to display metadata:

   ```
   SELECT OBJTYPID, INSTIDNT, NAME, DBNAME, OWNER, TABLES
     FROM USERXYZ.TABLES
     WHERE NAME = 'Employee'
   ```

   The statement returns the following information:

   ```
   '000001', '0040608795',  'Employee', 'MYDBASE', 'USERABC', 'EMPL_TAB'
   ```

4. To retrieve the relationships between the TABLES instance SOURCE and COLUMN instance TARGET:

```
SELECT TARGET FROM FLG.RELINST
WHERE SOURCE = '0000010040608795'
  AND TARGET LIKE '000007%'
  AND RELTYPE = 'C'
```

The statement returns the following two objects:

```
('0000079238400354')
('0000079843095410')
```

5. To retrieve the information about the two returned COLUMN objects:

```
SELECT NAME, SHRTDESC, DATATYPE, LENGTH FROM USERXYZ.COLUMNS
WHERE INSTIDNT IN ('9238400354', 9843095410')
```

The statement returns the following information:

```
('Name', 'Employee name information', 'CHAR', '80')
('Address', 'Employee address information', 'CHAR', '220')
```

Appendix B. Template planning worksheet

Use this worksheet to collect the values that your partner application needs to provide.

Write the value of the token in the table. For tokens that have a specific list of allowed values, circle one of the allowed values.

Table 131. Tokens for required metadata in the templates

Token	Value
**AgentSite*	
**AgentSiteContact*	
**AgentSiteDescription*	
**AgentSiteNotes*	
**AgentSiteOSType*	One of the following values: **ISV_windowsNT** Windows NT **ISV_AIX** AIX **ISV_os2** OS/2 **ISV_as400** AS/400 **ISV_Solaris** SUN **ISV_MVS** MVS
**AgentSiteTCPIPHostName*	
**AgentSiteUserid*	
**ColumnAllowsNulls*	One of the following values: **ISV_NULLSYES** The column allows null data. **ISV_NULLSNO** The column does not allow null data.

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
***ColumnDataIsText*	One of the following values: **ISV_ISTEXTYES** The column contains only text data. **ISV_ISTEXTNO** The column does not contain only text data.
***ColumnDescription*	
***ColumnEditionType*	One of the following values: **ISV_ColumnIsEditionColumn** The column is an edition column. **ISV_ColumnIsNormal** The column is a normal column.
***ColumnKeyPosition*	
***ColumnLength*	
***ColumnName*	

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**ColumnNativeDataType*	One of the following values: ISV_NATIVE_CHAR ISV_NATIVE_VARCHAR ISV_NATIVE_LONGVARCHAR ISV_NATIVE_VARCHAR2 ISV_NATIVE_GRAPHIC ISV_NATIVE_VARGRAPHIC ISV_NATIVE_LONGVARGRAPHIC ISV_NATIVE_CLOB ISV_NATIVE_INT ISV_NATIVE_TINYINT ISV_NATIVE_BLOB ISV_NATIVE_SMALLINT ISV_NATIVE_INTEGER ISV_NATIVE_FLOAT ISV_NATIVE_SMALLFLOAT ISV_NATIVE_DOUBLE ISV_NATIVE_REAL ISV_NATIVE_DECIMAL ISV_NATIVE_SMALLMONEY ISV_NATIVE_MONEY ISV_NATIVE_NUMBER ISV_NATIVE_NUMERIC ISV_NATIVE_DATE ISV_NATIVE_TIME ISV_NATIVE_TIMESTAMP ISV_NATIVE_LONG ISV_NATIVE_RAW ISV_NATIVE_LONGRAW ISV_NATIVE_DATETIME ISV_NATIVE_SMALLDATETIME ISV_NATIVE_SYSNAME ISV_NATIVE_TEXT ISV_NATIVE_BINARY

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**ColumnNativeDataType* (continued)	One of the following values: ISV_NATIVE_VARBINARY ISV_NATIVE_LONGVARBINARY ISV_NATIVE_BIT ISV_NATIVE_IMAGE ISV_NATIVE_SERIAL ISV_NATIVE_DATETIMEYEARTOFRACTION ISV_NATIVE_DBCLOB ISV_NATIVE_BIGINT
**ColumnNotes*	
**ColumnOffsetFromZero*	
**ColumnOrdinalNumber*	
**ColumnPositionNumber*	
**ColumnPrecision*	
**ColumnUserActions*	
**CurrentCheckPointID++*	
**DatabaseContact*	
**DatabaseDescription*	
**DatabaseName*	
**DatabaseNotes*	
**DatabasePhysicalName*	

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**DatabaseType*	One of the following values: **ISV_IR_DB2Family** DB2 Family **ISV_IR_Oracle** Oracle **ISV_IR_Sybase** Sybase **ISV_IR_MSSQLServer** Microsoft SQLServer **ISV_IR_Informix** Informix **ISV_IR_GenericODBC** Generic ODBC **ISV_IR_FFLan** Flat File LAN **ISV_IR_VSAM** VSAM **ISV_IR_IMS** IMS
**DatabaseTypeExtended*	One of the following values: **ISV_IR_DB2400CISC** DB2 UDB for AS/400® for CISC **ISV_IR_DB2400RISC** DB2 UDB for AS/400 for RISC **ISV_IR_FFLanLocalCmd** Local flat file **ISV_IR_FFLanFTPCopy** Local flat file sent using FTP from a remote system
**DatabaseServerName*	
**DatabaseUserid*	
**DatabaseVersion*	
**PostStepName*	
**ProcessContact*	
**ProcessDescription*	
**ProcessName*	

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**ProcessNotes*	
**ProcessType*	One of the following values: **ISV_ProcessType_Normal** Process is a normal user process. **ISV_ProcessType_Meta_pub** Process is a metadata publication process. **ISV_ProcessType_Notify** Process is a notification process.
**SecurityGroup*	ISV_DEFAULTSECURITYGROUP
**StarSchemaContact*	
**StarSchemaDBName*	
**StarSchemaDescription*	
**StarSchemaName*	
**StarSchemaNotes*	
**StepCommit*	One of the following values: **ISV_Step_Incremental_Commit_On** The data is to be incrementaly commited at the target. **ISV_Step_Incremental_Commit_Off** The data is not to be incrementaly commited at the target.
**StepCommitAfterNumberRows*	
**StepContact*	
**StepDataNotPresent*	One of the following values: **ISV_StepDataNotPresent_OK** If data is not present, continue processing. **ISV_StepDataNotPresent_Warning** If data is not present, issue a warning and continue processing. **ISV_StepDataNotPresent_Error** If data is not present, issue an error message and stop processing.
**StepDescription*	

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**StepExternalPopulation*	One of the following values: **ISV_StepExternalNo** The table will not be externally populated by other means. **ISV_StepExternalYes** The table will be externally populated by other means.
**StepName*	
**StepNotes*	
**StepSelectStatement*	
**StepSelectStatementGenerated*	One of the following values: **ISV_StepSelectStatementNo** The SELECT statement is not generated, but is included in the **StepSelectStatement*. **ISV_StepSelectStatementYes** The SELECT statement is generated, and **StepSelectStatement* is ignored.
**StepSQLWarning*	One of the following values: **ISV_StepSQLWarning_OK** If an SQL warning occurs, continue processing. **ISV_StepSQLWarning_Warning** If an SQL warning occurs, issue a warning and continue processing. **ISV_StepSQLWarning_Error** If an SQL warning occurs, issue an error and stop processing.

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**StepType*	One of the following values: **ISV_StepType_Editioned_Append** The data in the table will be appended when the Step is run. **ISV_StepType_Full_Replace** The data in the table will be replaced when the Step is run. **ISV_StepType_Uneditioned_Append** The data in the table will be appended when the Step is run. **ISV_StepType_VWP_Population** The data in the table is populated by a Data Warehouse Center program.
**SubjectArea*	
**SubjectAreaContact*	
**SubjectAreaDescription*	
**SubjectAreaNotes*	
**TableBinaryIfFile*	One of the following values: **ISV_DR_FILE_IS_BINARY** The file is binary. **ISV_DR_FILE_IS_NOT_BINARY** The file is in ASCII or mixed format.
**TableCreatedByDWC*	One of the following values: **ISV_TableIsToBeCreatedByDWC** The table is to be created by the Data Warehouse Center. **ISV_TableIsNotToBeCreatedByDWC** The table is not to be created by the Data Warehouse Center.
**TableCreateStatement*	
**TableDelimiterIfFile*	
**TableDescription*	
**TableFirstRowNamesIfFile*	One of the following values: **ISV_DR_ROW_CONTAINS_NAMES** The first row of the file contains column names.

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
	ISV_DR_ROW_DOES_NOT_CONTAIN_NAMES The first row of the file contains data.
**TableFullName*	
**TableGenerateCreateStatement*	One of the following values: **ISV_GenerateCreateTableStmt** The Data Warehouse Center should generate the CREATE TABLE statement. **ISV_DoNotGenerateCreateTableStmt** The Data Warehouse Center should not generate the CREATE TABLE statement.
**TableGrantedToPublic*	One of the following values: **ISV_GrantTableAccessToPublic** Grant PUBLIC access to this table. **ISV_DoNotGrantTableAccessToPublic** Do not grant PUBLIC access to this table.
**TableIsAnAlias*	One of the following values: **ISV_TableIsAnAlias** This table is an alias for another table. **ISV_TableIsNotAnAlias** This table is not an alias for another table.
**TableIsADimensionTable*	One of the following values: **ISV_TableIsADimensionalTable** The table is a dimensional table. **ISV_TableIsNotADimensionalTable** The table is not a dimensional table.
**TableIsAFactTable*	One of the following values: **ISV_TableIsAFactTable** The table is a fact table. **ISV_TableIsNotAFactTable** The table is not a fact table.

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**TableIsAView*	One of the following values: **ISV_TableIsAView** The table is a view. **ISV_TableIsNotAView** The table is not a view.
**TableIsPersistent*	One of the following values: **ISV_TableIsPersistent** The table is to be considered persistent. **ISV_TableIsTransient** The table is to be considered transient.
**TableMaximumEditions*	
**TableNotes*	
**TableOwner*	
**TablePhysicalName*	
**TableTypeIfFile*	One of the following values: **ISV_DR_REL_TABLE** The table is a relational table. **ISV_DR_COMMA_DELIMITED** The columns in the file are separated by commas. **ISV_DR_FIXED_FORMAT** The columns in the file are in fixed format. **ISV_DR_TAB_DELIMITED** The columns in the file are separated by tabs. **ISV_DR_CHAR_DELIMITED** The columns in the file are separated by the value of **TableDelimiterIfFile*.
**VWPGroup*	
**VWPGroupDescription*	
**VWPGroupNotes*	
**VWPProgramInstanceKey*	
**VWPProgramInstanceParameterData*	
**VWPProgramInstanceParameterKey*	

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**VWPProgramInstanceParameterName*	
**VWPProgramInstanceParameterOrder*	
**VWPProgramInstanceParameterType*	One of the following values: **ISV_ParameterTypeNone** The parameter type is unknown. **ISV_ParameterTypeCharacter** The parameter type is character. **ISV_ParameterTypeNumeric** The parameter type is numeric. **ISV_ParameterTypePassword** The parameter type is password.
**VWPProgramTemplateDescription*	
**VWPProgramTemplateExecutableName*	
**VWPProgramTemplateFunctionName*	
**VWPProgramTemplateName*	
**VWPProgramTemplateNotes*	
**VWPProgramTemplateType*	One of the following values: **ISV_PROGRAMTYPEDLL** The Data Warehouse Center program is loaded from a dynamic link library (DLL) or is a load module. **ISV_PROGRAMTYPECOMMAND** The Data Warehouse Center program is a command file. **ISV_PROGRAMTYPEEXECUTABLE** The Data Warehouse Center program is an executable file.
**VWPProgramTemplateParameterData*	
**VWPProgramTemplateParameterKey*	
**VWPProgramTemplateParameterName*	
**VWPProgramTemplateParameterOrder*	

Table 131. Tokens for required metadata in the templates (continued)

Token	Value
**VWPProgramTemplateParameterType*	One of the following values: **ISV_ParameterTypeNone** The parameter type is unknown. **ISV_ParameterTypeCharacter** The parameter type is character. **ISV_ParameterTypeNumeric** The parameter type is numeric. **ISV_ParameterTypePassword** The parameter type is password.

Appendix C. Writing your own program to use with the Data Warehouse Center

You can write Data Warehouse Center programs in any language that supports one of the following program types: executable, batch program, or dynamic link library.

If the program has a program type of executable, command file, or dynamic link library, it must reside on the agent site. The Data Warehouse Center agent starts the program at the scheduled time. On Windows operating systems, the agent runs as a system process by default. The program cannot access resources or programs that require a user ID. Also, any environment variables that the program needs to access must be system variables.

Parameter passing

At run time, the Data Warehouse Center generates a command-line parameter list that it passes as input to your program. Whenever possible, test your program from the command line before using it in a step.

Example: The Data Warehouse Center program VW 5.2 DB2 load replace (VWPLOADR) selects data from a file and loads the data into a database. It uses the following parameters:

- Source file name
- Target database name
- Target database user ID
- Target database password
- Target table name
- Column delimiter

The program gets the parameters as shown in The following figure:

```
char * sourceFile;
     sourceFile = argv[1]:
     char * dbName;
     dbName = argv[2];
     char * dbUser;
     dbUser = argv[3];
     char * dbPassword
     dbPassword = argv[4];
     char * dbTable;
     dbTable = argv[5]
     char * fileMod;
     if(argc>6) fileMod = argv[6];
     else fileMod = NULL;
```

Figure 60. Reading parameters from the command line

The program uses the target parameters to connect to the target database, as shown in Figure 61:

```
rc = SQLConnect (hdbc, (SQLCHAR *)dbName, SQL_NTS,
           (SQLCHAR *)dbUser, SQL_NTS,       /* UID */
           (SQLCHAR *)dbPassword, SQL_NTS);  /* Password */
```

Figure 61. Connecting to the target database

The program then uses the DB2 load utility to load data into the database.

Status information return

After your Data Warehouse Center program runs, it must return a return code to the step that uses the program. The return code must be a positive integer. If your program does not return a return code, the step using the program fails. The Data Warehouse Center displays the return code in the **Error RC2** field of the Log Details window when the value of **Error RC1** is 8410.

Your Data Warehouse Center program can return additional status information to the Data Warehouse Center:

- Another return code, which can be the same as or different from the code that is returned by the Data Warehouse Center program.
- A warning flag that indicates that the Data Warehouse Center is to treat the return code as a warning. When your program sets this flag, the step that uses this program will have Warning status in the Operations Work in Progress window.
- A message, which is displayed in the **System Message** field of the Log Viewer Details window
- The number of rows of data that the program processed.

The Data Warehouse Center displays the number in the Log Viewer Details window for the step.

- The number of bytes of data that the program processed.

  The Data Warehouse Center displays the number in the Log Viewer Details window for the step.
- The SQLSTATE return code, which the Data Warehouse Center displays in the SQL state field of the Log Viewer Details window.

The Data Warehouse Center agent transfers the additional status information to the warehouse server.

Transferring the information to the Data Warehouse Center

To transfer the additional status information to the warehouse agent, your program must create a file, called a *feedback file*, containing the additional status information. The path and file name for the feedback file must be the value of the VWP_LOG environment variable. The agent sets VWP_LOG before it calls the program. After the program finishes running, the agent checks whether the feedback file exists. If it exists, the agent processes the file. Otherwise, the agent will do nothing. If the program cannot create the file, it should continue to run.

Format of the feedback file

Your program can write the additional status information to the feedback file in any order, but must use the following format to identify information. Enclose each returned item within the begin tag <tag> and end tag </tag> in the following list. Each begin tag must be followed by its end tag; you cannot include two begin tags in a row. For example, the following tag format is valid:

<RC>...</RC>...<MSG>...</MSG>

The following embedded tag format is not valid:

<RC>...<MSG>...</RC>...</MSG>

You can specify the following information in the feedback file:

Return code
<RC>*return code*</RC>, where *return code* is a positive integer.

Return code warning flag
<WARNING>1</WARNING> sets the return code warning flag to On.

Data Warehouse Center system message
<MSG>*message text*\n</MSG>

message text
The text of one or more messages

\n The new line character. Include this character at the end of each message if there are multiple messages.

Comment
<COMMENT>*comment text*</COMMENT>, where *comment text* is the text of the comment.

Number of rows of data processed
<ROWS>*number of rows*</ROWS>, where *number of rows* is any positive integer.

Number of bytes processed
<BYTES>*number of bytes*</BYTES>, where *number of bytes* is any positive integer.

SQLSTATE
<SQLSTATE>*sqlstate string*</SQLSTATE>, where *sqlstate string* is any string whose length is greater than 0 and less than or equal to 5 digits.

The following figure shows an example of the feedback file.

```
<RC> 20</RC>
<ROWS>2345</ROWS>
<MSG>The parameter type is not correct</MSG>
<COMMENT> Please supply the correct parameter type (PASSWORD
     NOTREQUIRED, GETPASSWORD, ENTERPASSWORD)</COMMENT>
<BYTES> 123456</BYTES>
<WARNING> 1</WARNING>
<SQLSTATE>12345</SQLSTATE>
```

Figure 62. Example of the feedback file

How the feedback determines the step status

The return codes and step status for the program that are displayed in the Log Viewer vary. They depend on the following values set by the program:

- The value of the return code that the program returned
- Whether a feedback file exists
- The value of the return code in the feedback file
- Whether the warning flag is set to On

The following table lists the possible combinations of these values and the results that they produce.

Table 132. Feedback file conditions and results

Conditions				**Results**	
				Step status[1]	**Values of Error RC1 and RC2**
Data Warehouse Center program return code is 0	No feedback file exists[2]			Successful	**RC1** = 0; **RC2** = 0
	A feedback file exists[2]	The value of <RC> in the feedback file is 0[3]	<WARNING> is not set in the feedback file	Successful	**RC1** = 0; **RC2** = 0
			The value of <WARNING> in the feedback file is 1	Warning	**RC1** = 0; **RC2** = 0
		The value of <RC> in the feedback file is non-0[3]	<WARNING> is not set in the feedback file	Failed	**RC1** = 8410 (the program failed); **RC2** = the value of <RC> in the feedback file
			The value of <WARNING> in the feedback file is 1	Warning	**RC1** = 0; **RC2** = the value of <RC> in the feedback file

Table 132. Feedback file conditions and results (continued)

Conditions				**Results**	
				Step status[1]	**Values of Error RC1 and RC2**
The Data Warehouse Center program return code is nonzero	No feedback file exists[2]			Failed	**RC1** = 8410 (the Data Warehouse Center program failed); **RC2** = the code returned by the Data Warehouse Center program
	A feedback file exists[2]	The value of <RC> in the feedback file is 0[3]	<WARNING> is not set in the feedback file	Successful	**RC1** = 0; **RC2** = 0
			The value of <WARNING> in the feedback file is 1	Warning	**RC1** = 0; **RC2** = 0
		The value of <RC> in the feedback file is non-0	<WARNING> is not set in the feedback file	Failed	**RC1** = 8410 (the Data Warehouse Center program failed); **RC2** = the code returned by the Data Warehouse Center program
			The value of <WARNING> in the feedback file is 1	Warning	**RC1** = 0; **RC2** = the value of <RC> in the feedback file

Table 132. Feedback file conditions and results (continued)

Conditions	Results	
	Step status[1]	**Values of Error RC1 and RC2**

Notes:

1. The step processing status, which is displayed in the Work in Progress window.
2. The Data Warehouse Center checks for the existence of the feedback file, regardless of whether the return code for the program is 0 or nonzero.
3. The Data Warehouse Center always displays the value of <RC> in the feedback file as the value of the **RC2** field in the Log Details window.

Notices

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

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

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

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

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

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

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

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

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

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

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

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

COPYRIGHT LICENSE:

This information may contain sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both, and have been used in at least one of the documents in the DB2 UDB documentation library.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 UDB documentation library:

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

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

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Bibliography

For information about how to use the Data Warehouse Center, see the online help. The Data Warehouse Center provides help for specific windows and for general tasks, such as creating warehouse sources and steps.

For information about IBM products that are related to the Data Warehouse Center, go to the IBM Data Management Web site at http://www.software.ibm.com/data/

The Data Warehouse Center library includes the following publications:

IBM DB2: DB2 Warehouse Manager Installation Guide, gc27-1122
IBM DB2: Information Catalog Center Administration Guide, SC27-1125
IBM DB2 OLAP Server: Using DB2 OLAP Server, SC26-9235

Index

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-237-5511 for customer service
- 1-888-426-4343 to learn about available service options
- 1-800-IBM-4YOU (426-4968) for DB2 marketing and sales

In Canada, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-800-465-9600 to learn about available service options
- 1-800-IBM-4YOU (1-800-426-4968) for DB2 marketing and sales

To locate an IBM office in your country or region, check IBM's Directory of Worldwide Contacts on the web at www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, client downloads, newsgroups, FixPaks, news, and links to web resources.

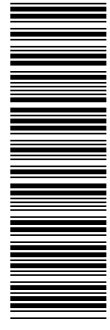
If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at www.ibm.com/planetwide

Part Number: CT176NA

Printed in U.S.A.



(1P) P/N: CT176NA

SC27-1124-00

Spine information:

IBM® DB2 Universal Database™

Data Warehouse Center Application Integration Guide

Version 8